Neural mechanisms of information processing and transmission

Dissertation
zur Erlangung des Grades eines Doktors der Naturwissenschaften
eingereicht am Fachbereich Humanwissenschaften
der Universität Osnabrück

vorgelegt von
Johannes Leugering
Osnabrück, Januar 2021

Neural mechanisms of information processing and transmission

Dissertation
for a doctoral degree in natural sciences
submitted to the School of Human Sciences
of Osnabrück University

by
Johannes Leugering
Osnabrück, January 2021

Contents

Abstract

English

This (cumulative) dissertation is concerned with mechanisms and models of information processing and transmission by individual neurons and small neural assemblies. In this document, I first provide historical context for these ideas and highlight similarities and differences to related concepts from machine learning and neuromorphic engineering. With this background, I then discuss the four main themes of my work, namely dendritic filtering and delays, homeostatic plasticity and adaptation, rate-coding with spiking neurons, and spike-timing based alternatives to rate-coding. The content of this discussion is in large part derived from several of my own publications included in appendix C, but it has been extended and revised to provide a more accessible and broad explanation of the main ideas, as well as to show their inherent connections. I conclude that fundamental differences remain between our understanding of information processing and transmission in machine learning on the one hand and theoretical neuroscience on the other, which should provide a strong incentive for further interdisciplinary work on the domain boundaries between neuroscience, machine learning and neuromorphic engineering.

Deutsch

Diese (kumulative) Dissertation behandelt Mechanismen und Modelle der Informationsverarbeitung und -übertragung durch einzelne Neuronen sowie kleine neuronale Assemblies. In diesem Dokument stelle ich erst den historischen Kontext dieser Ideen dar, und zeige Gemeinsamkeiten und Unterschiede zu verwandten Ansätzen beim maschinellen Lernen und Neuromorphic Engineering auf. Vor diesem Hintergrund entwickele ich im Anschluss die vier Kernthemen meiner Arbeit: dendritische Filterung und Delays, homeostatische Plastizität und Adaption, Ratencodierung durch gepulste Neuronen, sowie spike-timing-basierte Alternativen zur Ratencodierung. Der Inhalt dieser Darstellung basiert im Wesentlichen auf mehreren eigenen Publikationen, welche im Appendix C angehängt sind, er wurde allerdings weiterentwickelt und ergänzt um die Kernideen einfacher zugänglich zu machen, umfassender zu erklären und ihre inhaltlichen Verbindungen herauszustellen. Ich schließe die Diskussion mit der Schlussfolgerung ab, dass nach wie vor fundamentale Unterschiede in unserem Verständnis von Informationsverarbeitung und -übertragung bei maschinellem Lernen auf der einen, und theoretischen Neurowissenschaften auf der anderen Seite bestehen, die einen starken Anreiz für weitere interdisziplinäre Arbeiten im Grenzbereich zwischen Neurowissenschaften, maschinellem Lernen und Neuromorphic Engineering bieten sollten.

Preface

What this thesis is about

In this dissertation, I talk about several aspects of neural information processing that I believe to be very important for biological systems, but which are often overlooked or under-appreciated in models of (artificial) neurons. The topics of this thesis are therefore situated between the fields of theoretical neuroscience, machine learning and neuromorphic hardware. In order to explain the similarities and differences between these fields, the first two chapters offer a brief historical perspective of how they came to be (chapter 1), and how each of them understands and uses (artificial) neurons and networks today (chapter 2). Chapter 3 gives a brief tour of the field of *neuromorphic hardware* — my application domain for concepts from theoretical neuroscience. In each of the subsequent chapters, I then address one important aspect of neural computation, i.e. computing with dendritic filters and delays in chapter 4, improving computation with homeostatic plasticity in chapter 5, rate-coding with spiking neurons in chapter 6, and finally spike-timing and event-based computation in chapter 7.

What this thesis is not about

It is impossible for me to give a full account of all the topics related to neural information processing in one thesis, and even for the topics that I want to discuss, there is a large host of prior work that is better summarized elsewhere. For those who are interested in a deeper discussion of these topics as well as the historical context, I can highly recommend the books by Rosenblatt [1], Ashby [2], Maass and Bishop [3], Turing and Copeland [4], Eliasmith and Anderson [5], Laughlin [6], and Stone [7].

During my time in the *Neuroinformatics* lab, I also worked on other topics in machine learning and statistical modeling that I have decided to not incorporate into this thesis, since they are thematically disconnected. These include:

- Joint work with Olivera Stojanovic and the Robert-Koch-Institute on a Bayesian spatio-temporal model of the spread of infectious diseases [8]. I have attached the text of this publication on pages 236ff, but I will not address its content here.
- Joint work with Kristoffer Appel, among others, on the creation of the TRAUMSCHREIBER, a low-power mobile EOG/ECG/EMG/EEG device for polysomnography [9], as well as a software-stack to go with it and a block-course on wearable electronics.

- Joint work with Pascal Nieters, the German Meteorological Service and others on a model to predict precipitation using deep learning [10].
- The contents of a lecture series on *Ensemble methods for machine learning*, developed and held with Olivera Stojanovic in the summer term of 2017.
- The supervision of 16 Bachelor's and 7 Master's theses and several student projects on various topics.

What are the main scientific contributions within this thesis?

Most of the chapters in this thesis summarize ideas that are explored in depth in some corresponding publication(s), which I have attached as pages 99 and following. These contributions are the following three journal papers[1], two peer-reviewed conference papers, one book chapter, one patent[2], one non-peer-reviewed article and one conference poster, each of which is introduced in more detail in the corresponding chapter(s):

[1] The first two are published in peer-reviewed journals, the third has only been published as a pre-print and submitted for review.
[2] The patent has been filed and is currently pending.

1. P. Nieters, **J. Leugering**, and G. Pipa, "Neuromorphic computation in multi-delay coupled models," *IBM Journal of Research and Development*, vol. 61, no. 2/3, 8:7–8:9, 1, 2017, ISSN: 0018-8646, 0018-8646. DOI: 10.1147/JRD.2017.2664698.
2. **J. Leugering** and G. Pipa, "A Unifying Framework of Synaptic and Intrinsic Plasticity in Neural Populations," *Neural Computation*, vol. 30, no. 4, pp. 945–986, 17, 2018, ISSN: 0899-7667. DOI: 10.1162/neco_a_01057.
3. **J. Leugering**, P. Nieters, and G. Pipa, "Event-based pattern detection in active dendrites," *bioRxiv*, p. 690 792, 17, 2020. DOI: 10.1101/690792.
4. F. Meyer zu Driehausen, R. Busche, **J. Leugering**, and G. Pipa, "Bistable Perception in Conceptor Networks," in *Artificial Neural Networks and Machine Learning – ICANN 2019: Workshop and Special Sessions*, 2019, ISBN: 978-3-030-30493-5. DOI: 10.1007/978-3-030-30493-5_3.
5. **J. Leugering**, "Making spiking neurons more succinct with multi-compartment models," in *Proceedings of the Neuro-Inspired Computational Elements Workshop*, 17, 2020, ISBN: 978-1-4503-7718-8. DOI: 10.1145/3381755.3381763.
6. **J. Leugering**, P. Nieters, and G. Pipa, "Computational Elements of Circuits," in *The Neocortex*, W. Singer, T. J. Sejnowski, and P. Rakic, eds., red. by J. Lupp, vol. 27, The MIT Press, 2019, pp. 195–209, ISBN: 978-0-262-04324-3.
7. **J. Leugering**, P. Nieters, and G. Pipa, "Neuromorpher Musterdetektor und neuromorphe Schaltkreisanordnung hiermit," pat. pending.
8. **J. Leugering**, "A visit to the neuromorphic zoo," in *Embedded World Conference 2020 – Proceedings*, 2020, ISBN: 978-3-645-50186-6.
9. P. Nieters, **J. Leugering**, and G. Pipa, "Neuromorphic Adaptive Filters for event detection, trained with a gradient free online learning rule," presented at the Machine Learning Summer School (MLSS-Africa 2019), 1, 2019.

But some content is also new, or at least not covered by my own publications. In particular, chapters 4 and 6 contain work that motivated me to pursue the ideas of chapter 7, but ultimately did not directly appear in any of my publications yet. I have therefore decided to include some of this additional content in appendices A and B, respectively, in the hope that it will help to keep the rest of the text concise.

The main body of this thesis is intended to provide a more accessible summary of these publications, to highlight the links between various topics, and to embed them into the bigger picture that has motivated my work. Since I have compiled this thesis over a long time-span, some of my views have also evolved, and I chose to introduce some of these older ideas in a new, hopefully clearer way. In some places, this has revealed some new interesting connections that were not explored in the original work.

Who should read this thesis?

Naturally, I hope the PhD committee will like this text, but I'm writing this with a different audience in mind, as well. Over the last few years, I have recognized more and more underappreciated similarities between theoretical neuroscience on the one hand, and engineering fields like electronics, signal processing and communication systems on the other — both in terms of what questions are asked ("How much information can be transmitted over this kind of channel? Is a pulse-based code effective? How can I realize this computation with these components?"), and in terms of the tools and models used to answer these questions (information theory, signal processing, dynamical systems, control theory, etc.). Similarly, I think that a lot of the early results of cybernetics and connectionism are often overlooked today; but reading papers and books by Minsky and Papert, Ashby, Turing, von Neumann, Rosenblatt and others shows how many of the seemingly revolutionary ideas of the last few years are already implied there! In particular at the fringes where these different fields meet, namely neuromorphic hardware, the close connection and shared history between theoretical neuroscience, computer science, machine learning and engineering becomes obvious. It is therefore not a coincidence that some of the most inspiring books I have read during my time as a PhD student are actually rooted in engineering disciplines. I have tried to follow their example, giving this text a bit of an engineering flavor.

Hopefully, the high-level descriptions given here make these results more accessible than the original publications (which were rather specifically written for other neuroscientists) and thus also prove useful for scientists and engineers from different fields, e.g. neuromorphic hardware designers or machine learning researchers, who are interested in abstract models of neural information processing mechanisms.

Johannes Leugering
Nürnberg,
2020

Ancillary material is available in this code repository:
`https://github.com/jleugeri/phd`

Acknowledgments

During my time in the Prof. Pipa's Neuroinformatics group I was given an unusual amount of freedom to work on diverse topics that really interested me, most of which did not find a way into this thesis. But these apparently "unproductive" activities, too, played an important role for me, because they offered new perspectives and insights that came in handy in completely unforeseen circumstances. Today, I probably wouldn't be working on neuromorphic hardware, were it not for all the hours I had to invest into the design of the TRAUMSCHREIBER! This transition would not have been possible in an environment that singularly values publication metrics rather than curiosity.

I am therefore grateful to our dean of studies at the time, Prof. Achim Stephan, for fostering such a liberating studying environment, to my colleagues and forebears for creating a pleasant and inspiring working environment, to Anna Rushing-Jungeilges for ironing out any and all tensions, and to my supervisor Prof. Gordon "Entropy" Pipa for caring more about content than form and leaving me the freedom to find my own way. Last but not least, I'm thankful to my family and my girlfriend Olivera for being at my side through the ups and downs during these years!

It was the best of times, it was the worst of times,
it was the age of wisdom, it was the age of foolishness,
it was the epoch of belief, it was the epoch of incredulity,
it was the season of Light, it was the season of Darkness

— *A Tale of Two Cities* by Charles Dickens

The issues that give rise to excitement today seem much the same as those that were responsible for previous rounds of excitement. The issues that were then obscure remain obscure today because no one yet knows how to tell which of the present discoveries are fundamental and which are superficial.

— *Perceptrons — Expanded Edition* by Marvin Minsky and Seymour Papert

1 *The computer and the brain*

When photons hit the retina and cause a neuron to emit a spike, a physical effect becomes information. How is this information represented and processed by the neural network that constitutes the brain? How does it extract structure from its sensory inputs, and learn to adapt to its environment?

These are fundamental questions that have kept generations of scientists and philosophers busy. To answer them, we'll need to thoroughly understand the basic mechanisms at play in neural information processing. The objective of theoretical neuroscience is therefore to identify these principles, from the level of individual neurons and synapses all the way up to networks and brain areas, and to abstract them into theoretical (i.e. mathematical) models, which can be understood without all the overwhelming complexity that has developed over hundreds of millions of years of evolutionary history.

The sudden and rapid development of Deep Learning in the last couple of years might have given many people the impression that we have now finally "cracked the code" of how neural networks work, and that we are on the verge of solving the mystery of the brain and (artificial) intelligence. While this is certainly an exciting perspective, it's important not to forget, that similar claims have been made multiple times before, and the celebration has always turned out to be premature. For example, consider the following brutal assessment by Marvin Minsky and Seymour Papert from the year 1988 and mentally substitute the older term "Connectionism" with its modern counterpart "Deep Learning":

> [...] [L]ittle of significance had changed since 1969, when the book was first published[...]. One reason why progress has been so slow in this field is that researchers unfamiliar with its history have continued to make many of the same mistakes that others have made before them. Some readers may be shocked to hear it said that little of significance has happened in this field. Have not perceptron-like networks — under the new name connectionism — become a major subject of discussion at gatherings of psychologists and computer scientists? Has not there been a "connectionist revolution?" Certainly yes, in that there is a great deal of interest and discussion. Possibly yes, in the sense that discoveries have been made that may, in time, turn out to be of fundamental importance. But certainly no, in that there has been little clear-cut change in the conceptual basis of the field. The issues that give rise to excitement today seem much the same as those that were responsible for previous rounds of excitement. The issues that were then obscure remain obscure today because no one yet knows how to tell which of the present discoveries are fundamental and which are superficial. Our position remains what it was when we wrote the book: We believe this realm of work to be immensely important and rich, but we expect its growth to require a degree of critical analysis that its more romantic advocates have always been reluctant to pursue — perhaps because the spirit of connectionism seems itself to go somewhat against the grain of analytic rigor. [20]

So what has changed since then? Are we about to make the same mistakes again? To get a better understanding of where we stand today, I'd like to start with a bit of historical background of the field(s).

1.1 The origins of computational (neuro-)science and machine learning

The medical study of the central nervous system can be traced back for more than three millennia [21], but the mechanism by which it operates has remained a mystery throughout most of this history. It was only after a series of remarkable scientific discoveries in the 19th century, notably the observation of so-called "animal electricity" [22], advances in microscopy and histology [23], the theory of evolution [24] and the popularization of cell theory [25] that the *neuron doctrine* took root [26] and modern scientific theories of the brain's function began to emerge. In 1943, in the middle of World War II, Warren McCulloch and Walter Pitts wrote a landmark paper *A Logical Calculus of Ideas Immanent in Nervous Activity* [27], in which they first proposed that networks of interconnected nerve cells could implement a powerful symbolic logic calculus. A sufficiently large network of neurons, endowed with the necessary periphery and memory, could therefore satisfy the conditions of a universal machine as outlined just seven years prior by Alan Turing [28]. They write [27]:

> It is easily shown: first, that every net, if furnished with a tape, scanners connected to afferents, and suitable efferents to perform the necessary motor-operations, can compute only such numbers as can a Turing machine; second, that each of the latter numbers can be computed by such a net; and that nets with circles can be computed by such a net; and that nets with circles can compute, without scanners and a tape, some of the numbers the machine can, but no others, and not all of them. This is of interest as affording a psychological justification of the Turing definition of computability and its equivalents, Church's A-definability and Kleene's primitive recursiveness: if any number can be computed by an organism, it is computable by these definitions, and conversely.

This connection between the biological connectivity of neurons and an abstract, mathematical notion of *computation* created a theoretical foundation for the field of *computational neuroscience*. But the concept of computability did not merely provide a language for neuroscientists to *describe* the operation of the brain — it also made it conceivable to *simulate* neural behavior, and therefore *intelligent* behavior, on any appropriate universal machine. Turing became fascinated by this idea and in 1948 wrote a visionary publication entitled *Intelligent Machinery* [29] that today reads like a prescient outline for many subsequent developments in machine learning.[1]

In 1949, Donald Hebb provided the first mechanistically plausible theory of (unsupervised) learning in neural networks, the now famous *Hebbian learning rule*, which in its most explicit form stated that *"[w]hen one cell repeatedly assists in firing another, the axon of the first cell develops synaptic knobs (or enlarges them if they already exist) in contact with the soma of the second cell."* [33] Thus the study of *synaptic plasticity* and *learning* in neural networks was born. Ross Ashby extended this view of self-organization as an essential property of the brain (and life in general), and ultimately proposed in his highly influential 1954 book *Design for a Brain* [2] the "Homeostat", a self-regulating machine, as an example of artificial life.

But since these algorithmic mechanisms could also be simulated by a Turing machine, it now seemed conceivable to simulate intelligent behavior, and, even more interestingly, learning. As Turing himself suggested in private correspondence to Ross Ashby, his *Automatic Computing Engine* (ACE) could be used to that end:

[1] In it, he discussed, for example, not just recurrently connected neural networks, but also proposed randomly initialized networks, which are then trained through reward and punishment, as a reasonable analogy for (some parts of) cortex — a view that anticipated some recently resurfaced ideas in the field of reservoir computing [30]. His B-Type networks furthermore bear some resemblance to gated recurrent units (GRUs) [31] which have been popularized recently by the LSTM model [32].

> It would be quite possible for the machine to try out variations of behavior and accept or reject them in the manner you describe and I have been hoping to make the machine do this. […] Thus, although the brain may in fact operate by changing its neuron circuits by the growth of axons and dendrites, we could nevertheless make a model, within the ACE, in which this possibility was allowed for, but in which the actual construction of the ACE did not alter, but only the remembered data, describing the mode of behavior applicable at any time. I feel that you would be well advised to take advantage of this principle, and do your experiments on the ACE, instead of building a special machine. I should be very glad to help you over this. [4]

In his later publications and talks, Turing pursued the idea of intelligent and learning machines (or rather software programs?) further, and in his 1950 essay *Computing Machinery and Intelligence* [4] presented the *Imitation Game*, today known as the *Turing Test*, which was meant to illustrate how sufficiently powerful computing machines could be considered to be as intelligent (or more so) than their human counterpart. He was quite outspoken about this conviction:

> The original question, "Can machines think?" I believe to be too meaningless to deserve discussion. Nevertheless, I believe that at the end of the century the use of words and general educated opinion will have altered so much that one will be able to speak of machines thinking without expecting to be contradicted. I believe further that no useful purpose is served by concealing these beliefs.

These ideas set in motion the development of ever more powerful computer architectures, which in turn enabled generations of increasingly complex artificial neural network models and learning methods. This progress continues well into the present era of deep learning, which owes part of its success to the highly parallelized computing architectures that have emerged in recent decades. But it's worth keeping in mind that this transition from serial "von-Neumann" to parallel "non-von-Neumann" computer architectures is less of a revolutionary new idea than it is a return to the roots of computer science and neuromorphic hardware. In fact, John von Neumann himself had both studied models of biological systems and developed artificial computers like the ENIAC [34], and therefore understood the respective strengths and weaknesses of both approaches. But in a time when computers were still excessively large, expensive and memory a limited resource, he concluded in his tragically incomplete lecture notes *The Computer and the Brain* [35], from which I have stolen the title of this chapter:

> That is, large and efficient natural automata are likely to be highly parallel, while large and efficient artificial automata will tend to be less so, and rather to be serial. […] More specifically, not everything serial can be immediately paralleled — certain operations can only be performed after certain others, and not simultaneously with them (i.e. they must use the results of the latter). In such a case, the transition from a serial scheme to a parallel one may be impossible, or it may be possible but only concurrently with a change in the logical approach and organization of the procedure. Conversely, the desire to serialize a parallel procedure may impose new requirements on the automaton. Specifically, it will almost always create new memory requirements, since the results of the operations that are performed first must be stored while the operations that come after these are performed. Hence, the logical approach and structure in natural automata may be expected to differ widely from those in artificial automata.

Half a century later and with new materials and manufacturing processes at hand, neuromorphic hardware might finally be able to bridge this gap between natural and artificial automata.

1.2 From Perceptrons to Deep Neural Networks

The theoretical study of artificial neural networks as (simulated) learning machines continued, first under the label of *cybernetics*, then *connectionism*, into the modern field of *deep learning*. First, Frank Rosenblatt's original Perceptron [1] demonstrated that even a simple feed-forward network model, composed of one layer of (random) feature detectors followed by a single McCulloch-Pitts neuron, could solve many perceptual problems. Minsky and Papert [20] thoroughly analyzed the capabilities and limitations of this and similar kinds of network with their corresponding learning rules mathematically, and provided sound arguments why these networks were still impractical for many relevant problems. Despite the fact that they explicitly limited this critique to perceptrons with a single trainable layer [2], this may have had an adverse impact on the amount of research and funding dedicated to the study of perceptrons at the time — a period that is sometimes referred to, a bit melodramatically, as the *first AI winter*.

[2] Their critique also went well beyond the often mentioned inability of individual threshold-linear functions to solve the XOR problem, and included questions of learning speed, complexity of the required networks and even the information content required for specifying all coefficients. Despite significant advances in the field, all of these questions are still relevant today.

Over the course of a few years, multi-layer perceptrons [36] gradually became more powerful and offered a first flavor of the abstract artificial neural networks (ANNs) still in use today: a hierarchy of (affine) linear combinations of inputs followed by non-linear transformations (in this case a step-function) with coefficients that could all be chosen or learned. Kunihiko Fukushima's *Cognitron* [37] made use of a deep hierarchy of neural network layers to solve a complex computer-vision problem, and could therefore be considered one of the first deep neural networks — although its weights were not optimized through end-to-end supervised learning, but partly derived from expert models, partly trained through a competitive form of unsupervized learning. A later extension, the *neocognitron* [38], even introduced shift-invariant features and could be considered an early form of *convolutional neural network* [39]. A series of proofs, e.g. in [40], finally extended the analysis of the computational power of perceptrons by Minsky and Papert to multi-layered networks and showed that different kinds of feed-forward neural networks are capable of uniformly approximating arbitrary real-valued functions. These proofs of universal function approximation capabilities didn't require particularly deep neural networks — a single hidden layer suffices in principle, so many practitioners questioned whether stacking many layers of neurons into deep neural networks would serve any practical "computational" purpose at all. In a curious repetition of history, Minsky and Papert reaffirmed their skepticism of neural networks in a practically unchanged revision of their influential Perceptron book [20], and to similar effect. They wrote:

> The perceptron has shown itself worthy of study despite (and even because of!) its severe limitations. It has many features to attract attention: its linearity; its intriguing learning theorem; its clear paradigmatic simplicity as a kind of parallel computation. There is no reason to suppose that any of these virtues carry over to the many-layered version. Nevertheless, we consider it to be an important research problem to elucidate (or reject) our intuitive judgment that the extension is sterile. Perhaps some powerful convergence theorem will be discovered, or some profound reason for the failure to produce an interesting "learning theorem" for the multilayered machine will be found.

And in some sense their words became a self-fulfilling prophecy, with many researchers opting for the simpler to train and to use shallow network architectures (which had stiff competition from more sophisticated machine learning methods). The following "second AI winter" spelled the end of this *connectionist* era, even though the 'interesting "learning theorem" ', as Minsky and Papert had asked for, already existed unbeknownst to many in the form of the backpropagation algorithm, which was repeatedly re-discovered over the preceding and the following decades [41].

In parallel to these studies of feed-forward networks, *recurrent neural network* (RNN) models were developed, to endow networks with some form of memory and/or allow them to process temporally varying information. Jeffrey Elman introduced *context units*, i.e. hidden neurons that receive the network's previous outputs as additional inputs, into an otherwise feed-forward network, thus retaining previous activity in a form of "active" *working memory* [42]. John Hopfield took inspiration from Ising models [43], which were being developed in statistical physics to model the dynamics of the spins of electromagnetically coupled atoms, and provided an alternative account of memory, where each "memory" is associated with a stable fixed-point of a recurrently connected network's dynamics. These two novel perspectives on memory, a form of volatile memory realized by the networks momentary state and a persistent memory encoded in the network's connectivity, inextricably linked the concepts of memory and computation. In Elman's words:

> In this account, memory is neither passive nor a separate subsystem. One cannot properly speak of a memory for sequences; that memory is inextricably bound up with the rest of the processing mechanism. [42]

1.3 The "Deep Learning Revolution"

After a phase of relative tranquility, (feed-forward) neural networks entered the spotlight for a third time after several convolutional neural network architectures [44–46] won several computer vision challenges, most famously the network nicknamed AlexNet by Krizhevsky, Sutskever, and Hinton, which severely out-performed the competing machine learning methods and thus proved the impressive capabilities of deep neural networks to a wider audience. The real reason for the breakthrough success of deep learning has since been debated intensely. But besides scientific reasons, which we shall look at in chapter 2, the success of deep learning can be attributed at least in part to the availability of "big data", i.e. large, unstructured datasets, which are ideally suited as training material for (deep) neural networks with their large number of parameters. Another factor is certainly the rapid improvement of computer hardware, graphic cards and dedicated accelerators, and a corresponding surge in optimized software tools for simulating large networks such as TensorFlow [47] and PyTorch [48], which enabled many researchers to develop and test countless variations of network architectures.

But the most compelling explanation, in my opinion, is neither better data, software or hardware, nor better performance of deep networks *per se*. Instead, deep learning owes much of its success to the surprising[3] efficiency of the gradient-based optimization of neural networks. This only works because deep neural networks, despite being complex nonlinear models, can be easily differentiated with respect to all their parameters and optimized using *stochastic gradient descent*, also called *(error-)backpropagation* in Deep Learning [39]. The same optimization tools can also be applied to train recurrent networks in discrete time, by a procedure called *backpropagation through time* [49]. If a task can be expressed by a differentiable *loss* function, as it is often the case in machine learning problems, we can therefore use variations of the greedy (stochastic) gradient descent algorithm to iteratively reduce the loss. This offers a very simple interface towards applications, because it only requires specifying the goal of a task in terms of a differentiable loss function and providing some data — little domain knowledge required! So, deep learning really is all about *learning*, albeit in the narrow context of optimization, rather than biology or psychology.

[3] We will see in chapter 2 why this is surprising.

1.4 The state of the field(s) today

Of course, how neural networks "learn" has been a critical question not just in machine learning, but also for theoretical/computational neuroscience. However, neuroscientific models of learning naturally have to work within the confines set by biologically plausible mechanisms, and are thus primarily concerned with questions about how *unsupervised* (possibly modulated by other factors) local plasticity mechanisms might interact, what kind of top-down error signals may be provided by the nervous system, or how they might be propagated. Machine learning, on the other hand, does not have to play by the same rules, and instead application-driven questions of reliability, speed, performance and efficient usage of limited labelled training data take center stage there.

Today, half a century after the conception of the multi-layer perceptron, deep neural networks are the dominant method throughout many application areas of machine learning, where they have displaced other approaches such as kernel methods and decision trees from the leaderboards of most competitions.

However, just as machine learning has advanced over the last decades, so has neuroscience, and the early models, such as the logic calculus proposed by McCulloch and Pitts, from which *artificial* neural networks were derived, no longer reflect our best current understanding of *biological* neural networks. Since the first full, mechanistic, dynamic model of a biological neuron by Hodgkin and Huxley [50], major technological and methodical improvements in experimental neuroscience have revealed more and more about the complex biological mechanisms at play, and our theoretical models of neurons and networks have changed accordingly. Chapters 4 to 7 are about some of these developments.

To make a long story short, what once started as a single research question — how neurons process information — has since split into three distinct areas of research: the study of how abstract (deep) artificial neural networks can be used to implement intelligent or learning machines, which today are major subfields of *artificial intelligence* and *machine learning*, the study of how biological neurons process information, which we now refer to as *computational neuroscience*, and the study of how similar artificial systems could be realized efficiently in hardware, now called *neuromorphic hardware*.

References for chapter 1:

1. F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958, ISSN: 1939-1471(ELECTRONIC),0033-295X(PRINT). DOI: 10.1037/h0042519 (cit. on pp. vii, 4, 14, 24).
2. W. R. Ashby, *Design for a Brain: The Origin of Adaptive Behaviour (2nd Ed. Rev.).* Chapman & Hall, 1960. DOI: 10.1037/11592-000 (cit. on pp. vii, 2, 45).
4. A. M. Turing and B. J. Copeland, *The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life, plus the Secrets of Enigma.* Clarendon Press ; Oxford University Press, 2004, ISBN: 978-0-19-825079-1 978-0-19-825080-7 (cit. on pp. vii, 3, 31, 32).
20. M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*, Expanded ed. MIT Press, 1988, ISBN: 978-0-262-63111-2 (cit. on pp. 1, 4, 11, 12, 26).
21. J. J. van Middendorp, G. M. Sanchez, and A. L. Burridge, "The Edwin Smith papyrus: A clinical reappraisal of the oldest known document on spinal injuries," *European Spine Journal*, vol. 19, no. 11, pp. 1815–1823, 2010, ISSN: 0940-6719. DOI: 10.1007/s00586-010-1523-6. pmid: 20697750 (cit. on p. 2).
22. M. Piccolino, "Animal electricity and the birth of electrophysiology: The legacy of Luigi Galvani," *Brain Research Bulletin*, vol. 46, no. 5, pp. 381–407, 1998, ISSN: 03619230. DOI: 10.1016/S0361-9230(98)00026-4 (cit. on p. 2).
23. C. GOLGI, "Sur la structure des cellules nerveuses," *Arch. Ital. Biol.*, vol. 30, pp. 60–71, 1898 (cit. on p. 2).
24. C. Darwin, *On the Origin of Species by Means of Natural Selection, or Preservation of Favoured Races in the Struggle for Life.* John Murray, 1859 (cit. on p. 2).
25. T. Schwann, *Mikroskopische Untersuchungen Über Die Uebereinstimmung in Der Struktur Und Dem Wachsthum Der Thiere Und Pflanzen*, 1. Auflage. Sander, 1839 (cit. on p. 2).
26. C. Golgi, "The Neuron Doctrine: Theory and Facts," in *Physiology or Medicine*, vol. 1 (1901-1921), 1906, ISBN: 981-02-3409-0 (cit. on p. 2).
27. W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1, 1943, ISSN: 0007-4985, 1522-9602. DOI: 10.1007/BF02478259 (cit. on pp. 2, 10, 25).
28. "On computable numbers, with an application to the Entscheidungsproblem," *J. of Math*, vol. 58, no. 345-363, p. 5, 1936 (cit. on p. 2).
29. A. M. Turing, "Intelligent machinery," 1948 (cit. on p. 2).
30. H. Jaeger, W. Maass, and J. Principe, "Special issue on echo state networks and liquid state machines.," 2007 (cit. on pp. 2, 14, 62).
31. J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. (2014). "Empirical evaluation of gated recurrent neural networks on sequence modeling." arXiv: 1412.3555 (cit. on p. 2).
32. S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1, 1997, ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735 (cit. on pp. 2, 10).
33. D. O. Hebb, *The Organization of Behavior. A Neuropsychological Theory.* John Wiley & Sons, 1949 (cit. on p. 2).
34. H. H. Goldstine and A. Goldstine, "The electronic numerical integrator and computer (eniac)," *Mathematical Tables and Other Aids to Computation*, vol. 2, no. 15, pp. 97–110, 1946 (cit. on p. 3).
35. J. von Neumann, *The Computer and the Brain.* Yale University Press, 1958, ISBN: 978-0-300-08473-3 978-0-300-00793-0 978-0-300-02415-9 (cit. on pp. 3, 31).
36. A. Gamba, L. Gamberini, G. Palmieri, and R. Sanna, "Further experiments with PAPA," *Il Nuovo Cimento (1955-1965)*, vol. 20, no. 2, pp. 112–115, 1961, ISSN: 1827-6121. DOI: 10.1007/BF02822639 (cit. on p. 4).

37. K. Fukushima, "Cognitron: A self-organizing multilayered neural network," *Biological Cybernetics*, vol. 20, no. 3-4, pp. 121–136, 1, 1975, ISSN: 0340-1200, 1432-0770. DOI: 10.1007/BF00342633 (cit. on p. 4).

38. K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980, ISSN: 0340-1200, 1432-0770. DOI: 10.1007/BF00344251 (cit. on p. 4).

39. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT press Cambridge, 2016, vol. 1 (cit. on pp. 4, 5, 9, 12, 85).

40. G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, vol. 2, no. 4, pp. 303–314, 1989, ISSN: 0932-4194, 1435-568X. DOI: 10.1007/BF02551274 (cit. on pp. 4, 10).

41. J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015. DOI: 10.1016/j.neunet.2014.09.003 (cit. on pp. 4, 9).

42. J. L. Elman, "Finding Structure in Time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990, ISSN: 03640213. DOI: 10.1207/s15516709cog1402_1 (cit. on pp. 5, 11).

43. J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences*, vol. 79, no. 8, pp. 2554–2558, 1, 1982, ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.79.8.2554. pmid: 6953413 (cit. on pp. 5, 14).

44. K. Chellapilla, S. Puri, and P. Simard, "High Performance Convolutional Neural Networks for Document Processing," p. 7, (cit. on p. 5).

45. D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Flexible, High Performance Convolutional Neural Networks for Image Classification," p. 6, (cit. on p. 5).

46. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 24, 2017, ISSN: 0001-0782, 1557-7317. DOI: 10.1145/3065386 (cit. on p. 5).

47. M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: A System for Large-Scale Machine Learning," presented at the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), 2016, ISBN: 978-1-931971-33-1 (cit. on pp. 5, 12).

48. A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, 2019 (cit. on p. 5).

49. P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990 (cit. on p. 5).

50. A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of Physiology*, vol. 117, no. 4, pp. 500–544, 28, 1952, ISSN: 0022-3751. pmid: 12991237 (cit. on pp. 6, 61).

Connectionists use learning rules in big networks of simple components — loosely inspired by nerves in a brain. Connectionists take pride in not understanding how a network solves a problem. [...] If you just have a single problem to solve, then fine, go ahead and use a neural network. But if you want to do science and understand how to choose architectures, or how to go to a new problem, you have to understand what different architectures can and cannot do.

— Marvin Minsky

2 *Information processing in artificial neural networks*

When we talk about information processing by *biological* neurons and networks, the best place to start is probably the highly simplified, biologically inspired model of *artificial* neural networks. But as we already saw in chapter 1, the development of these models for machine learning purposes has since become a science of its own, and has produced models that satisfy the constraints of computer hardware very well, but differ from their biological inspiration in substantial ways. These differences are what I want to explore throughout the rest of this thesis.

2.1 Terminology

Throughout this thesis, I will rely on a lot of terminology and basic concepts from theoretical neuroscience and machine learning. If you are already familiar with these, feel free to skip ahead to section 2.2; otherwise the following section offers a brief and high-level summary. For a more in-depth or rigorous definition, any current review or book concerned with neural networks should do, e.g. Goodfellow, Bengio, and Courville [39], Schmidhuber [41], and Strang [51]. In abstract terms, an *artificial neural network* (ANN) is a graph of *nodes* and *directed, weighted edges*. The nodes represent *neurons*, the edges represent *synaptic connections* between neurons. Each synapse connects its *pre-synaptic neuron* to its *post-synaptic neuron*. In turn, the synapse is one of its pre-synaptic neuron's *outgoing*, and one of its post-synaptic neuron's *incoming* connections. Each edge can be assigned a *synaptic weight* or *efficacy*, which either represents the gain that the synapse applies to its signal or the probability with which an individual *spike* (also called *never impulse* or *action potential*) is transmitted by the synapse. These weights of a network can be collected in a so-called *weight matrix*, also called the *connectome* of the network. In a machine learning context, neurons are typically grouped into an ordered list of $k > 0$ *layers* of neurons with typically no connections within each layer.[1] For the most common kind in machine learning, *feed-forward* networks, the graph is acyclic[2], meaning that its directed synapses define a partial ordering of the neurons from the input all the way "forward" to output neurons (hence the name). If the graph describing the network is cyclic, it is referred to as a *recurrent network*.[3] Biological neural networks can also be stratified into layers, but here the definition is based on the neurons' cell-bodies' locations, rather than their connectivity, and hence cannot be used interchangeably.

Some neurons or layers (for feed-forward networks typically the first layer(s) in the hierarchy) receive external input signals and are called the *input neurons* or *layers*, whereas the neurons or layer(s) whose states are taken as the output of the network (for feed-forward

[1] The corresponding graph is k-partite, and the synaptic weights of such a network form a block-matrix.

[2] The weight matrix of such a network is (block) triangular without diagonal entries.

[3] Sometimes, a bit inconsistently, also groups of neurons with mutual connections that are embedded in an otherwise feed-forward structure are referred to as *recurrent layers* of a feed-forward network. The corresponding synaptic weights would then form a block-triangular matrix *with* super-diagonal entries in the diagonal blocks.

networks typically found higher up in the hierarchy) are called the *output neurons* or *layer(s)*. Neurons or layers that are neither input nor output are called *hidden*. Following a rather vague heuristic, a network is typically called a *deep neural network* if it is composed of particularly many layers and its weights are inferred from data — but exceptions to this rule exist, e.g. the Long-Short-Term-Memory [32] architecture is commonly counted among the deep neural network architectures despite it being a recurrent network. Conversely, networks with only very few hidden layers are sometimes called *shallow*. Since the concept of depth is not applicable at all in cyclic graphs, the distinction between *deep*, *shallow* and other networks, is therefore blurry in practice and often determined by the historical context.

The graph structure defines the *topology* of the network, and together with a choice of *neuron* and *synapse models* determines the *network architecture*. The precise behavior of the network depends on the values of its *parameters* such as *synaptic weights*, *gain* or *bias terms*.

Artificial neural networks are often used to approximate functions that are only partially specified by some *training dataset* of input-output pairs. The act of optimizing the network parameters for this task is then referred to as *training*. Using a trained neural network to map (previously unseen) inputs onto corresponding outputs is referred to as *inference*.

When simulating neural networks, time can be represented in different ways. For feed-forward networks in a Machine Learning context, time is typically quantized into *discrete update steps*, where the information is assumed to propagate through the entire network from the input layer to the output layer within one time-step. For recurrent networks in discrete time, one time-step resembles one simultaneous update to each neuron's output, and the time-varying outputs of the network are represented by *real-valued sequences*. In a computational neuroscience context, especially for analog and spiking networks, time is often represented *continuously*. The signals emitted by the neurons are then modeled as *real-valued functions*, *stochastic processes*, or *spike trains*. While it is in principle possible to mix these different kinds of signals within one network, most architectures assume signals to be either all continuous, discrete, or spiking.

2.2 Artificial neural networks are function approximators

For machine learning applications, the utility of neural networks derives solely from their remarkable ability to approximate highly non-linear functions. For example, if a neural network is used to distinguish images of cats from images of dogs, it is implicitly assumed that the network is *in principle* capable, once the correct parameters are chosen, to solve the problem — at least to a sufficiently good degree of approximation. In other words, we expect the network, a complex mathematical model with lots of parameters, to be able to *approximate some (unknown) function* that maps from the space of photographs to the discrete set {cat, dog}. This may seem trivial now, but proving that this holds for the kind of networks we use today turned out to be a quite challenging task. It has long been known that networks of simple McCulloch-Pitts neurons with a step-function as nonlinearity can represent any *boolean* function [27], but showing that this concept can be generalized to *arbitrary continuous* functions for neurons with other nonlinearities such as the rectified linear or the hyperbolic tangent function is anything but trivial. Cybenko [40] finally provided the very general proofs, that any continuous function on the n-dimensional unit cube or any indicator function of a finite measurable subset on the n-dimensional unit cube can be uniformly approximated by an expression of the form

$$y(x) = W^{out} f(W^{in} x + b),$$

with parameters W^{out}, W^{in} and b if f is continuous and discriminatory [4] (e.g. any sigmoid

[4] A function is defined to be discriminatory if its integral w.r.t. a non-negative measure is zero only if the measure is identically zero. The most popular choice of activation function in deep learning, the rectified linear activation functions of the form $ReLU(x) = \max(x, 0)$ for example is not discriminatory, since it is constant on the negative half-plain.

function would do). Hornik [52] showed that the same in fact holds true for *any* arbitrary function that is continuous, bounded and non-constant. [5] A more recent insight about the approximation capabilities of deep neural networks comes from the study of splines: a feed-forward network with rectified-linear activation functions — a very popular choice in machine learning — provides a parameter-efficient way to construct a continuous, piece-wise linear (CPWL) function or *spline* [53–55].

[5] This proof still does not directly cover the ReLU function, which is not bounded, but since a linear combination of two ReLUs can be used to construct a bounded function that satisfied the requirements, this indirectly proves the same power for ReLU functions as well [51].

In simple terms this means, that both regression problems (i.e. approximating a continuous function) and classification problems (i.e. approximating an indicator function over some subset) can be solved arbitrarily well by a neural network with one or more hidden layers, *if* there are enough neurons in the hidden layers.

The addition of recurrent connections allows these models to retain information for extended periods of time in the network's state, endowing the system with a form of memory. This was observed by Elman [42], who argued that by including context units, which feed back the system's current output as an additional input, the network can "memorize" and distinguish sequences of input. This idea has been generalized by Funahashi and Nakamura [56], proving that in fact any continuous function of time $F(t)$, including of course the state of an autonomous dynamical system, can be uniformly approximated arbitrary well on a (finite) time-interval by some recurrent neural network with sufficiently many hidden neurons. The argument was further extended by Chow and Li [57] to non-autonomous, i.e. input-driven, systems under few generous additional conditions.

The function/system approximation paradigm therefore also covers recurrent artificial neural networks. Similar arguments can be made for both feed-forward and recurrent spiking neural networks (see chapter 6), which are universal computers as well, given sufficiently many neurons [58].

To summarize the summary, these existence proofs show that feed-forward and recurrent artificial neural networks, spiking or not, can *in principle* (i.e. if they are sufficiently large, which depends on the task at hand) approximate both instantaneous functions of the input and input-driven dynamical systems arbitrarily well. This makes artificial neural networks extremely powerful tools for machine learning and, as we shall see, neuromorphic hardware.

2.3 A bird's eye view of artificial neural networks

The existence proofs above only promise that neural networks are in principle capable of approximating functions, but give no indications of *how* such networks can be constructed. A large part of neural network research since the Connectionists' era has therefore been concerned with finding different architectures that are good at solving different tasks, as well as smart, systematic ways to combine them. As Minsky and Papert [20] already wrote:

> Different kinds of networks lend themselves best to different kinds of representations and to different sorts of generalizations. [...] This is why we maintain that the scientific future of connectionism is tied not to the search for some single, universal scheme to solve all problems at once but to the evolution of a many-faceted technology of "brain design" that encompasses good technical theories about the analysis of learning procedures, of useful architectures, and of organizational principles to use when assembling those components into larger systems.

Today, there are at least four major conceptual frameworks that provide tools for constructing useful, large neural networks. We'll have a look at each of them in the following, and see why neither of them is suitable for studying information processing on the lower level of individual biological neurons that I am interested in.

2.3.1 Deep Learning

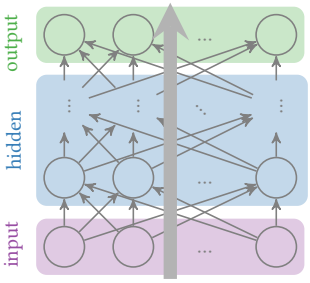


Figure 2.1. A feed-forward network with input, hidden, and output layer(s). Information flows only from input towards output layers.

Currently, the most popular framework by far for constructing large artificial neural networks is *deep learning*. As the name implies, it typically involves stacking many layers of neurons into deep hierarchies, and training the entire network through gradient-based optimization methods.

Despite its apparent success, this approach was (and still is) met with some skepticism: Why would stacking more than one hidden layer have any *qualitative* benefit for the network's computational power, if the proofs above show that a single hidden layer is apparently enough? Also, given their typically rather large number of parameters, the numerics of optimizing these networks alone are a serious challenge that requires capable linear algebra tools [47]. And more fundamentally, according to basic results from statistical learning theory [59], these networks should either require an infeasibly large amount of training data and/or strong regularization of the parameters, [6] or generalize poorly to new data. This is apparent from the large amount of information content that the network weights store, as already pointed out by Minsky and Papert [20][7]. Another concern from the perspective of nonlinear optimization of such large models should be local minimal of the loss function — i.e. mediocre solutions of the problem, which are difficult to improve for greedy gradient based methods and would require more sophisticated optimization methods.

Nevertheless, the training of large deep neural networks with gradient methods appears to stubbornly defy these intuitions, and has been surprisingly effective in practice. One argument in favor of building such deeper hierarchies is, that this "compositionality" allows for a much more efficient approximation of high-dimensional functions by comparatively few neurons. For the example of networks with rectified-linear activation functions, which correspond to piece-wise linear splines as we already saw above, the *number of neurons* required to approximate a given function can in some cases be exponentially *reduced* as we *increase* the *number of layers* in the network![8] Another argument is based on the surprising observation that the higher number of parameters in deep neural networks appears to make the optimization *easier*! Even deep neural networks that have parameters in excess of training data samples, make no explicit use of regularization, and perfectly interpolate all the training data have been shown to generalize well to previously unseen data [62]. This point has been debated a lot, but the answer might come from two rather unexpected directions: First, in the high dimensional parameter-space of neural networks, local minima of the loss function are comparatively *less* common in relation to e.g. saddle points, which pose much less of a problem for gradient methods [63]. Second, it appears that the commonly used *stochastic gradient descent* algorithm itself has an *implicit regularizing* effect on the learned coefficients, and leads the network coefficients towards a minimum norm solution [62, 64, 65]. A different, albeit controversial, explanation of the same phenomenon can be made in terms of the *information bottleneck principle* [66], which we'll return to in a different context in chapter 5. The *lottery ticket hypothesis* [67] goes one step further and suggests, that *much* smaller sub-networks (the *winning tickets*) that perform equally well can typically be extracted from deep neural networks by eliminating most of the synapses, neurons or even layers. Sometimes, the learned weights of entire layers within a deep neural network can be completely irrelevant for the task, as can be demonstrated by randomizing them with only negligible impact on performance [68]. However, *finding* such a sparser or shallower structure *directly*, e.g. by optimizing a smaller neural network to begin with, fails in practice. This, the argument goes, is because the parameter-rich deep neural networks offer a large search space, or a *scaffold* of sorts, in which the much smaller networks (which are sufficient to solve the task) can be constructed efficiently using gradient based optimization methods.

[6] There are a lot of explicit and heuristic options for regularization in deep learning, such as Nesterov optimizers, dropout, weight decay, data augmentation and early stopping [39].

[7] For example, the MegatronLM GPT2 model with its 345 million parameters comes at a compressed size of about 650MB [60], its bigger cousin with 8.3 billion parameters [61] is correspondingly larger.

[8] To be precise: there are functions f for every integer k such that a *deep* network with k^3 neurons distributed over k^2 layers can approximate f at least as well as any *shallower* network with $\leq k$ layers and $\frac{1}{2}k^{k+1} - 1$ or more neurons! [54].

Therefore, gradient descent and deep learning are so intimately coupled, that one of the most prominent figures in the field, Yann LeCun, even advocated for altogether switching to the more accurate name "differentiable programming" instead of "deep learning", since the optimization of neural networks by gradient-based methods is *the* defining feature, rather than the depth of the networks, as the term "deep learning" would imply [69].

An increasingly important practical consideration of *deep learning* research today is also the search of *parameter efficient* network architectures for different domains, i.e. by trying to reduce the number of neurons or synapses [70, 71], efficiently reusing synaptic weights [72] or even lowering the precision of weights [73, 74]. Over time, deep neural network architectures have thus evolved, much like their biological counterparts, and in the process incorporated a lot of domain-specific optimizations and inductive biases that make them well suited to specific tasks, but also pose a risk as they are typically not well understood (see also the note below).

Note: *Nature* or *nurture* in deep learning?

One important question in neuroscience is, how much of behavior is genetically predetermined (i.e. by *nature*), and how much is learned (i.e. by *nurture*). A very similar question could in fact be asked for deep learning! On first sight, it may seem obvious that all network coefficients are learned from data, and the amount of domain expertise explicitly build into the network is also minimal compared to early work by Connectionists, hence learning should play the major part. However, this is not true if we look at the discipline of deep learning as a whole: Deep neural networks are typically evaluated and compared to each other by training and testing them on the same benchmark tasks with the same datasets. Naturally, the more effective solutions are more likely to be picked up and developed further. What is kept and modified from one implementation to the next is typically not the weights (although pre-trained networks exist), but the *network architecture* — i.e. the types and numbers of layers, activation functions, etc. But, if we now evaluate the next generation of networks on the same dataset(s), we have committed a cardinal sin of statistical modeling by peeking at the test dataset (i.e. starting from an architecture, a meta-parameter of the model, selected on the same test dataset) before training the model! We could avoid this by testing each model on entirely new testing data, but even in that case merely choosing the best-performing model as a starting-point effectively bakes domain "knowledge" (or rather, information) into the network architecture. The result of this dynamic, which happens not on the level of the individual researcher but across the entire field, could be viewed as an evolutionary algorithm that produces ever more powerful, complex and specialized network architectures for these benchmarks. This is partially deliberate, since we *want* to find better and more useful models, partially unwittingly, since we might underestimate *how much* of bias this can introduce. A funny and slightly worrying example of this are the *weight agnostic neural networks* by Gaier and Ha [75], which show that some network architectures are *so specifically* designed for a certain task that they can solve it reasonably well even if *all* of their coefficients are fully randomized — reducing the role of *learning* in deep learning *ad absurdum*!

In a nutshell, the deep learning framework uses large, mostly feed-forward artificial neural networks with weights optimized by gradient descent to approximate functions that minimize some differentiable measure of "loss" or "cost". Its surprising effectiveness is at least in part due to the *quantitative* improvements that this optimization approach can offer, such

as the reduction of local minima and implicit regularization, but this can come at the expense of over-complete networks, where many neurons or synapses may be entirely redundant. The choice of network architectures plays an important part, as well, but is often based on heuristics and incremental improvements of prior work. Despite interesting attempts to address the issue [76, 77], this "black-box" character of deep neural networks remains a contentious topic of debate [78] to date.

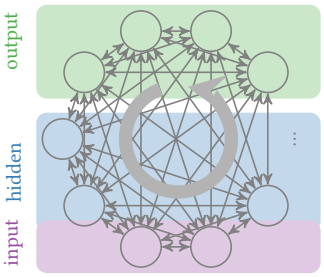


Figure 2.2. A recurrent network with input, hidden, and output neurons. Information is actively maintained in the network by recurrent activity.

2.3.2 Attractor Networks

Attractor networks offer a completely different explanation of information processing by artificial neural networks. As we have seen above, recurrent neural networks can, if large enough, implement arbitrary dynamical systems. The computation realized by a neural network could therefore also be attributed to the long-term dynamics of the network as a whole, such as convergence of the network's state to some fixed-point, rather than the instantaneous output of a network in response to input. This insight underlies the model proposed by Hopfield [43]. He studied the linearized dynamics of recurrently connected neural networks and showed, that under certain constraints on the connectivity between neurons (e.g. symmetric and bounded connection weights), the network state must —without external input— converge to one of possibly many stable equilibria, depending on the network's initial state. Starting from a perturbed state that is similar but not identical to one of these stable attractors brings the network activity towards the attractor, and thereby "restores" the unperturbed stable state. By converging to a fixed-point, the network thus retrieves and reconstructs a perfect "memory" from an incomplete or corrupted version of that memory — a form of memory that he called *content addressable memory*. Rather than a *function*, the network thus implements an iterative *algorithm*! Since each memory is implemented by a fixed-point of the network dynamics, it imposes a constraint on the weight matrix of the network, and the total number of memories that can be stored this ways (and the robustness with which each memory can be recovered) depends on the size of the network. While Hopfield networks themselves are barely used in machine learning, they have been very influential in theoretical neuroscience. Similar ideas can be found today also in *reservoir computing* approaches.

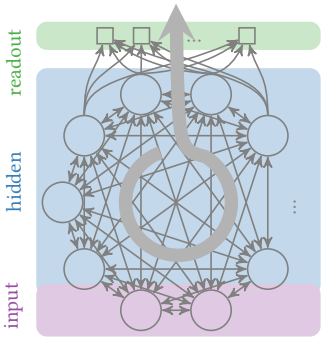


Figure 2.3. A reservoir computer with input and hidden neurons and a linear readout layer. Information is actively maintained in the network by recurrent activity and accessed through the readout layer.

2.3.3 Reservoir Computing and Conceptors

Reservoir computing (also known as *echo state networks* or *liquid state machines*) [30] represents a more radical approach to using recurrent network dynamics. It uses the transient dynamics of *randomly* connected recurrent neural networks, which are continuously driven by the network's inputs. The time-varying state of the networks' neurons thus provides a random, non-linear embedding of the network's input history into a high-dimensional vector space. For this embedding to be useful, it only needs to satisfy a few basic and easy to satisfy conditions [58]. The recurrent network is then called a *reservoir computer*, and a linear transformation of the network's high-dimensional state vector can be used to approximate a time-varying target signal, i.e. some function of the network's recent input history. This transformation is also called the network's linear *readout*, and its coefficients are the only parameters of the reservoir computer that are optimized for a specific task. [9] The *conceptor* framework [80] combines this approach with ideas from attractor networks. By controlling the attractors and limit-cycle dynamics of the reservoir (like a Hopfield network), a conceptor network can shape the dynamics to produce or resonate with certain stable time-varying patterns of activity, which a linear readout (like in a reservoir computer) can shape into a desired time-varying output signal. For an example of a network architecture built from such conceptor networks, see contribution 1.

[9] This combination of a random high-dimensional, non-linear feature-expansion with a simple linear regression model is reminiscent of the *kernel trick* popularized by support vector machines [79] and the original Perceptron [1].

Contribution 1: Bistable Perception in Conceptor Networks

This conference paper explores, how a hierarchy of conceptor networks, which can act as generative models for time-series signals, can be used to actively suppress noise and minimize prediction errors. The idea of such a hierarchical predictive coding scheme is in line with biological observations and provides an appealing model of perception. When presented with ambiguous superposition of two stimuli, this architecture reproduces the well known psychological phenomenon of bi-stable perception, where either of the two pure stimuli is perceived in isolation for a period of time, before the percept switches to the other. It matches empirical results with surprising fidelity, including the distribution of the time-spans for which either of the stimuli is perceived! This paper extends ideas developed within Felix Meyer zu Driehausen's thesis, which I had the pleasure of supervising. Felix and Rüdiger Busche subsequently turned it into a viable model and a nice conference paper, for which they deserve all the credit.

Reference (see also appendix C, page 100ff for the full text):

F. Meyer zu Driehausen, R. Busche, **J. Leugering**, and G. Pipa, "Bistable Perception in Conceptor Networks," in *Artificial Neural Networks and Machine Learning – ICANN 2019: Workshop and Special Sessions*, 2019, ISBN: 978-3-030-30493-5. DOI: 10.1007/978-3-030-30493-5_3.

2.3.4 *The Neural Engineering Framework*

The *Neural Engineering Framework* (NEF) incorporates ideas from both deep learning and reservoir computing, but it offers a rather different perspective on how to construct large neural networks. A great in-depth discussion of this approach can be found in [5]. It focuses on the design of modular neural network architectures from smaller building blocks with well-defined function, rather than end-to-end optimization of network coefficients from data. Each of these building blocks is itself a randomly connected feed-forward network or reservoir computer, whose activation encodes (or *represents*) some variable, typically low-dimensional. Connections between the blocks are optimized to implement functions (or *transformations*) of these variables. By stacking and connecting many such modules together, large and complex networks with well-defined behavior can thus be constructed. This approach is very general and works well for different kinds of neuron models, from the simple linear-nonlinear neurons to more complex dynamic neuron models like leaky integrate-and-fire neurons, because it doesn't rely on the backpropagation of gradient information through deeply nested hierarchies of neural network layers. Figure 2.4 illustrates this modular approach.

A practical limitation of this approach is that it requires the problem to be analytically decomposed into sub-problems that are each simple enough to be efficiently solved by a single module, and we need either an analytical solution of each of these sub-problems or sufficient data to train them. But when this is possible, the neural engineering framework provides a straight-forward way to compose smaller feed-forward and recurrent neural networks into very large networks with well-understood, highly complex functionality.[10]

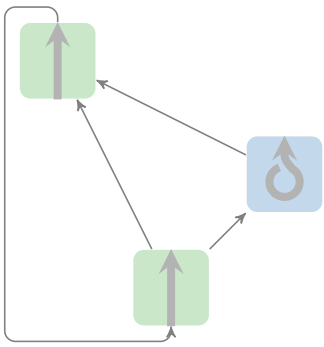


Figure 2.4. A network constructed out of mutually connected feed-forward and reservoir modules according to the neural engineering framework.

[10] The Nengo software tool [81] offers a very fast and entertaining way to construct neural networks using the NEF.

2.4 Where artificial and biological neural networks diverge

We now looked at four popular frameworks for constructing large artificial neural networks, each of which offers a different interpretation of *'neural information processing'*. But do these machine learning frameworks explain information processing in the brain? I believe not, because despite their shared history, terminology and many conceptual connections, the artificial neurons and networks shown above are extreme simplifications, idealizations and modifications of their biological inspirations. To name just of a few of the fundamental differences that have emerged between the artificial neural networks models form machine learning and those from theoretical neuroscience:

1. Artificial neural network models employ a single kind of neuron and a single kind of synapse, the behavior of which is fully parameterized by a single bias and weight coefficient each. In the brain, however, neurons are morphologically and behaviorally so diverse that even a classification into distinct classes can be challenging. But with around 60 different types of neurons in the retina alone, the number of different classes of neurons in the human brain may well be in the hundreds [82] — not to even speak of other cell types like glia cells, which are entirely absent from artificial neural network models. Similarly for synapses, whose inhibitory and excitatory effects are not adequately expressed by positive and negative synaptic weights alone [83]. Also, electrical gap junctions, which directly and bi-directionally couple neurons and play an important role in some biological neural networks [84], are not modeled at all in artificial neural networks. In addition to these neural structures, the vast variety of neurotransmitters or -modulators is ignored in artificial neural networks, despite their critical influence on behavior in biology [85].
2. A lot of biological structures are genetically pre-determined, rather than learned in an end-to-end fashion as is popular in machine learning. For instance, in simple multicellular organisms such as *C. elegans* [86] or tadpole larvae [87], neurons are assembled in very specific, genetically determined patterns; so specific in fact, that the entire connectomes can be described on the level of individual neurons. Neurons can also form specific *motives* [88] of mutual connectivity patterns, that are reproduced many times throughout the nervous system, or larger homogenous groups of strongly coupled neurons called *cell assemblies* [89], or even larger structures called *canonical microcircuits*, possibly arranged in a columnar structure called *cortical (micro-)columns* — but this latter point remains a contentious topic [90]. These innate, highly specific structures don't fit any of our machine learning paradigms above, yet they "compute" nonetheless.
   This is particularly relevant if one considers the fact that neurons and networks must have *evolved* from such and even simpler structures. Studies of the evolutionary origins of neurons and nervous systems point to simple sensory cells arranged in homogenous *nerve nets* [91] that made use of available electrical or chemical messaging processes [92] to broadcast sensory stimuli or motor commands across the animal's body. None of these mechanisms seem to fit the frameworks discussed above — or even the function approximation paradigm in general!
3. Conversely, many biological neural networks also rely heavily on local synaptic, intrinsic and structural plasticity, and thus continuously adjust to changing circumstances (see chapter 5). The weights, biases and nonlinearities of deep networks, on the other hand, are typically assumed to remain fixed after initial training.
4. The transmission of signals also incurs delays, which are ignored in most artificial network models, yet they have significant and unavoidable implications on the behavior, in particular for real-time or recurrently connected neural networks (see chapter 4).

5. The neuron and synapse models in artificial neural networks are also typically modeled as *memory-less* functions, that instantaneously map an input onto an output. Biological neurons, on the other hand, integrate information in time (see chapter 4), adapt (see chapter 5) and can have a rather complex internal state and memory (see chapter 7).
6. For the majority of neurons in the brain, the output is not communicated as a real-valued signal, but via distinct spike events, which requires a fundamentally different mathematical framework (see sec. 6 and chapter 7) of event-based detection of spike patterns, rather than rate-based function approximation.
7. The complex morphology of biological neurons and their dendritic arbors is typically ignored in favor of point-neuron models — despite ample evidence of the relevance of that morphology for behavior (see chapter 7).

Given the remarkable complexity of biological neurons, it seems like a fool's errand to attempt to give a single model of neural computation on the same level of abstraction as current artificial neural networks and to then expect that it applies to "the brain" in general — let alone across different species. In the following chapters, I will therefore only attempt to describe a few of these points in some more detail. For a deep dive into the intricacies of many of these neural mechanisms, I'd refer to Singer, Sejnowski, and Rakic [93], which also contains contribution 2, or Laughlin [6] for a more "bottom-up" perspective.

Contribution 2: Computational Elements of Circuits

The book *"The Neocortex"*, published by the Ernst Strüngmann Forum, compiles the current state of knowledge about the basic principles of operation of the neocortex. In our contribution to this work, the book chapter entitled *"Computational Elements of Circuits"*, we discuss several fundamental properties of neural computation — from homeostasis to delayed interactions, synchronization, random feature expansion and reservoir computing. Within this book chapter, my own largest contribution can be found in the section *"Information Processing in Single Neurons and Populations"*, which elaborates and generalizes ideas from contribution 6.

Reference (see also appendix C, page 111ff for the full text):

J. Leugering, P. Nieters, and G. Pipa, "Computational Elements of Circuits," in *The Neocortex*, W. Singer, T. J. Sejnowski, and P. Rakic, eds., red. by J. Lupp, vol. 27, The MIT Press, 2019, pp. 195–209, ISBN: 978-0-262-04324-3.

References for chapter 2:

1. F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958, ISSN: 1939-1471(ELECTRONIC),0033-295X(PRINT). DOI: 10.1037/h0042519 (cit. on pp. vii, 4, 14, 24).

5. C. Eliasmith and C. H. Anderson, *Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems*. MIT press, 2004 (cit. on pp. vii, 15, 60, 64).

6. S. (O. N. Laughlin University Of C, *Principles of Neural Design*. 2017, ISBN: 978-0-262-53468-0 (cit. on pp. vii, 17, 34, 51, 60, 66).

14. F. Meyer zu Driehausen, R. Busche, **J. Leugering**, and G. Pipa, "Bistable Perception in Conceptor Networks," in *Artificial Neural Networks and Machine Learning – ICANN 2019: Workshop and Special Sessions*, 2019, ISBN: 978-3-030-30493-5. DOI: 10.1007/978-3-030-30493-5_3 (cit. on pp. viii, 15).

16. **J. Leugering**, P. Nieters, and G. Pipa, "Computational Elements of Circuits," in *The Neocortex*, W. Singer, T. J. Sejnowski, and P. Rakic, eds., red. by J. Lupp, vol. 27, The MIT Press, 2019, pp. 195–209, ISBN: 978-0-262-04324-3 (cit. on pp. viii, 17).

20. M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*, Expanded ed. MIT Press, 1988, ISBN: 978-0-262-63111-2 (cit. on pp. 1, 4, 11, 12, 26).

27. W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1, 1943, ISSN: 0007-4985, 1522-9602. DOI: 10.1007/BF02478259 (cit. on pp. 2, 10, 25).

30. H. Jaeger, W. Maass, and J. Principe, "Special issue on echo state networks and liquid state machines.," 2007 (cit. on pp. 2, 14, 62).

32. S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1, 1997, ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735 (cit. on pp. 2, 10).

39. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT press Cambridge, 2016, vol. 1 (cit. on pp. 4, 5, 9, 12, 85).

40. G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, vol. 2, no. 4, pp. 303–314, 1989, ISSN: 0932-4194, 1435-568X. DOI: 10.1007/BF02551274 (cit. on pp. 4, 10).

41. J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015. DOI: 10.1016/j.neunet.2014.09.003 (cit. on pp. 4, 9).

42. J. L. Elman, "Finding Structure in Time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990, ISSN: 03640213. DOI: 10.1207/s15516709cog1402_1 (cit. on pp. 5, 11).

43. J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences*, vol. 79, no. 8, pp. 2554–2558, 1, 1982, ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.79.8.2554. pmid: 6953413 (cit. on pp. 5, 14).

47. M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: A System for Large-Scale Machine Learning," presented at the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), 2016, ISBN: 978-1-931971-33-1 (cit. on pp. 5, 12).

51. G. Strang, *Linear Algebra and Learning from Data*. Wellesley-Cambridge Press, 2019 (cit. on pp. 9, 11).

52. K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1, 1991, ISSN: 0893-6080. DOI: 10.1016/0893-6080(91)90009-T (cit. on p. 11).

53. T. Poggio, L. Rosasco, A. Shashua, N. Cohen, and F. Anselmi, "Notes on Hierarchical Splines, DCLNs and i-theory," Center for Brains, Minds and Machines (CBMM), Technical Report, 29, 2015 (cit. on p. 11).

54. R. Arora, A. Basu, P. Mianjy, and A. Mukherjee. (27, 2018). "Understanding Deep Neural Networks with Rectified Linear Units." arXiv: 1611.01491 [cond-mat, stat], [Online]. Available: http://arxiv.org/abs/1611.01491 (visited on 10/17/2020) (cit. on pp. 11, 12).

55. M. Unser, "A Representer Theorem for Deep Neural Networks," *Journal of Machine Learning Research*, vol. 20, no. 110, pp. 1–30, 2019, ISSN: 1533-7928 (cit. on p. 11).

56. K.-i. Funahashi and Y. Nakamura, "Approximation of dynamical systems by continuous time recurrent neural networks," *Neural Networks*, vol. 6, no. 6, pp. 801–806, 1, 1993, ISSN: 0893-6080. DOI: 10.1016/S0893-6080(05)80125-X (cit. on p. 11).

57. T. Chow and X.-D. Li, "Modeling of continuous time dynamical systems with input by recurrent neural networks," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 47, no. 4, pp. 575–578, 2000, ISSN: 1558-1268. DOI: 10.1109/81.841860 (cit. on p. 11).

58. W. Maass and H. Markram, "On the computational power of circuits of spiking neurons," *Journal of Computer and System Sciences*, vol. 69, no. 4, pp. 593–616, 1, 2004, ISSN: 0022-0000. DOI: 10.1016/j.jcss.2004.04.001 (cit. on pp. 11, 14).

59. V. N. Vapnik, "An overview of statistical learning theory," *IEEE transactions on neural networks*, vol. 10, no. 5, pp. 988–999, 1999 (cit. on p. 12).

60. (). "Megatron GPT2 345M | NVIDIA NGC," [Online]. Available: https://ngc.nvidia.com/catalog/models/nvidia:megatron_lm_345m (visited on 08/28/2020) (cit. on p. 12).

61. M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro. (13, 2020). "Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism." arXiv: 1909.08053 [cs], [Online]. Available: http://arxiv.org/abs/1909.08053 (visited on 08/28/2020) (cit. on p. 12).

62. C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. (2016). "Understanding deep learning requires rethinking generalization." arXiv: 1611.03530 (cit. on p. 12).

63. Y. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio. (10, 2014). "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization." arXiv: 1406.2572 [cs, math, stat], [Online]. Available: http://arxiv.org/abs/1406.2572 (visited on 08/22/2020) (cit. on p. 12).

64. B. Neyshabur, R. Tomioka, and N. Srebro. (2014). "In search of the real inductive bias: On the role of implicit regularization in deep learning." arXiv: 1412.6614 (cit. on p. 12).

65. M. Belkin, D. Hsu, S. Ma, and S. Mandal, "Reconciling modern machine-learning practice and the classical bias–variance trade-off," *Proceedings of the National Academy of Sciences*, vol. 116, no. 32, pp. 15 849–15 854, 2019 (cit. on p. 12).

66. N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in *2015 IEEE Information Theory Workshop (ITW)*, 2015. DOI: 10.1109/ITW.2015.7133169 (cit. on pp. 12, 46).

67. J. Frankle and M. Carbin. (2018). "The lottery ticket hypothesis: Finding sparse, trainable neural networks." arXiv: 1803.03635 (cit. on p. 12).

68. C. Zhang, S. Bengio, and Y. Singer. (24, 2019). "Are All Layers Created Equal?" arXiv: 1902.01996 [cs, stat], [Online]. Available: http://arxiv.org/abs/1902.01996 (visited on 08/22/2020) (cit. on p. 12).

69. (5, 2018). "Yann LeCun," [Online]. Available: https://www.facebook.com/yann.lecun/posts/10155003011462143 (visited on 08/20/2020) (cit. on p. 13).

70. Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in Neural Information Processing Systems*, 1990 (cit. on p. 13).

71. E. Karnin, "A simple procedure for pruning back-propagation trained neural networks," *IEEE Transactions on Neural Networks*, vol. 1, no. 2, pp. 239–242, 1990, ISSN: 1941-0093. DOI: 10.1109/72.80236 (cit. on p. 13).

72. B. Lautrup, L. K. Hansen, I. Law, N. Mørch, C. Svarer, and S. C. Strother, "Massive weight sharing: A cure for extremely ill-posed problems," in *Workshop on Supercomputing in Brain Research: From Tomography to Neural Networks*, 1994 (cit. on p. 13).

73. S. Han, H. Mao, and W. J. Dally. (2015). "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." arXiv: 1510.00149 (cit. on pp. 13, 65).

74. M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*, 2016 (cit. on pp. 13, 26).

75. A. Gaier and D. Ha, "Weight Agnostic Neural Networks," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d\ textquotesingle Alché-Buc, E. Fox, and R. Garnett, eds., Curran Associates, Inc., 2019, pp. 5364–5378 (cit. on p. 13).

76. M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why should i trust you? ": Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, 2016, ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939778 (cit. on p. 14).

77. A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski, "Plug & play generative networks: Conditional iterative generation of images in latent space," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.374 (cit. on p. 14).

78. G. Marcus. (2018). "Deep learning: A critical appraisal." arXiv: 1801.00631 [cs, stat], [Online]. Available: http://arxiv.org/abs/1801.00631 (visited on 09/21/2019) (cit. on p. 14).

79. A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, 1, 2004, ISSN: 1573-1375. DOI: 10.1023/B:STCO.0000035301.49549.88 (cit. on p. 14).

80. H. Jaeger. (22, 2017). "Controlling Recurrent Neural Networks by Conceptors." arXiv: 1403.3369 [cs], [Online]. Available: http://arxiv.org/abs/1403.3369 (visited on 08/22/2020) (cit. on p. 14).

81. T. Bekolay, J. Bergstra, E. Hunsberger, T. DeWolf, T. C. Stewart, D. Rasmussen, X. Choo, A. Voelker, and C. Eliasmith, "Nengo: A Python tool for building large-scale functional brain models," *Frontiers in Neuroinformatics*, vol. 7, 2014, ISSN: 1662-5196. DOI: 10.3389/fninf.2013.00048 (cit. on p. 15).

82. R. H. Masland, "Neuronal cell types," *Current Biology*, vol. 14, no. 13, R497–R500, 2004, ISSN: 09609822. DOI: 10.1016/j.cub.2004.06.035 (cit. on p. 16).

83. S. J. Mitchell and R. A. Silver, "Shunting Inhibition Modulates Neuronal Gain during Synaptic Excitation," *Neuron*, vol. 38, no. 3, pp. 433–445, 8, 2003, ISSN: 0896-6273. DOI: 10.1016/S0896-6273(03)00200-9 (cit. on p. 16).

84. D. H. Hall, "Gap junctions in C. elegans: Their roles in behavior and development," *Developmental Neurobiology*, vol. 77, no. 5, pp. 587–596, 2017, ISSN: 1932-846X. DOI: 10.1002/dneu.22408 (cit. on p. 16).

85. C. I. Bargmann, "Beyond the connectome: How neuromodulators shape neural circuits," *BioEssays*, vol. 34, no. 6, pp. 458–465, 2012, ISSN: 1521-1878. DOI: 10.1002/bies.201100185 (cit. on p. 16).

86. J. G. White, E. Southgate, J. N. Thomson, and S. Brenner, "The Structure of the Nervous System of the Nematode Caenorhabditis elegans," *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, vol. 314, no. 1165, pp. 1–340, 1986, ISSN: 0080-4622. JSTOR: 2990196 (cit. on p. 16).

87. K. Ryan, Z. Lu, and I. A. Meinertzhagen, "The CNS connectome of a tadpole larva of Ciona intestinalis (L.) highlights sidedness in the brain of a chordate sibling," *eLife*, vol. 5, E. Marder, ed., e16962, 6, 2016, ISSN: 2050-084X. DOI: 10.7554/eLife.16962 (cit. on p. 16).

88. R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network Motifs: Simple Building Blocks of Complex Networks," *Science*, vol. 298, no. 5594, pp. 824–827, 25, 2002, ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.298.5594.824. pmid: 12399590 (cit. on p. 16).

89. K. D. Harris, "Neural signatures of cell assembly organization," *Nature Reviews Neuroscience*, vol. 6, no. 5, pp. 399–407, 5 2005, ISSN: 1471-0048. DOI: 10.1038/nrn1669 (cit. on p. 16).

90. N. M. M. Amorim Da Costa and K. Martin, "Whose cortical column would that be?" *Frontiers in Neuroanatomy*, vol. 4, 2010, ISSN: 1662-5129. DOI: 10.3389/fnana.2010.00016 (cit. on p. 16).

91. T. Katsuki and R. J. Greenspan, "Jellyfish nervous systems," *Current Biology*, vol. 23, no. 14, R592–R594, 2013, ISSN: 09609822. DOI: 10.1016/j.cub.2013.03.057 (cit. on pp. 16, 74).

92. W. B. Kristan, "Early evolution of neurons," *Current Biology*, vol. 26, no. 20, R949–R954, 24, 2016, ISSN: 0960-9822. DOI: 10.1016/j.cub.2016.05.030 (cit. on p. 16).

93. W. Singer, T. J. Sejnowski, and P. Rakic, eds., *The Neocortex*, red. by J. Lupp. The MIT Press, 2019, vol. 27, ISBN: 978-0-262-04324-3 (cit. on p. 17).

What I cannot create, I do not understand.

— Richard Feynman

In particular, this requirement [of a physical implementation] will help to prevent the solution from being a mere verbalistic 'explanation', for in the background will be the demand that we build a machine to do these things.

— W. Ross Ashby, *Design for a Brain*

3 *Neuromorphic computing — a bridge between engineering and neuroscience*

As we have seen in chapters 1 and 2, machine learning can produce very large artificial neural networks that require correspondingly large data sets and a lot of power to train. Since the available compute resources are a major bottleneck for the deployment of deep neural networks in many real-world applications, the numerical efficiency of artificial neural networks has become a major concern of deep learning research. This optimization of neural network models for the available hardware has certainly produced impressive results, but it also limits the scope of research to just those kinds of models that *can* be efficiently simulated on current hardware. Due to this compromise, modern deep learning on the one hand makes use of tools and algorithms that are not available to biological neurons, and on the other hand it cannot use many of the interesting biological mechanisms studied in theoretical neuroscience.

The field of *neuromorphic hardware* approaches this issue from the other side: If a neural mechanism seems promising for improving computation, but it lacks efficient hardware-support, then we should develop custom hardware rather than compromise our models! The design constraints of neuromorphic hardware are therefore determined only by what can be efficiently realized by analog and digital (or even ionic and photonic) circuits. A large range of biological mechanism that are difficult to integrate into a classical machine learning setting, such as spiking neural networks, are thus commonly used in the neuromorphic computing community. At the same time, the design constraints imposed by the hardware development raise other important questions that can help to challenge and improve theoretical models. As we shall see, many of the constraints faced by engineers are in fact quite similar to the limitations that biological neurons have to overcome. Hence, I believe that neuromorphic hardware can build a bridge between the two disciplines by providing engineers with biological inspirations, and neuroscientists with tools and measures to evaluate their models.

3.1 The neuromorphic zoo

In the wake of the impressive success of deep learning and the foreseeable end of Moore's Law, the research of new, alternative computing architectures and technologies on which to efficiently execute these neural network models attracts considerable interest from both academia and industry [95]. This has led to a revival of research on *neuromorphic hardware*, which promises to convert the theoretical insights from neuroscience into tangible benefits

Contribution 3: A Visit to the Neuromorphic Zoo

In this paper, which accompanies a public talk held at the Embedded World conference, I provide a brief overview over current concepts and academic as well as commercial developments in the field of AI-hardware acceleration in general, and neuromorphic hardware in particular. This paper was selected by WEKA Fachmedien for a re-publication in the magazine DESIGN&ELEKTRONIK, where it appeared in German translation under the title *"Neuromorphe Hardware"*. Since the Embedded World conference accompanies an industry fair, the proceedings are targeted towards an engineering audience and are intended to offer an accessible high-level perspective. (This papers passed an editorial process, but no scientific peer-review.)

Reference (see also appendix C, page 128ff for the full text):

J. Leugering, "A visit to the neuromorphic zoo," in *Embedded World Conference 2020 – Proceedings*, 2020, ISBN: 978-3-645-50186-6.

A German translation of this article appeared also in:

J. Leugering, "Neuromorphe Hardware," *DESIGN&ELEKTRONIK*, no. 7/2020, pp. 41–47, 2020.

for the development of biologically inspired, highly efficient computing hardware. While this idea is not entirely new[1], there are several good reasons for the renewed interest:

The first is *economical*: The stunning success of neural networks in recent years has revealed many new potential application areas for neural networks, from sensor and image processing and voice control all the way to autonomous robots and vehicles. Since many of these applications are "at the edge" [96], i.e. they do not have direct access to high performance computing infrastructure, they require on-board hardware capable of executing specific neural network architectures. Neuromorphic hardware can address this new and growing market.

Second, *technological* breakthroughs in the development of new materials such as various memristive devices [97], carbon nanotubes, in-silicon photonics, spintronics and much more [98], as well as improved procedures of lithography provide new freedom to implement neuromorphic architectures efficiently in hardware.

Third, there are promising new *theoretical* concepts in the field of neural networks as well as in electronics design. For example, an increasing emphasis is placed on non-volatile memory that can persist even when power is switched off. This is often paired with (analog) *in-memory* computing [99], which brings simple processing elements such as logic gates directly together with storage elements. That is great news for neuromorphic hardware, which can leverage this for an efficient implementation of synaptic connections [100, 101]. Similarly, a lot of theoretical models like spiking neural networks, that play only a minor role in conventional machine learning, are being actively explored in the neuromorphic computing domain [102].

All of these factors combined explain the current resurgence of neuromorphic hardware as one leg of the so-called *next generation computing* (NGC, the other leg being quantum computing). Recently, a large variety of hardware implementations, using analog, digital, mixed signal and even photonic circuits have been developed, and the research of neuro-

[1] In fact, neuromorphic hardware is as old as neural networks, since they both pre-date the emergence of powerful general purpose computers. An early example is the Perceptron Mark 1, a physical implementation of the perceptron model by Rosenblatt [1].

morphic computing has since developed into an independent discipline in academia and industry alike. For a brief review of the current state of the neuromorphic hardware field, see contribution 3.

3.2 A signal processing view of neuron models

In chapter 2 we saw how machine learning frameworks build large networks from individual neurons. Each neuron in that context is really just a function of the form $x_j(t+1) = f_j(\sum_i w_{j,i} x_i(t) + b_j)$, which becomes a seamless part of the larger function that describes the network as a whole. It makes little sense to ask, how much energy this neuron consumes, how much memory it has, or what it's latency is. But biological neurons have to have a physical realization of some sort, and this kind of question becomes critical. I therefore think the analogy between neurons and *electronic components* can be much more illustrative and satisfying than the mathematical abstraction of neural networks as function approximators if we want to better understand the behavior of real neurons. In the following, we will look at a few common neuron models, and we will explore how each of them resembles a well-known electronic component with a specific application in computer science or signal processing. These analogies will allow us to transfer some intuitions from engineering disciplines to neuroscience, and will thus help us better understand the computational capabilities and limitations of various neuron models. Figure 3.1 shows an overview of neuron models and closely related electric circuits.

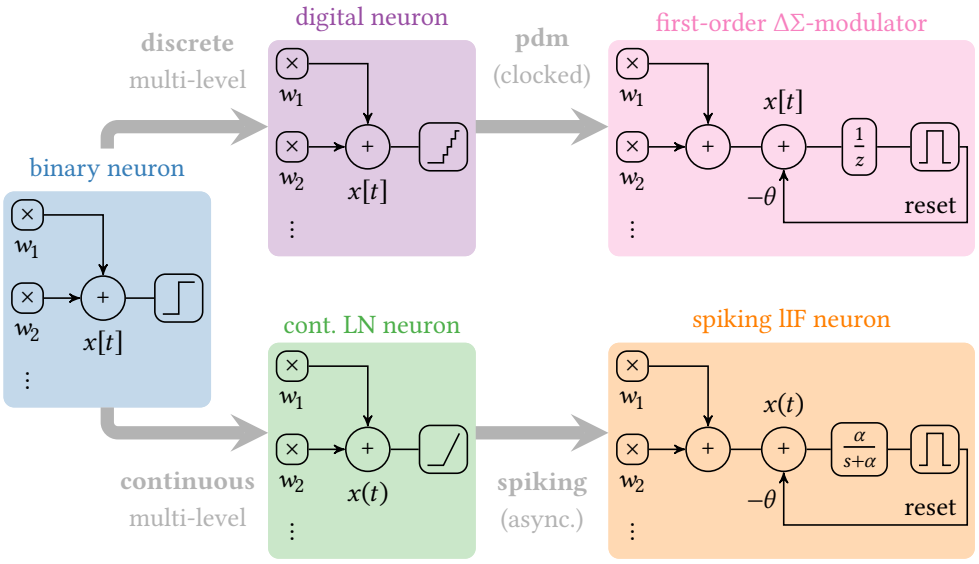


Figure 3.1. *Neuron models, electronics components and their relationships.* Boolean logic gates are equivalent to binary neurons (*left*). They can be extended by allowing multi-valued outputs, resulting in the standard linear-nonlinear neuron model in either digital (*top center*) or continuous (*bottom center*) form. Transmitting these multi-valued outputs by a pulse-density modulated code yields the first-order ΔΣ modulator (*top right*) for discrete time models or the leaky integrate-and-fire spiking neuron model (*bottom right*) for continuous time.

3.2.1 Binary neurons are logic gates

To get started, let's consider the most basic neuron model and logic calculus that McCulloch and Pitts [27] proposed. It states that the neuron's binary output $y[t]$ within some brief time-interval t is `1` (the neuron emits a spike) if its membrane potential exceeds a critical threshold and `0` otherwise. Formally, $y[t] = f(\sum_{i=1}^{N} w_i s_i[t]) - b$, where $f(x) = \mathbb{1}_{[0,\infty)}(x)$ is the Heaviside step-function, w_i are the synaptic weights, b is a bias term and $s_i[t]$ are the values of the input signals at time-interval t. See figure 3.2 for an illustration. The neuron thus maps its N binary input signals onto a single binary output signal, and hence implements a Boolean function. If we limit ourselves to ternary weights as McCulloch and Pitts did, i.e. $w_i \in \{-1, 0, 1\}$, each input is either inverted (an inhibitory input), absent, or left unchanged

(an excitatory input). We can then view the neuron as a *logic gate* that calculates whether *at least b* of its N weighted inputs are equal to 1. This "gate" includes as special cases the `AND` gate ($b = N$) and the `OR` gate ($b = 1$), negations thereof, the constant `true` gate ($b = 0$) as well as the constant `false` gate ($b > N$); it is therefore a *functionally complete set* of first-order logic. [2]

Figure 3.2 shows a schematic of this model.

By assigning each input with an appropriate weight, we can thus construct arbitrary logic circuits, or *binary neural networks with ternary synaptic weights* [103]. From the perspective of deep learning, such *extremely quantized* networks may seem tedious, since they are difficult to train using gradient based-methods. In fact, logic circuits are traditionally optimized using numerically slower discrete optimization methods such as the Quine-McCluskey algorithm [104]. But recent advances in deep compression techniques are leveraging the powerful gradient-based optimization methods from deep learning for the optimization of ternary neural networks (or *XNOR networks*) and have produced highly promising results [74].

[2] The *exclusive* `OR` (`XOR`) gate, however, was famously shown by Minsky and Papert [20] to be not representable by only a single layer of such neurons.

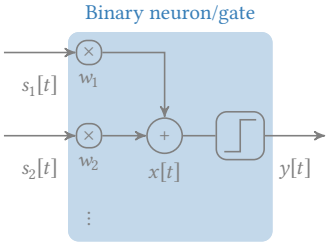


Figure 3.2. A binary neuron is a special boolean logic gate.

3.2.2 *Linear-nonlinear neurons are summing amplifiers*

A natural extension of these binary neurons to real-valued signals leads to the typical (discrete-time continuously-valued) *linear-nonlinear* neuron model that is typically used in deep learning. Here, the hard threshold is simply replaced with a continuous, typically monotonic non-linear function, e.g. a sigmoid function like $f(x) = \tanh(x)$ or the rectified linear function $f(x) = \max(0, x)$. Figure 3.3 shows a schematic. This reflects the observation, that the *firing rate* of a biological neuron increases as a continuous function[3] of its membrane potential (see e.g. chapter 6). We already saw in chapter 2 that a network of such continuous *linear-nonlinear neurons* can be used to approximate arbitrary continuous functions, rather than just Boolean functions. But more importantly, the nonlinear function f that replaces the step-function of the McCulloch-Pitts neuron can be chosen to be (piece-wise) differentiable, which makes the entire network differentiable with respect to its parameters! Deep learning makes use of this fact and relies on gradient-based optimization methods to fix the synaptic weights.

Of course, a similar device is also useful for countless signal processing applications, in particular if we choose the rectified-linear function f as the nonlinearity. The behavior of this neuron could then, in engineering terms, be described as an ideal *summing unity-gain amplifier* or *buffer* that clips off negative values — a common component e.g. in analog audio circuits. See figure 3.3 for a schematic.

This linear-nonlinear model can be realized in hardware either by a digital or a fully analog electronic circuit, and both options are used in practice. Computing directly with analog voltages and currents is an extremely appealing concept and the cornerstone of many *neuromorphic hardware* designs, because it can result in very high energy efficiency and low latency. But analog computation comes with its own drawbacks: for one, analog signals in continuous time are difficult to buffer or route and hence require dedicated physical connections between the neurons, the number of which grows quadratically with the number of neurons. This arrangement may work well in the three-dimensional brain, [4] but poses a serious challenge for neuromorphic hardware that must be laid out in two dimensions. In addition to that, transmitting analog signals over large distances makes them susceptible to noise. A digital implementation of the same model (see figure 3.4) can alleviate these problems, but possibly at the cost of reduced accuracy due to quantization, increased circuit size and complexity, and increased power consumption. This discrete-time, discrete-value implementation is used in many digital neuromorphic circuits and is emulated by most DNN software-models.

[3] Strictly speaking, this only holds for the so called *class-I* or *type-I* class of neuron models; *type-II* neurons have a discontinuous jump in their firing-rate response [105].

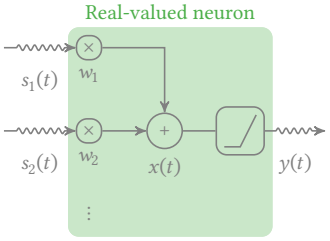


Figure 3.3. The linear-nonlinear neuron can be implemented in analog electronics.

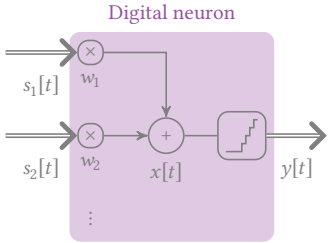


Figure 3.4. The neuron model in figure 3.3 can be discretized into a digital neuron model.

[4] Even in the brain, the longer-ranging synaptic connections of the white-matter alone can make up half of the cortex by volume [106].

3.2.3 (Leaky)-Integrate-and-Fire neurons are ΔΣ-modulators

Spiking neurons combine some benefits of analog computation (namely energy efficiency and speed) with the benefits of binary transmission (namely noise robustness)[5], because they process signals in the analog domain (whether in a biological neuron's dendrite or a neuromorphic circuit) while sending out only a series of binary pulses (see also chapter 6)!

[5] For a neuromorphic hardware designer, there is the additional benefit of being able to route the discrete spikes through a bus system.

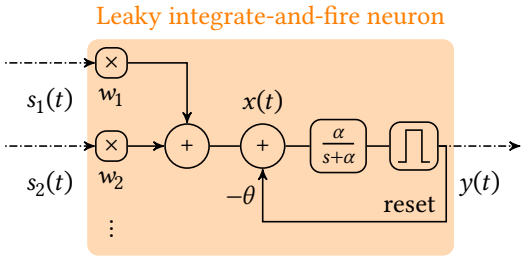


Figure 3.5. A leaky integrate-and-fire neuron converts analog or spiking input signals into a continuous-time spike-train. A negative feed-back loop resets the neuron after each pulse. The leaky integrator with leak-rate α is represented here by its Laplace-transform, $\alpha/s+\alpha$.

The simplest example of this is the well known *integrate-and-fire neuron*. In essence, it integrates its input(s) over time and fires a pulse whenever the integral $x[t]$ exceeds a threshold. A negative feed-back loop then resets the system, and the process begins anew. Instead of a perfect integrator, a first-order exponentially decaying filter is often used to describe the response of biological neurons, which accounts for the fact that absent any input, the neuron's membrane potential tends to return to its resting potential over time. This model is called the *leaky integrate-and-fire neuron* (LIF neuron, see figure 3.5). Instead of the exponential filter, other filters could be used as well, which affects the neuron's response in interesting ways that we'll discuss in chapter 4.

The *firing-rate* of such a neuron encodes the input in a very similar way to the linear-nonlinear neuron, it only uses a pulse-based code to transmit its output. For band-width limited signals, this encoding can be entirely lossless [107], but it may require rather large firing rates (see chapter 6 for a discussion).

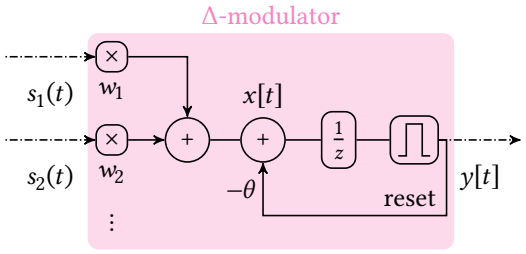


Figure 3.6. A first-order Δ-modulator converts analog input signals into a clocked sequence of binary pulses. A negative feed-back loop resets the component after each pulse. The discrete-time integrator is represented here by its z-transform, $1/z+1$.

Such pulse-based communication schemes are also popular in electronics. In fact, the LIF neuron directly resembles a very popular electronic circuit, the so-called ΔΣ-*modulator*, which is an integral part of ΔΣ analog-to-digital converters [108] and pulse-width-modulators (PWM). In this comparison, the dendrite of the LIF neuron corresponds to a *demodulator*, which converts, sums and integrates the pulse-based input signals into a single analog signal, while the spike-generation mechanism at the soma uses Δ-*modulation* to encode whenever this signal has increased beyond a fixed threshold. Figure 3.6 shows a schematic.

To improve the noise characteristics of analog-to-digital converters further, another feed-back can be added that subtracts the recent average output signal from the input and thus prevents the accumulation of quantization errors over time. This, too, has a direct counterpart in biological neuron models, namely the adaptive exponential integrate-and-fire neuron (AdExp, [109]), which also happens to be a more faithful representation of biological spiking neurons than the simpler LIF neuron [110]. We will discuss similar adaptation mechanisms in

chapter 5. By replacing the first-order integrator by higher-order filters, this can be improved further — an idea that we will also return to in chapter 4.

In an electronics context, the pulse-train generated by such a circuit constitutes an "over-sampled" digital *pulse-density modulated* (PDM) signal, which can then either be *decimated*, i.e. converted into a higher bit-precision signal at a reduced sampling rate, or directly transmitted over a digital connection. From this perspective, the (time-varying) density of the binary pulses (or *firing rate*) encodes the (time-varying) value of the analog signal. This view mirrors the *rate-coding* perspective, which we will look at in chapter 6.

A different perspective would be to treat the circuit as an event-detector, which emits a spike once it has accumulated enough input. We will discuss this alternative in chapter 7.

3.3 Closing the gap

For machine learning applications of artificial neural networks, the physical implementation of the individual neuron is of little concern — it represents an abstract mathematical function that is viewed as an "atomic operation" inside a larger algorithm. But from both a neuroscience and a neuromorphic computing perspective, the internal mechanisms that *generate* this behavior are of great interest. The two fields can therefore benefit from each other, by using engineering methods to investigate the function of biological neurons, or by taking inspiration from biological mechanisms for the development of a new generation of computing hardware. Ultimately, I believe these two disciplines ought to come together in a single discipline which I'll just refer to as *neuromorphic science* — the study of neuroscience-inspired physical mechanisms of information processing.

In the following, I will therefore occasionally use tools from signal processing and engineering to describe the behavior and information processing capabilities of biological neurons, and focus on the kind of questions that is also relevant for neuromorphic design. In chapter 7, I will then present a neuron model derived entirely from biological observations, along with an efficient neuromorphic circuit to implement it.

References for chapter 3:

1. F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958, ISSN: 1939-1471(ELECTRONIC),0033-295X(PRINT). DOI: 10.1037/h0042519 (cit. on pp. vii, 4, 14, 24).

18. **J. Leugering**, "A visit to the neuromorphic zoo," in *Embedded World Conference 2020 – Proceedings*, 2020, ISBN: 978-3-645-50186-6 (cit. on pp. viii, 24).

20. M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*, Expanded ed. MIT Press, 1988, ISBN: 978-0-262-63111-2 (cit. on pp. 1, 4, 11, 12, 26).

27. W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1, 1943, ISSN: 0007-4985, 1522-9602. DOI: 10.1007/BF02478259 (cit. on pp. 2, 10, 25).

74. M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*, 2016 (cit. on pp. 13, 26).

94. **J. Leugering**, "Neuromorphe Hardware," *DESIGN&ELEKTRONIK*, no. 7/2020, pp. 41–47, 2020 (cit. on p. 24).

95. T. N. Theis and H.-S. P. Wong, "The End of Moore's Law: A New Beginning for Information Technology," *Computing in Science Engineering*, vol. 19, no. 2, pp. 41–50, 2017, ISSN: 1558-366X. DOI: 10.1109/MCSE.2017.29 (cit. on p. 23).

96. W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing: Vision and Challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016, ISSN: 2327-4662. DOI: 10.1109/JIOT.2016.2579198 (cit. on p. 24).

97. Q. Xia and J. J. Yang, "Memristive crossbar arrays for brain-inspired computing," *Nature Materials*, vol. 18, no. 4, pp. 309–323, 4 2019, ISSN: 1476-4660. DOI: 10.1038/s41563-019-0291-x (cit. on p. 24).

98. G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, "Recent advances in physical reservoir computing: A review," *Neural Networks*, vol. 115, pp. 100–123, 1, 2019, ISSN: 0893-6080. DOI: 10.1016/j.neunet.2019.03.005 (cit. on p. 24).

99. D. Ielmini and H.-S. P. Wong, "In-memory computing with resistive switching devices," *Nature Electronics*, vol. 1, no. 6, pp. 333–343, 6 2018, ISSN: 2520-1131. DOI: 10.1038/s41928-018-0092-2 (cit. on p. 24).

100. G. W. Burr, R. M. Shelby, A. Sebastian, S. Kim, S. Kim, S. Sidler, K. Virwani, M. Ishii, P. Narayanan, A. Fumarola, L. L. Sanches, I. Boybat, M. L. Gallo, K. Moon, J. Woo, H. Hwang, and Y. Leblebici, "Neuromorphic computing using non-volatile memory," *Advances in Physics: X*, vol. 2, no. 1, pp. 89–124, 2, 2017, ISSN: null. DOI: 10.1080/23746149.2016.1259585 (cit. on p. 24).

101. S. Yin, Z. Jiang, J.-S. Seo, and M. Seok, "XNOR-SRAM: In-Memory Computing SRAM Macro for Binary/Ternary Deep Neural Networks," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 6, pp. 1733–1743, 2020, ISSN: 1558-173X. DOI: 10.1109/JSSC.2019.2963616 (cit. on p. 24).

102. C. D. Schuman, T. E. Potok, R. M. Patton, J. D. Birdwell, M. E. Dean, G. S. Rose, and J. S. Plank. (19, 2017). "A Survey of Neuromorphic Computing and Neural Networks in Hardware." arXiv: 1705.06963 [cs], [Online]. Available: http://arxiv.org/abs/1705.06963 (visited on 08/23/2020) (cit. on p. 24).

103. H. M. A. Andree, G. T. Barkema, W. Lourens, A. Taal, and J. C. Vermeulen, "A comparison study of binary feedforward neural networks and digital circuits," *Neural Networks*, vol. 6, no. 6, pp. 785–790, 1, 1993, ISSN: 0893-6080. DOI: 10.1016/S0893-6080(05)80123-6 (cit. on p. 26).

104. E. J. McCluskey, "Minimization of Boolean functions," *The Bell System Technical Journal*, vol. 35, no. 6, pp. 1417–1444, 1956, ISSN: 0005-8580. DOI: 10.1002/j.1538-7305.1956.tb03835.x (cit. on p. 26).

105. E. M. Izhikevich, *Dynamical Systems in Neuroscience: The Geometry of Excitability and Bursting*. The MIT Press, 2006, ISBN: 978-0-262-27607-8. DOI: 10.7551/mitpress/2526.001.0001 (cit. on pp. 26, 93).

106. B. Mota, S. E. D. Santos, L. Ventura-Antunes, D. Jardim-Messeder, K. Neves, R. S. Kazu, S. Noctor, K. Lambert, M. F. Bertelsen, P. R. Manger, C. C. Sherwood, J. H. Kaas, and S. Herculano-Houzel, "White matter volume and white/gray matter ratio in mammalian species as a consequence of the universal scaling of cortical folding," *Proceedings of the National Academy of Sciences*, vol. 116, no. 30, pp. 15 253–15 261, 23, 2019, ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.1716956116. pmid: 31285343 (cit. on p. 26).

107. A. A. Lazar and L. T. Tóth, "Time encoding and perfect recovery of bandlimited signals," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 6, 25, 2003 (cit. on pp. 27, 59, 73).

108. R. Gray, "Oversampled Sigma-Delta Modulation," *IEEE Transactions on Communications*, vol. 35, no. 5, pp. 481–489, 1987, ISSN: 1558-0857. DOI: 10.1109/TCOM.1987.1096814 (cit. on pp. 27, 63).

109. M. V. Nair and G. Indiveri, "An Ultra-Low Power Sigma-Delta Neuron Circuit," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2019. DOI: 10.1109/ISCAS.2019.8702500 (cit. on p. 27).

110. R. Brette and W. Gerstner, "Adaptive Exponential Integrate-and-Fire Model as an Effective Description of Neuronal Activity," *Journal of Neurophysiology*, vol. 94, no. 5, pp. 3637–3642, 1, 2005, ISSN: 0022-3077. DOI: 10.1152/jn.00686.2005 (cit. on p. 27).

A single neuron in the brain is an incredibly complex machine that even today we don't understand. A single "neuron" in a neural network is an incredibly simple mathematical function that captures a minuscule fraction of the complexity of a biological neuron. So to say neural networks mimic the brain, that is true at the level of loose inspiration, but really artificial neural networks are nothing like what the biological brain does.

— Andrew Ng

4 *Dendritic filters and delays*

In chapters 2 and 3, we discussed information processing in artificial neural networks of simple *point-neurons*. However, while these models offer a very convenient simplification, they don't account for the complex structure and behavior of real dendritic arbors, the behavior of which is better described by *neural cable theory* [111]. If we take the attenuation and delays into account that inevitably occur as membrane potentials are propagated along the dendrite, then even the location of a synapse on the spatially extended dendrite influences the effect of an input signal on the neuron's firing rate [112]! This makes the behavior much more difficult to describe, but it could also increase the computational complexity of the individual neuron considerably by endowing it with a notion of time or *memory*. Among the first to see the serious implications of this arrangement was, once again, von Neumann [35]:

> It may well be that certain nerve pulse combinations will stimulate a given neuron not simply by virtue of their number but also by virtue of the spatial relations of the synapses to which they arrive. That is, one may have to face situations in which there are, say, hundreds of synapses on a single nerve cell, and the combinations of stimulations on these that are effective (that generate a response pulse in the last-mentioned neuron) are characterized not only by their number but also by their coverage of certain special regions on that neuron (on its body or on its dendrite system, cf. above), by the spatial relations of such regions to each other, and by even more complicated quantitative and geometrical relationships that might be relevant. […] Lastly, I would like to mention that systems of nerve cells, which stimulate each other in various possible cyclical ways, also constitute memories. These would be memories made up of active elements (nerve cells).[35].

In a very similar sense, the delayed interaction of neurons by synaptic spikes, which could be seen as an imperfection of an idealized neuron and thus a nuisance to be avoided, also increases the complexity of the neuron's behavior by introducing long-lasting dependencies, i.e. *memory*. These complex nonlinear dynamics and long memory are very appealing for reservoir computing, where neurons with delayed feedback have recently been evaluated as a potential computational substrate. But like many other concepts mentioned in this thesis, the idea to use delayed self-coupling as a form memory and computing device is actually surprisingly old and goes all the way back to Turing and Copeland [4], who proposed building a computer based on the delayed interaction of shock waves in tubes of mercury:

> It is proposed to build 'delay line' units consisting of mercury […] tubes about 5' long and 1" in diameter in contact with a quartz crystal at each end. The velocity of sound in …mercury […] is such that the delay will be 1.024ms. The information to be stored may be considered to be a sequence of 1024 'digits' (0 or 1) […] These digits will be represented by a corresponding sequence of pulses. The digit 0 […] will be represented by the absence of a pulse at the appropriate time,

> the digit 1 [...] by its presence. This series of pulses is impressed on the end of the line by one piezo crystal, it is transmitted down the line in the form of supersonic waves, and is reconverted into a varying voltage by the crystal at the far end. This voltage is amplified sufficiently to give an output of the order of 10 volts peak to peak and is used to gate a standard pulse generated by the clock. The pulse may be again fed into the line by means of the transmitting crystal, or we may feed in some altogether different signal. We also have the possibility of leading the gated pulse to some other part of the calculator, if we have need of that information at the time. Making use of the information does not of course preclude keeping it also [4].

Therefore, two aspects of biological neurons that are often disregarded as a nuisance, *dendritic filtering* and *synaptic transmission delays*, could theoretically serve an important purpose for neural computation. But is all this complexity really instrumental, or is it just an inevitable side effect of some biological process, through which nature approximates a much simpler mechanism? In this chapter, we'll make use of some simple tools from signal processing to investigate the role of dendritic filtering and delays for neural information processing.

4.1 Terminology

Before we get started, I'd like to introduce a few terms that are used inconsistently across different disciplines: The general mathematical formalism we'll use is temporal *convolution*, which is also called *filtering* in engineering domains. Since physical systems cannot be retroactively affected by future events, *causal filters* play a special role, which are linear integral operators that only depend on the values of the signal in the recent past. Such a filter operator K can be applied to a continuous-time signal s by the convolution

$$\mathrm{K}s(t) = (s * \kappa)(t) = \int_{-\infty}^{t} s(\tau)\kappa(t-\tau)\mathrm{d}\tau$$

where κ is the *kernel* or *impulse-response function* of K. *Delays* are a special case of causal filtering that can be represented by the convolution $(s * \kappa)(t) = s(t - \Delta t)$ with a shifted Dirac-δ-distribution kernel of the form $\kappa(t) = \delta(t - \Delta t)$, where Δt is the duration of the delay. Conversely, we can interpret a continuous filter kernel κ as the limiting case of a linear combination of delay terms (for a derivation, see appendix A.1). This equivalence is used in the (digital) signal processing domain to design filters for *periodically sampled* discrete-time signals [113, p. 67]. Therefore, filtering and delays are really two sides of the same coin!

I will also use the one-sided Laplace transform $\mathscr{L}\{\kappa\}(s) = \int_0^\infty \kappa(t)\exp(-st)\mathrm{d}t$, which is closely related to the Fourier transform, to represent a filter with kernel $\kappa(t)$ in the frequency domain [114]. The Laplace transform can simplify the analysis greatly, because it allows us to represent a concatenation of multiple filters in the frequency domain simply by the multiplication $\kappa_1(s) \cdot \kappa_2(s) \cdot \kappa_3(s) \cdot \kappa_4(s) \cdots$ of their individual transformations, rather than by the convolution $\kappa_1(t) * \kappa_2(t) * \kappa_3(t) * \kappa_4(t) \ldots$ in the time domain. If it is clear from context, I will just write $\kappa(t)$ and $\kappa(s) := \mathscr{L}\{\kappa\}(s)$ to denote the filter kernel in the time- or the Laplace-domain, respectively.

4.2 Dendritic filtering improves information transmission

Why should we care about the filtering effect of dendrites, in the first place? For one, because including it greatly increases the accuracy with which the firing-rates of biological neurons can be approximated when compared to simpler point neurons [3, 115]. But more importantly, filtering offers an opportunity to improve the information transmission and

processing capabilities of individual neurons. This same argument can be made in multiple ways, and we'll look at five different perspectives in the following.

4.2.1 Denoising

From the perspective of information theory, the linear-nonlinear neuron is a noisy channel with limited capacity to transmit information (see also chapter 2 of [7] and the later chapter 5). This is formalized in the Shannon-Hartley theorem [7, 113], which states that an analog channel's ability to transmit information (in some frequency band) is limited by the *channel capacity* $C = B\log_2(1+R)$, where B is the analog channel bandwidth and R is the *signal-to-noise-ratio*, i.e. the ratio of the expected power of the signal to be transmitted relative to the power of the independent noise signal. Since the amount of transmitted information adds up linearly across distinct frequency bands, it follows that a carefully chosen filter can selectively amplify or attenuate different frequency bands to boost the signal while suppressing the noise. Therefore, a key benefit that an appropriately chosen dendritic filter can offer for information transmission is to improve the signal-to-noise-ratio and thus the channel capacity of the neuron. [1] In general, it will be impossible to completely filter out all noise this way, because the spectra of signal and noise are likely to overlap. But the signal-to-noise ratio can always be optimized by the so-called (causal) *Wiener filter* or *matched filter* [116], which shapes the spectrum in a way that maximizes the relative power of the signal while minimizing that of the noise.

[1] The simplest example is when the signal and the noise occupy distinct frequency bands altogether, in which case a simple band-pass filter can be used to fully isolate the signal and suppress the noise.

4.2.2 Pattern detection and sparse coding

Denoising can also be understood in a rather different way. Let's consider the special case that the signal $s(t)$ consists only of repetitions of some stereotypical pattern $g(t), 0 \le t \le T$ with duration T subject to white noise $\eta(t)$, i.e.

$$s(t) = \sum_{\tau_i \le t} g(t-\tau_i) + \eta(t) = (g * \chi)(t) + \eta(t) \quad \text{with } \chi(t) = \sum_{\tau_i \le t} \delta(t-\tau_i).$$

Then the power spectrum of the noise is flat, and the kernel of the Wiener filter simplifies to the time-reversed pattern $h(t) = g(T-t)$. *Denoising* the signal s with this filter h yields the signal

$$(h * s)(t) = (r * \chi)(t) + (h * \eta)(t) \quad \text{where } r(t) = (h * g)(t) = \int_0^t g(\tau)g(\tau + T - t)\mathrm{d}\tau \text{ for } t \le t.$$

So whenever the stereotypical pattern g is seen in the input, the filter responds with r, the *autocorrelation function* of g, which always has a distinct maximum at $t = T$. These peaks in the signal can be easily detected by an appropriately chosen threshold despite the noise, because filtering the white noise η with the same kernel merely results in colored noise with comparatively lower amplitude. Therefore, a neuron with appropriate dendritic filter can become an efficient *pattern detector* (with delay T) for the stereotypical pattern g. Detectors of this sort have long been used, for example, in radar systems to detect reflected radio pulses of known shape [116], and the same ideas transfer to other time-series signals. In chapter 7, we'll contrast this to a rather different type of pattern detector.

4.2.3 Deconvolution

The problem of pattern detection can also be approached from a different perspective: Imagine that in the same setting as in section 4.2.2 we'd like the filter h to directly return $(h * s)(t) = \chi(t) + (h * \eta)(t)$ rather than $(r * \chi)(t) + (h * \eta)(t)$. This inverse problem, called

deconvolution, is unfortunately generally ill-conditioned[2] and outright impossible if we are limited to causal filters. [3] But we could introduce another filter q and try to solve the relaxed problem $h * s = q * \chi + h * \eta$ instead, i.e. $h * g = q$. If we choose q well, this can be (approximately) solved for h even if g cannot be inverted. Looking back at section 4.2.2, we can e.g. choose $q = r$, which again gives us the matched filter $h(t) = g(T - t)$. Or we could choose the filter q to approximate a delay-line with delay $T' > T$, which would allow us to approximate a *delayed* deconvolution (with delay T')!

[2] For a discrete convolution operator, which can be represented by a matrix, this can be done through iterative matrix-inversion methods with strong regularization [117], and it becomes more complex for continuous-time operators.

[3] Just consider, for example, the kernel $\delta(t - \tau)$ with a delay $\tau > 0$. Inverting this filter would imply inverting the time-shift, which would require an acausal "negative delay"!

4.2.4 Equalization

Another way of looking at the same idea is the *equalization* [118] (sometimes also called *whitening*) of signals, which removes temporal correlations from the signal itself and thus leads to an *equalized* or flat power spectrum, resembling that of *white noise* (hence the name). This is done in the context of communication systems with bandwidth-limited communication channels, where the most information can be transmitted if all the available spectral bandwidth is used to convey relevant (i.e. non-predictable) information. A very similar argument can be made for the neuron as well, and we'll return to this idea also in chapter 5 when we talk about optimal firing rate distributions.

4.2.5 Predictive Coding

A more biologically motivated perspective is *predictive coding* (see e.g. chapter 6 of [7]), which argues that the dendritic filter can subtract predicted *future* inputs, leaving only the residual error to be transmitted by the neuron. The benefit of such an encoding is, again, that (given a sufficiently good prediction) these residuals are temporally decorrelated (i.e. equalized), which leads to an information theoretically and metabolically efficient encoding [7, chapter 6]. This has been experimentally observed in visual [119, 120] and auditory neurons [121], as well as in other modalities and animals [6], and is believed to play an important role for neural information processing in general [122].

4.3 Dendritic filtering in the linear-nonlinear model

We saw that, when we view the neuron as a signal-processing device, the ability to implement dendritic filters is extremely attractive. We will now investigate, how this could be implemented on a mechanistic level.

If we are willing to ignore all the non-linear effects that can occur in neural dendrites, such as dendritic plateau potentials,[4] and instead focus exclusively on the linear effects described by neural cable theory [111], then the behavior of a dendrite can be approximated by assigning a specific impulse response to each synapse, depending on its location on the dendrite. The somatic membrane potential can then be approximated by a linear combination of these differently filtered synaptic inputs. The result is a more general type of linear-nonlinear neuron model, which I'll just call the "filter-nonlinear" model in the following (see figure 4.1).

[4] I actually believe that these non-linear effects are absolutely crucial for dendritic computation, and I argue for this position in chapter 7.

A very simple special case of this model is Rall's so-called *ball-and-stick model* [123]: Under certain simplifying assumptions (e.g. specific relationships between the thickness of dendritic branches), the effect of a synaptic input onto the soma's membrane potential only depends on the *distance* of the synapse to the soma — the complex tree-shaped topology of the dendrite can be ignored altogether. In this case the dendrite therefore behaves equivalently to a single cylindrical dendrite as shown in the top panel of figure 4.5. This cylinder can be approximated by a chain of multiple *compartments*, each of which receives and filters input from its "up-stream" neighbor compartment as well as synaptic inputs. How many of these

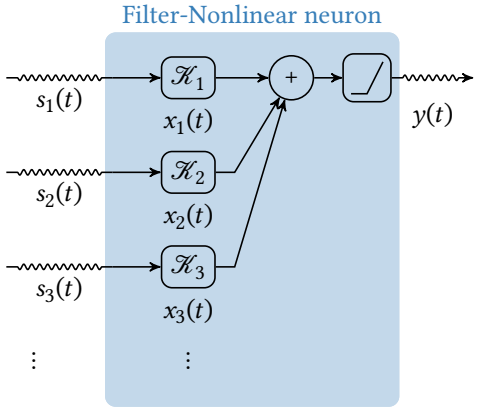


Figure 4.1. A linear-nonlinear neuron, where the dendrite is modeled by a set of filters $\mathscr{K}_i$. Each synaptic input $s_i(t)$ now produces a different post-synaptic membrane potential $x_i(t)$, which are then summed up and non-linearly transformed at the soma.

compartments a synaptic input has to traverse depends on the distance of the synapse to the soma, therefore the dendrite can be also viewed as a *filter bank*. See the middle panel of figure 4.5 for an illustration.

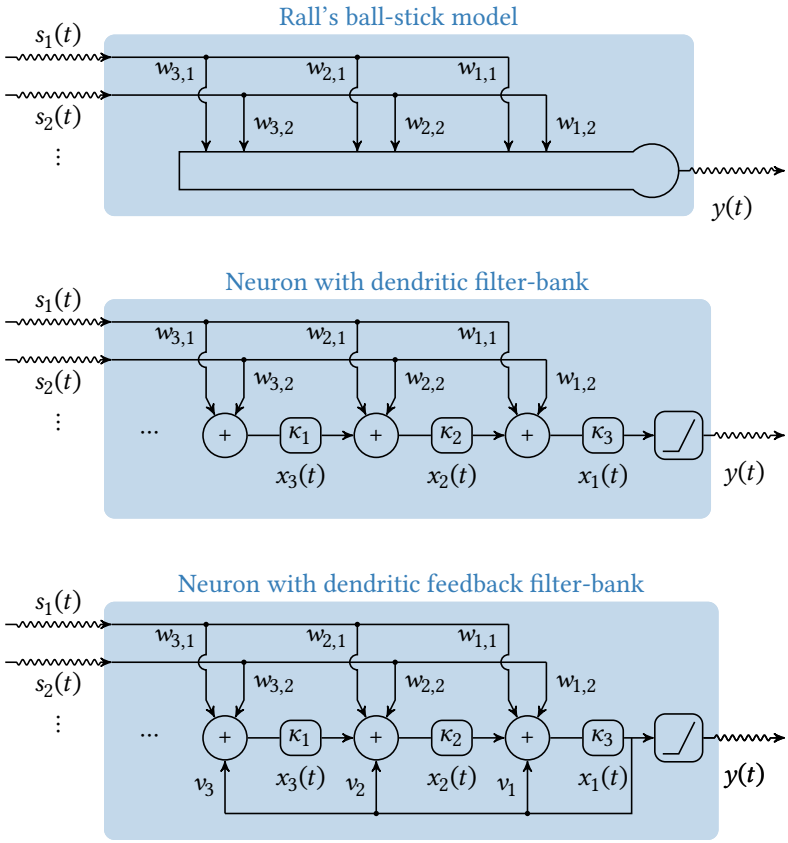


Figure 4.2. **Top:** The ball-and-stick model [123] abstracts the neuron's dendritic arbor into a single "equivalent cylinder" or *cable*, on which the propagation of activity can be modeled by a partial differential equation. The impulse response of an input signal depends on the location along the cylinder. **Middle:** A neuron with a dendrite modeled by a tapped filter-bank composed of individual filters κ_i. Each tap of the filter-bank provides a local state-variable $x_i(t)$. The neuron's output $y(t) = f(x_1)$ is then just a non-linear function of the somatic membrane potential x_1. **Bottom:** By adding a linear feedback term, more complex filters can be constructed.

Just like forward-propagation, the effects of backward-propagation of membrane potential in the *retrograde* direction, i.e. away from the soma, can be also incorporated by adding linear feedback terms (see the bottom panel of figure 4.5). In this model, the weights of the (feed-forward) input and the weights of the feedback term parameterize a family of dendritic filters that can approximate the filtering effect of a dendritic tree. [5]

In the following, we'll look at two particularly simple and relevant implementations of such a dendritic filter banks:

[5] Curiously, this type of dendritic filter with feed-back resembles a well known topology of infinite impulse response filters for electrical signal processing [124]!

4.4 Dendritic filtering in the Gamma Neuron

If we implement the dendrite shown in figure 4.5 as a bank of identical first-order low-pass filters with transfer function $\kappa_i(s) = \frac{\alpha}{s+\alpha}$ and time-constant α, then this much simpler multi-compartment model can be described by a system of *ordinary* differential equations instead of Rall's original *partial* differential equation. The resulting neuron is shown in figure 4.3.

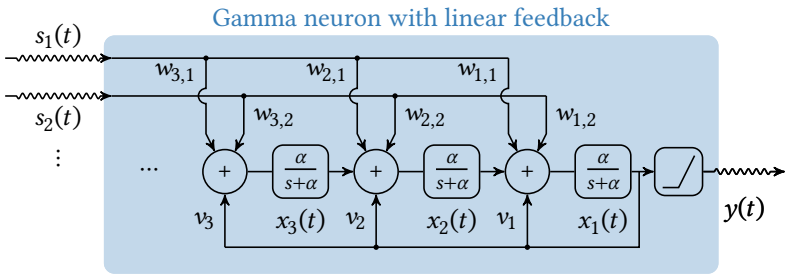


Figure 4.3. The Gamma Neuron uses a filter-bank composed of identical first-order filters with transfer function $\frac{\alpha}{s+\alpha}$ to model the dendrite. Inputs are projected onto the filter taps through multiple synapses with weights $w_{j,i}$. Similarly, feed-back paths can be added with weights v_j. These weights parameterize the dendritic filter. A nonlinear activation function is applied to the filter output.

The low-pass filter response of each tap resembles the simplified sub-threshold dynamics of the leaky integrate-and-fire neuron, and the filter bank can hence be interpreted as a chain of weakly coupled *dendrite compartments*. The result is the versatile *Gamma-neuron* [125], which can be represented by a particularly simple system of ordinary differential equations. This model owes its name to the fact that a synaptic spike arriving at the k^{th} compartment from the soma would be subjected to the transfer function $(\alpha/s+\alpha)^k$, which is the Laplace transform of the density function of some Gamma distribution. The impulse response of the dendrite therefore becomes broader and broader with every additional compartment that a spike has to traverse on its way to the soma (see figure 4.4).

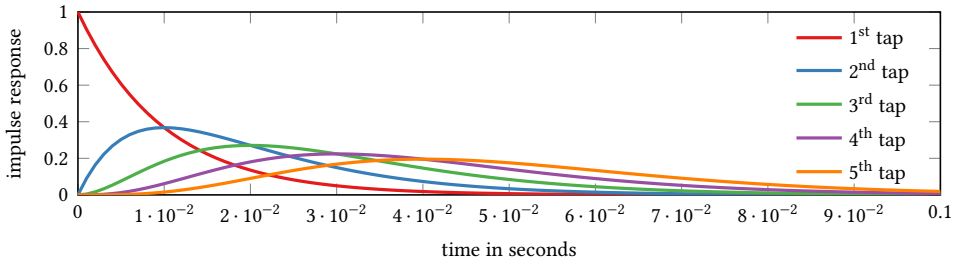


Figure 4.4. Open-loop impulse-responses of the somatic membrane potential in response to synaptic input spikes received at various taps along the dendrite. The impulse responses have the form of the probability density function of Gamma distributions, hence the name *Gamma Neuron*.

Now, if the same input signal s_i arrives at various taps j along the dendrite through multiple synapses with weights $w_{j,i}$, then the total effect of the dendrite on that input is a linear combination of the individual taps' responses. By adding a feedback path with additional coefficients v_i, we can extend this neuron model to allow much more complex filters to be implemented by the dendrite [126]. This family of filters can be formalized nicely (see the note below) and includes a lot of interesting special cases: For example, low-, high- and band-pass filters can be implemented with just two taps, and with increasing order the (open-loop) impulse response of the taps more and more resembles Gaussian filters. Also, for any filter κ that can be implemented by a Gamma Neuron, the derivative κ' can be trivially implemented as well. See appendix A.3 for derivations.

If we put all of this together, a single Gamma neuron could therefore theoretically

(a) extract relevant frequency-bands from each of its multiple input signals,

(b) equalize each signal to extract the maximum amount of information,

(c) calculate derivatives or ("leaky") integrals thereof,

Note: The ring of Gamma filters

The filters that can be implemented by the Gamma neuron span a finite-dimensional function space, parameterized by the input and feedback weights. With a bit of algebra, we can make this more precise (see appendix A.2 for a derivation): The space of filters that can be implemented by the Gamma neuron corresponds to exactly those with a *proper rational transfer function* in the Laplace domain. The addition and convolution (i.e. concatenation) of two implementable filters yields another implementable filter, but the inverse κ^{-1} of an implementable filter κ is in general not a proper rational function (i.e. κ^{-1} is acausal) and thus not implementable. Therefore, the class of filters that *can* be implemented by such a filter bank forms a *commutative ring* (or rather *pseudo-ring*) without multiplicative identity and inverse. This space of filters is extremely general; it contains *all* analog linear filters that can be implemented by networks of lumped electric elements, i.e. discrete resistors, capacitors and inductors [127], and *any* transfer function can at least be well approximated by such a rational function, also called a Padé approximant [128].

(d) and linearly combine them into a single signal that is then

(e) passed through a nonlinearity.

This also makes the individual neuron at least as powerful as a *PID controller* [129], a versatile tool from control theory and much more impressive than the simple logic gates we saw in chapter 3!

In contribution 4, we extend this model further and employ a synaptic plasticity rule to train individual (spiking) Gamma neurons to detect specific temporal patterns in their input.

Contribution 4: Training the Gamma Neuron for event detection

We extended the Gamma Neuron to a spike-based temporal pattern detector for a conference poster presented first at the Cognitive Computing 2018 conference in Hannover, Germany, and then again at the Machine Learning Summer-School (MLSS) 2019 held in Cape Town, South Africa. Here we investigated how this type of neuron model could be trained to produce a spike-based classification of temporal patterns through a local, reward-modulated synaptic learning rule.

Reference (see also appendix C, page 135ff for the full text):

P. Nieters, **J. Leugering**, and G. Pipa, "Neuromorphic Adaptive Filters for event detection, trained with a gradient free online learning rule," presented at the Machine Learning Summer School (MLSS-Africa 2019), 1, 2019.

4.5 Computing with synaptic delays

We already saw above that delays are just a special case of causal filtering and *vice versa*, but delays deserve special treatment in the study of neural systems. They are a fact of life, since no physical system can respond *instantaneously* to its input — and of course neurons are no exception. However, even minuscule delays can make otherwise benign dynamical systems

difficult to control or even chaotic [130], which is why they are often seen as a nuisance to be avoided by theoreticians and engineers alike. But biological neurons are inherently *analog* machines that work *asynchronously* and in real time, so we have no choice but to recognize and understand the effect of delays on their dynamics. As we shall see, this might be a blessing in disguise, since there are even (somewhat surprising) ways in which delays might actually *improve* the computational capabilities of neurons and networks!

To understand *how* delays, e.g. caused by synaptic transmission, could be used constructively by biological neurons, we need to make a brief detour into (digital) signal processing and control theory.

4.5.1 *Delay-embeddings, state-estimation and Koopman-control*

So far, we looked at continuous-time signals and filters, but a lot of the intuitions about information transmission come from the study of *sampled discrete-time* systems. The connection is established by the *Nyquist sampling theorem* [118, 124], which states that any analog bandwidth limited signal can be fully represented without loss of information by *samples* of the signal, if they are measured at a sufficiently high finite sampling rate. In that context, the same effect that a continuous filter would have on a continuous signal can be achieved by filtering the sampled signal with a discrete-time filter, which can be implemented by a linear combination of the outputs of a *tapped delay-line*. [6] Just like in the Gamma neuron, a linear feedback loop can be used to extend these filters, which allows us to also construct *infinite impulse response filters*. See [118, 124, 131] for an introduction into filter design and tapped delay-lines.

[6] This is analogous to how we used a continuous filter bank above to construct the dendritic filter of the Gamma neuron.

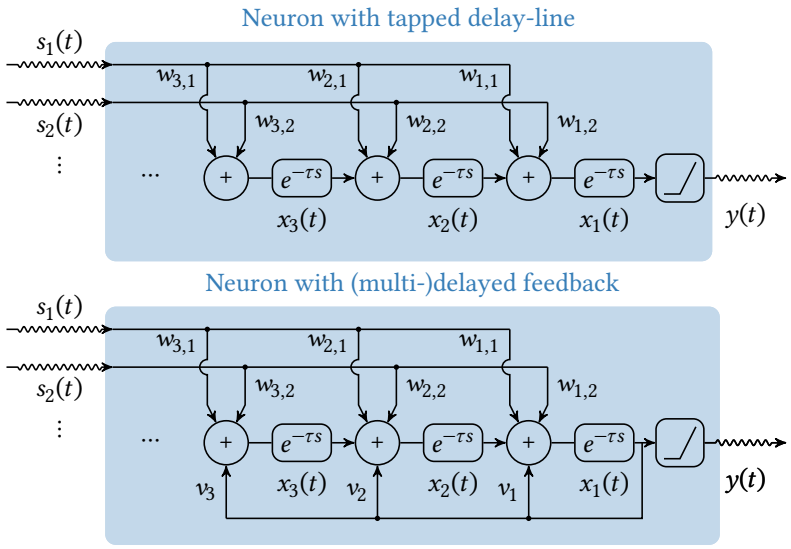


Figure 4.5. **Top:** A neuron with dendritic or synaptic delays modeled by a tapped delay-line composed of individual delay elements with transfer function $e^{-\tau s}$ and identical delay τ. This topology can implement finite impulse-response filters. **Bottom:** By adding a linear feedback term, infinite impulse-response filters can be constructed.

But of course, tapped delay-lines can also be utilized in a continuous-time setting. In that context, we'd say that the outputs of all the taps constitute a *delay embedding* of the signal — a higher dimensional representation of the signal and its recent past. Such an embedding contains a lot of information about the signal that can, for example, be used to estimate derivatives or to forecast the signal into the future. See also the note below. Under the more general heading of *embedding theory*, this has many practical applications for the study of dynamical systems, signal analysis and causality. Schumacher [132] has a great discussion of this subject. One fairly recent application of these ideas has been in control theory, specifically *Koopman control* [133–135], which comprises many state-of-the-art approaches to controlling non-linear systems by the use of delay embeddings.

Note: Derivatives, finite differences and delay embeddings

To give an intuitive example, how delay embeddings can be used to extract relevant information from a continuous-time signal, let's consider the definition of (higher-order) derivatives. The left derivative of a function f can be defined as $f'(t) = \lim_{\Delta t \to 0} \frac{f(t)-f(t-\Delta t)}{\Delta t}$. So if we can produce a delayed signal $\tilde{f}(t) = f(t-\Delta t)$ for a small delay Δt, we can use the signals f and $\tilde{f}$ to continuously estimate the derivative f'. In fact, the same idea can be applied repeatedly: if $f'(t) \approx \frac{f(t)-f(t-\Delta t)}{\Delta t}$ and $f'(t-\Delta t) \approx \frac{f(t-\Delta t)-f(t-2\Delta t)}{\Delta t}$, then $f''(t) \approx \frac{f'(t)-f'(t-\Delta t)}{\Delta t} \approx \frac{f(t)-2f(t-\Delta t)+f(t-2\Delta t)}{\Delta t^2}$, and so on. We can thus estimate the first N derivatives of a signal by a linear combination of $N+1$ delayed versions with delays $k \cdot \Delta t, k \in \{0, 1, 2, \dots, N\}$. I'll let $\delta^k(t)$ denote the delay line $\delta(t - k\Delta t)$. The mapping

$$f(t) \to \left((\delta^0 * f)(t) \quad (\delta^1 * f)(t) \dots\dots (\delta^N * f)(t)\right)$$

is then a so-called *delay embedding*, which embeds the one-dimensional time-varying signal f into an $N+1$-dimensional space. This can be implemented by a *tapped delay-line* composed of N concatenated delay elements, each with the same delay Δt. From this embedding, approximate derivatives of order $\leq N$ can be trivially read out by linear combinations of different taps' outputs. Therefore, an n-tap delay-embedding contains enough information about the signal to approximate a Taylor-approximation of order $n-1$ at the current point in time!

4.5.2 *Delayed nonlinear feedback*

So far, we only discussed linear systems with delay, but what if we include nonlinear feedback-loops? The result is a *nonlinear delay-* or *retarded differential equation*, which has an infinite dimensional state-space and can exhibit extremely complex, if not chaotic, behavior. The inherent complexity of such systems with nonlinear delayed feedback can introduce very long-lasting memory effects (see also [130] for more examples). [7]

[7] One example of such systems are *feedback shift registers*, whose very long memory is used for the generation of maximally long sequences of non-repeating pseudo-random numbers [136]!

4.5.3 *Filter- or multi-delay-coupled reservoir computing*

The idea to use the long memory of systems with delayed nonlinear feedback also underlies an admittedly weird neuron model called the *single node, multi-delay-coupled reservoir computer* (SNMDCR). It is an extension of the slightly simpler *single node, delay-coupled reservoir computer* (SNDCR), which is situated between machine learning, neuroscience and optical neuromorphic hardware [137].[8] We extend this model in contribution 5 to use *multiple* delayed feedback terms (hence the slight change in name), which considerably increases the neuron's ability to learn complex temporal dynamics. The resulting neuron model is summarized in figure 4.6.

[8] The original physical realization of the system was implemented by self-coupled lasers with optical delay elements.

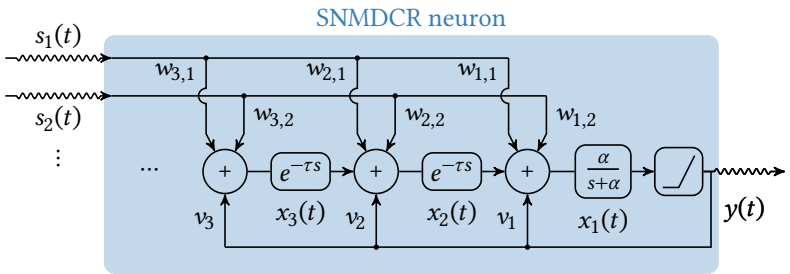


Figure 4.6. A *single-node multi-delay-coupled reservoir*. It resembles the neuron in figure 4.3, but with delay elements instead of exponential filters for all but the first tap. Each tap delays the signal by τ. Note that the feedback-signal is here taken *after* the nonlinearity, so the behavior of the entire system is no longer of the simple linear-nonlinear form.

Contribution 5: Neuromorphic computation in multi-delay coupled models

In this paper, we explored how delayed feedback, in particular the interaction between differently delayed feedback-loops, can be exploited to endow a single neuron, which could be implemented in an electrical or photonic circuit, with memory and the capability to compute complex functions of its input history. For a simple single-node multi-delay-coupled reservoir neuron, we show how the relationship between the delay terms leads to different complexity of behavior, and hence different performance of the trained neuron across different time-series regression tasks. Curiously, we can show that — and why — co-prime delays result in the best performance, and thus give some intuition for the complex behavior of delay-coupled systems.

Reference (see also appendix C, page 136ff for the full text):

P. Nieters, **J. Leugering**, and G. Pipa, "Neuromorphic computation in multi-delay coupled models," *IBM Journal of Research and Development*, vol. 61, no. 2/3, 8:7–8:9, 1, 2017, ISSN: 0018-8646, 0018-8646. DOI: 10.1147/JRD.2017.2664698.

Due to the combination of continuous dynamics and delayed feedback, the SN(M)DCR must be modeled by delay-differential equations, and it shows highly complex if not chaotic behavior, depending on the precise choice of the relative delays. We use this neuron with its complex dynamics as a substrate for reservoir computing, i.e. we inject various task-specific input signals into the neuron and use a weighted linear combination of the delay embedding of the neuron's output as a *readout*. As usual, (only) these weights are optimized such that the readout approximates the desired output signal of the task. To investigate the impact of the precise choice of delays on the ability of the SNMDCR to produce interesting behavior, we look at a neuron with just two feedback paths with different delays τ_1 and τ_2. We systematically vary one of the two delays τ_2 while keeping the other fixed, and for each choice of τ_2 optimize the neuron's weights for some simple task, like estimating the N-bit parity of a binary signal or approximating a fixed NARMA model of a continuously-valued signal. Remarkably, the SNMDCR performs very well on either task, but its performance critically depends on the relative timing of the two delays and deteriorates whenever this ratio approaches a ratio of small integers such as $1:1, 1:2, 2:3$ etc. Since the SNMDCR is described in discrete time and the delays are integer multiples of these time-steps, we can compare the location and magnitude of the performance drops to the greatest common divisor of the two delays and find a clear correspondence.

To get a better mechanistic understanding of how and why the SNMDCR works (and when it fails), we analyze how it integrates and recombines information over time, and conclude that co-prime delays provide the best "mixing" over time with the longest memory, which appears to be the critical factor for performance on these tasks. While these results are specific for an unusual type of neuron model and cannot be directly transferred to others[9], they nevertheless provide a useful intuition: Dendritic filtering and synaptic delays can be used to not only extract relevant information from time-varying signals, but also to improve information transmission, provide volatile memory, and implement computation within a single neuron!

[9] For example, in the continuous-time context of biological neurons, the inherently discrete concepts like co-primality and greatest common divisors are not applicable.

4.6 Dendritic filtering in the real world

We have seen for a couple of examples above how a neuron could, *in principle*, make constructive use of the delays and filtering effects introduced by synaptic transmission and the dynamics of ion currents in the dendrite. A natural question to ask now is: How much, what for and how, if at all, do biological neurons *actually* use dendritic filtering? A second question is: Should we use dendritic filtering in machine learning models of neural networks?

On the one hand, there has been a lot of biological evidence that shows dendrites using delays and filtering to do much more than just instantaneous linear combinations of incoming signals. For example, the distance-dependent filtering and delaying effect of dendrites has long been proposed as a critical feature for binaural localization of sounds [138]. There is even some evidence that these transmission delays can be fine-tuned by controlling the myelinization of axons [139], which would represent an entirely new form of plasticity mechanism! The huge theoretical potential that dendritic filtering can offer for processing time-series signals also makes it likely that evolution would have found ways to exploit it in some way.

On the other hand, it is also tempting to entirely brush off the intimidating complexity of biological dendrites as functionally irrelevant "implementation details", and there is also biological evidence to support this view. For example, the attenuation along the dendrite appears (in some cases) to be precisely counteracted by some other mechanism like synaptic scaling or "synaptic democracy" [140, 141]. This could ensure that each synaptic spike, regardless of its location on the dendrite, has the same effect on the somatic membrane potential. Through such regulatory mechanisms, an apparently complex nonlinear neuron could produce a rather simple linear behavior that is well described by the point-neuron model, after all. [10] But more critically, the real time-constants of dendritic filtering and synaptic delays (on the order of microseconds to tens of milliseconds) might just not be long enough to implement most useful filters on behaviorally relevant time-scales. And on a more fundamental level, the strongly nonlinear effects that can be observed within small dendritic branches call the assumption into question, that the temporal dynamics can be well approximated by *linear* filters, at all! Instead, the interaction of active, localized nonlinear processes within the dendrites need to be taken into account [142]. We will return to this point in detail in chapter 7.

[10] However, such a regulatory mechanism might only affect the amplitude, not the delay, of a synaptic input as a function of its location on the dendrite. In that case, the arguments of this section could still be applied.

It might therefore turn out, that biological neurons make only limited and rather specific use of (linear) dendritic filtering, e.g. for the purpose of adaptation (see chapter 5) and for the processing of spiking inputs (see chapter 6), while relying on different mechanisms on the neuron or network level, such as active dendritic processes (see chapter 7) or recurrent networks (see chapter 2) for more sophisticated temporal integration and processing of information.

Of course, for machine learning models and neuromorphic hardware these biological constraints do not apply, and dendritic filtering is certainly worth considering. However, the introduction of dendritic filters (in analog or digital hardware as well as in software) complicates the individual neuron substantially, and the additional slow dynamics makes temporal credit assignment difficult. Models like the Gamma neuron are therefore more difficult to simulate and to train using conventional gradient-based methods (see contribution 4). I therefore see the most promising applications of these ideas in conjunction with local learning rules (e.g. for unsupervised equalization of signals) and/or in the context of analog neuromorphic hardware, where filtering is inevitable and can be efficiently realized by simple electronic circuits.

References for chapter 4:

3. W. Maass and C. M. Bishop, *Pulsed Neural Networks*. MIT Press, 2001, 414 pp., ISBN: 978-0-262-63221-8. Google Books: jEug7sJXP2MC (cit. on pp. vii, 32, 59, 61–63, 73).

4. A. M. Turing and B. J. Copeland, *The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life, plus the Secrets of Enigma*. Clarendon Press ; Oxford University Press, 2004, ISBN: 978-0-19-825079-1 978-0-19-825080-7 (cit. on pp. vii, 3, 31, 32).

6. S. (O. N. Laughlin University Of C, *Principles of Neural Design*. 2017, ISBN: 978-0-262-53468-0 (cit. on pp. vii, 17, 34, 51, 60, 66).

7. J. V. Stone, *Principles of Neural Information Theory: Computational Neuroscience and Metabolic Efficiency*. Sebtel Press, 2018, 214 pp., ISBN: 978-0-9933679-2-2 (cit. on pp. vii, 33, 34, 46, 55, 60, 62, 66, 71, 98).

11. P. Nieters, **J. Leugering**, and G. Pipa, "Neuromorphic computation in multi-delay coupled models," *IBM Journal of Research and Development*, vol. 61, no. 2/3, 8:7–8:9, 1, 2017, ISSN: 0018-8646, 0018-8646. DOI: 10.1147/JRD.2017.2664698 (cit. on pp. viii, 40).

19. P. Nieters, **J. Leugering**, and G. Pipa, "Neuromorphic Adaptive Filters for event detection, trained with a gradient free online learning rule," presented at the Machine Learning Summer School (MLSS-Africa 2019), 1, 2019 (cit. on pp. viii, 37).

35. J. von Neumann, *The Computer and the Brain*. Yale University Press, 1958, ISBN: 978-0-300-08473-3 978-0-300-00793-0 978-0-300-02415-9 (cit. on pp. 3, 31).

111. C. Koch, "Cable theory in neurons with active, linearized membranes," *Biological Cybernetics*, vol. 50, no. 1, pp. 15–33, 1, 1984, ISSN: 1432-0770. DOI: 10.1007/BF00317936 (cit. on pp. 31, 34).

112. B. F. Behabadi, A. Polsky, M. Jadi, J. Schiller, and B. W. Mel, "Location-Dependent Excitatory Synaptic Interactions in Pyramidal Neuron Dendrites," *PLOS Computational Biology*, vol. 8, no. 7, e1002599, 19, 2012, ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1002599 (cit. on p. 31).

113. J. B. Anderson and R. Johnnesson, *Understanding Information Transmission*. John Wiley & Sons, 17, 2006, 323 pp., ISBN: 978-0-471-71119-3. Google Books: GD5GY4XyPXIC (cit. on pp. 32, 33, 46, 48, 60, 61).

114. J. L. Schiff, *The Laplace Transform: Theory and Applications*. Springer, 1999, 233 pp., ISBN: 978-0-387-98698-2. Google Books: N_jZBwAAQBAJ (cit. on p. 32).

115. K. Doya, S. Ishii, A. Pouget, and R. P. N. Rao, *Bayesian Brain: Probabilistic Approaches to Neural Coding*. MIT Press, 2007, 341 pp., ISBN: 978-0-262-04238-3. Google Books: bsQMWXXHzrYC (cit. on pp. 32, 77).

116. G. Turin, "An introduction to matched filters," *IRE Transactions on Information Theory*, vol. 6, no. 3, pp. 311–329, 1960, ISSN: 2168-2712. DOI: 10.1109/TIT.1960.1057571 (cit. on p. 33).

117. P. C. Hansen, "Deconvolution and Regularization with Toeplitz Matrices," p. 56, (cit. on p. 34).

118. M. D. Adams, *Continuous-Time Signals and Systems (Edition 2.0)*. Michael Adams, 29, 2020, 400 pp., ISBN: 978-1-55058-658-9. Google Books: BWPXDwAAQBAJ (cit. on pp. 34, 38).

119. M. V. Srinivasan, S. B. Laughlin, A. Dubs, and G. A. Horridge, "Predictive coding: A fresh view of inhibition in the retina," *Proceedings of the Royal Society of London. Series B. Biological Sciences*, vol. 216, no. 1205, pp. 427–459, 22, 1982. DOI: 10.1098/rspb.1982.0085 (cit. on p. 34).

120. T. Hosoya, S. A. Baccus, and M. Meister, "Dynamic predictive coding by the retina," *Nature*, vol. 436, no. 7047, pp. 71–77, 7047 2005, ISSN: 1476-4687. DOI: 10.1038/nature03689 (cit. on p. 34).

121. E. C. Smith and M. S. Lewicki, "Efficient auditory coding," *Nature*, vol. 439, no. 7079, pp. 978–982, 7079 2006, ISSN: 1476-4687. DOI: 10.1038/nature04485 (cit. on pp. 34, 51).

122. L. Aitchison and M. Lengyel, "With or without you: Predictive coding and Bayesian inference in the brain," *Current Opinion in Neurobiology*, vol. 46, pp. 219–227, 1, 2017, ISSN: 0959-4388. DOI: 10.1016/j.conb.2017.08.010 (cit. on p. 34).

123. W. Rall, "Electrophysiology of a Dendritic Neuron Model," *Biophysical Journal*, vol. 2, no. 2, pp. 145–167, 1962, ISSN: 00063495. DOI: 10.1016/S0006-3495(62)86953-7 (cit. on pp. 34, 35).

124. M. K. Mandal and A. Asif, *Continuous and Discrete Time Signals and Systems*. Cambridge University Press, 2007, 865 pp., ISBN: 978-0-521-85455-9 (cit. on pp. 35, 38, 91).

125. B. de Vries and J. C. Principe, "The gamma model—A new neural model for temporal processing," *Neural Networks*, vol. 5, no. 4, pp. 565–576, 1992, ISSN: 08936080. DOI: 10.1016/S0893-6080(05)80035-8 (cit. on pp. 36, 89, 91).

126. J. Principe, B. de Vries, and P. de Oliveira, "The gamma-filter-a new class of adaptive IIR filters with restricted feedback," *IEEE Transactions on Signal Processing*, vol. 41, no. 2, pp. 649–656, 1993, ISSN: 1941-0476. DOI: 10.1109/78.193206 (cit. on pp. 36, 89).

127. A. I. Zverev, *Handbook Of Filter Synthesis*. 1967 (cit. on p. 37).

128. G. A. Baker, G. A. B. Jr, G. Baker (A.), P. Graves-Morris, and S. S. Baker, *Pade Approximants: Encyclopedia of Mathematics and It's Applications, Vol. 59 George A. Baker, Jr., Peter Graves-Morris*. Cambridge University Press, 26, 1996, 762 pp., ISBN: 978-0-521-45007-2. Google Books: Vkk4JNLKbLoC (cit. on p. 37).

129. Kiam Heong Ang, G. Chong, and Yun Li, "PID control system analysis, design, and technology," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 4, pp. 559–576, 2005, ISSN: 1558-0865. DOI: 10.1109/TCST.2005.847331 (cit. on pp. 37, 92).

130. H. Smith, *An Introduction to Delay Differential Equations with Applications to the Life Sciences*. Springer New York, 2011, vol. 57, ISBN: 978-1-4419-7645-1 978-1-4419-7646-8. DOI: 10.1007/978-1-4419-7646-8 (cit. on pp. 38, 39).

131. B. A. Shenoi, *Introduction to Digital Signal Processing and Filter Design/ B.A. Shenoi*. Wiley, 2006, ISBN: 978-0-471-46482-2 978-0-471-65442-1 (cit. on p. 38).

132. J. Schumacher, "Time series analysis informed by dynamical systems theory," Institute of Cognitive Science, 2015 (cit. on p. 38).

133. J. L. Proctor, S. L. Brunton, and J. N. Kutz, "Generalizing Koopman Theory to Allow for Inputs and Control," *SIAM Journal on Applied Dynamical Systems*, vol. 17, no. 1, pp. 909–930, 2018, ISSN: 1536-0040. DOI: 10.1137/16M1062296 (cit. on p. 38).

134. M. Kamb, E. Kaiser, S. L. Brunton, and J. N. Kutz. (14, 2020). "Time-Delay Observables for Koopman: Theory and Applications." arXiv: 1810.01479 [cs, math], [Online]. Available: http://arxiv.org/abs/1810.01479 (visited on 08/23/2020) (cit. on p. 38).

135. E. Kaiser, J. N. Kutz, and S. L. Brunton. (19, 2020). "Data-driven discovery of Koopman eigenfunctions for control." arXiv: 1707.01146 [math], [Online]. Available: http://arxiv.org/abs/1707.01146 (visited on 08/23/2020) (cit. on p. 38).

136. J. Massey, "Shift-register synthesis and BCH decoding," *IEEE Transactions on Information Theory*, vol. 15, no. 1, pp. 122–127, 1969, ISSN: 1557-9654. DOI: 10.1109/TIT.1969.1054260 (cit. on p. 39).

137. J. Schumacher, H. Toutounji, and G. Pipa, "An Introduction to Delay-Coupled Reservoir Computing," in *Artificial Neural Networks*, P. Koprinkova-Hristova, V. Mladenov, and N. K. Kasabov, eds., vol. 4, Springer International Publishing, 2015, pp. 63–90, ISBN: 978-3-319-09902-6 978-3-319-09903-3. DOI: 10.1007/978-3-319-09903-3_4 (cit. on p. 39).

138. H. Agmon-Snir, C. E. Carr, and J. Rinzel, "The role of dendrites in auditory coincidence detection," *Nature*, vol. 393, no. 6682, pp. 268–272, 6682 1998, ISSN: 1476-4687. DOI: 10.1038/30505 (cit. on p. 41).

139. R. D. Fields, "A new mechanism of nervous system plasticity: Activity-dependent myelination," *Nature reviews. Neuroscience*, vol. 16, no. 12, pp. 756–767, 2015, ISSN: 1471-003X. DOI: 10.1038/nrn4023. pmid: 26585800 (cit. on pp. 41, 60).

140. M. Häusser, "Synaptic function: Dendritic democracy," *Current Biology*, vol. 11, no. 1, R10–R12, 9, 2001, ISSN: 0960-9822. DOI: 10.1016/S0960-9822(00)00034-8 (cit. on p. 41).

141. C. C. Rumsey and L. F. Abbott, "Synaptic Democracy in Active Dendrites," *Journal of Neurophysiology*, vol. 96, no. 5, pp. 2307–2318, 1, 2006, ISSN: 0022-3077. DOI: 10.1152/jn.00149.2006 (cit. on pp. 41, 76).

142. A. Polsky, B. W. Mel, and J. Schiller, "Computational subunits in thin dendrites of pyramidal cells," *Nature Neuroscience*, vol. 7, no. 6, pp. 621–627, 6 2004, ISSN: 1546-1726. DOI: 10.1038/nn1253 (cit. on pp. 41, 76).

The green reed which bends in the wind
is stronger than the mighty oak
which breaks in a storm.

— Confucius

'[A]daptive' behaviour is equivalent to the behaviour of a stable system

— W. Ross Ashby, *Design for a Brain*

5 *Homeostatic plasticity*

In the previous chapters, I presented networks and neurons as information processing "machines" and likened them to logic gates and other electronic components. But while logic gates are fed a steady diet of ones and zeros, nervous systems are embedded in biological organisms, and they are bombarded by noisy input signals from an ever-changing environment, perceived through sensors that themselves develop or degrade over time. To keep working in this chaotic setting requires the organism to take active counter-measures to *maintain itself*. This ability to adapt to changes is so critical for survival that early cyberneticists like Ashby [2] saw the concept of *homeostasis* as *the* defining feature of life, and one of the main differences that sets life (and nervous systems) apart from dead matter (and logic gates). The term "homeostasis" entails that some attribute(s) of the organism are maintained at a desirable state, and that the system can recover this state from small disturbances through some self-regulating mechanism. In the case of the nervous system, that could mean to remain functional (or to quickly regain functionality) even if certain aspects of the sensory inputs change abruptly. The idea of homeostatic adaptation is therefore quite central in theoretical and computational neuroscience, and it should play an important role for our understanding of (artificial) intelligence as well. In fact, a survey of different definitions of intelligence [143] found that at least 23 out of the 72 definitions see the ability to adjust or adapt to the environment as a defining feature, and many of the others imply it!

But in most current deep learning research, adaptation plays only a minor role, and most of the big data sets on which models are trained and evaluated have been explicitly preprocessed to remove any of the systematic changes or drifts that would require the system to adapt in the first place[1]. Since self-regulating, adaptive systems are also typically harder to understand, control and train than static ones, the homeostatic plasticity mechanisms that we know from biological neurons are hence still largely absent from machine learning models.

In this chapter, I'd like to illustrate why homeostatic adaptation is not just a biological necessity, but also a useful mechanism for neural information processing in general. I'll present an abstract framework that unifies two different forms of biological plasticity mechanisms to solve a practical, easily interpretable machine learning problem. Most content of this chapter is directly based on contribution 6, but it offers another, hopefully simpler motivation for the main results, while leaving out a lot of the technicalities here. Nevertheless, this will take us through a number of abstract mathematical concepts, and I will try my best to explain them here on a rather high level.

[1] Many image recognition challenges, for example, present a fixed set of training images in randomized order. Often, these images are color-adjusted, scaled to equal size, centered or otherwise prepared. It would be considerably harder, if the training set was allowed to change over time.

Contribution 6: A Unifying Framework of Synaptic and Intrinsic Plasticity in Neural Populations

In this rather long paper, I explore the relationships and interaction of intrinsic and synaptic plasticity for computation. The entire chapter 5 of my thesis is largely based on ideas contained within this publication. I try to motivate the same main results here using a slightly different approach that introduces concepts like optimal transport theory. But for most content of the current chapter, a more in-depth discussion can be found within this original publication.

Reference (see also appendix C, page 145ff for the full text):

J. Leugering and G. Pipa, "A Unifying Framework of Synaptic and Intrinsic Plasticity in Neural Populations," *Neural Computation*, vol. 30, no. 4, pp. 945–986, 17, 2018, ISSN: 0899-7667. DOI: 10.1162/neco_a_01057.

5.1 The Information Bottleneck Principle

The Information Bottleneck Principle [66, 144] loosely states that in neural networks, the capacity to transmit information from neuron to neuron, or from layer to layer, or from region to region, is often the limiting factor for computation — a bottleneck, so to speak. As such, the capacity should be used providently, and neural computation should be optimized to make efficient use of it. This provides a clear objective, towards which a neuron or network can be optimized: to convey as much information as possible about the input signal through an information channel with limited capacity. A popular and quite literal example of this idea are auto-encoders [145] in deep learning, where a feed-forward network is supposed to transmit its input signals without loss to its output layer, but with one important twist: some intermediate layers of the network contain only few neurons and thus present an information bottleneck. In order to reproduce the network's input signal on its output, the network layers leading up to this bottleneck must find a very low-dimensional, compressed representation of the input (encoding), which the subsequent layer can then decode again. This is illustrated in figure 5.1. By forcing the network to find such a low-dimensional *latent space* representation of its input, we can make sure that the network picks up only on the *most informative* features of its input — or so the story goes.

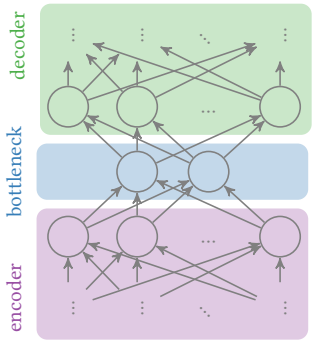


Figure 5.1. An auto-encoder with a narrow bottleneck in between the encoder and decoder layer(s).

As we already saw in chapter 4, the same principle can be also applied at a much smaller level, the individual neuron, which needs to reduce its high-dimensional input signal from thousands of incoming synaptic connections to the only one-dimensional output signal with finite bandwidth ([146]; see figure 5.2). Particularly when metabolic constraints and noise are considered, the capacity of individual neurons to transmit information is limited and a loss of information becomes inevitable. If we apply the information bottleneck principle to the single neuron, the neuron should be tuned to ensure that as much (relevant) information as possible about its inputs is preserved in its output. Example 1 gives some intuition for this idea.

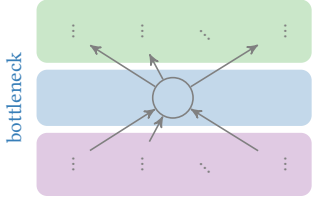


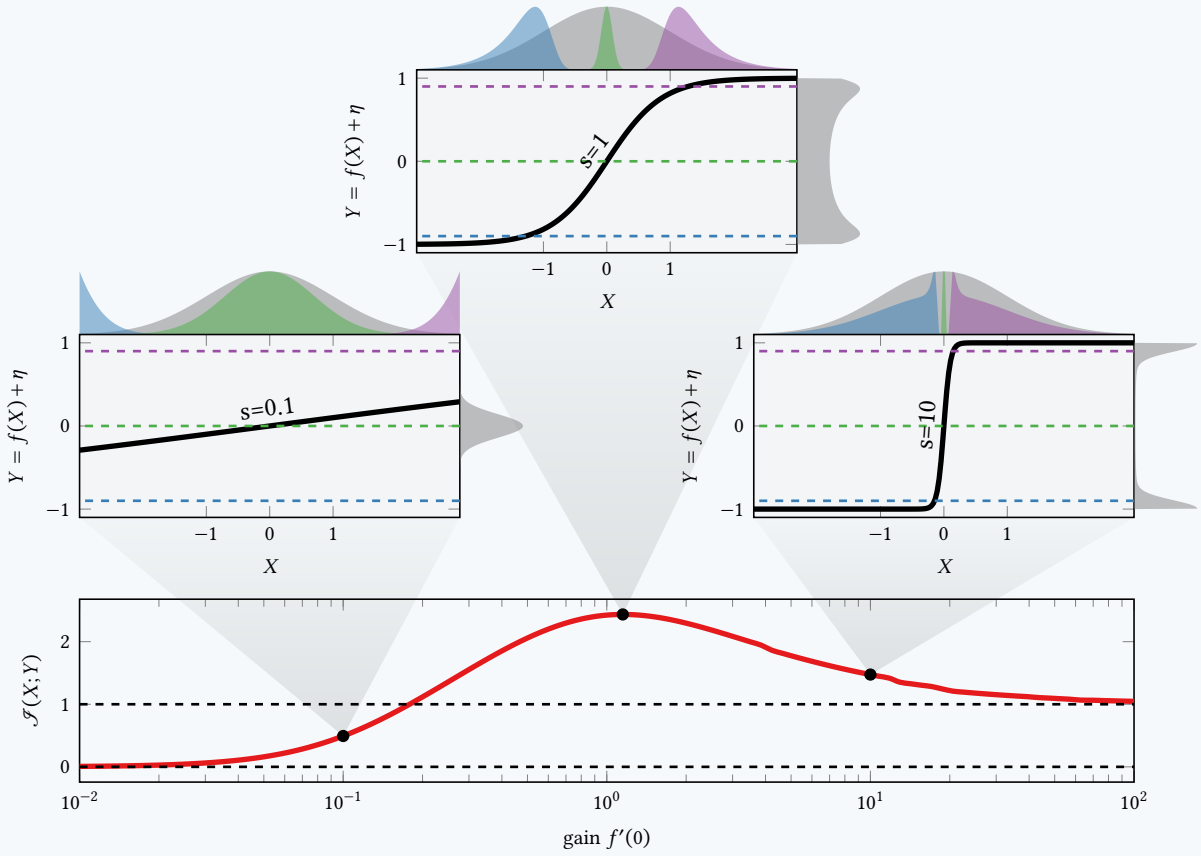
Figure 5.2. A single neuron acts as an information bottleneck in between its synaptic inputs and outputs.

5.2 Mutual information and maximum entropy

From an *information theoretical* perspective [7, 113, 147], the neuron represents a *noisy channel*, and the information bottleneck problem is a matter of maximizing the mutual

Example 1: Gain modulation and the information bottleneck

Consider a linear-nonlinear neuron of the form $f(x) = \tanh(s \cdot x)$ with a single free parameter, the slope $s = f'(0)$ at the origin. We assume that the output signal $Y = f(X) + \eta$ produced by the neuron is corrupted by additive noise η. A *perfect* recovery of the input signal X from the corrupted output signal is Y impossible, so the neuron becomes a lossy, non-linear noisy channel. How well this channel can transmit information depends on the choice of the gain s. To illustrate this, let's consider three different neurons, one with low, medium and high gain each *(see the three insets below, from left to right)*:



Each neuron receives input with the same prior distribution $X \sim \mathcal{N}(0, 1)$ *(gray, above each inset)*, which results in different output distributions $P(Y)$ *(gray, right of each inset)*. If we now observed the three output values $y_1 = 0.9$, $y_2 = 0.0$ and $y_3 = -0.9$ *(horizontal dashed lines)*, we can infer the conditional input probability distributions $P(X|y_i)$ *(shown above in corresponding colors)*. For a neuron with a low gain of $s \approx 0.1$, these conditional input distributions are quite broad and uninformative. For a slope of $s \approx 1$, each of the three outputs encodes a distinct, narrower input distribution. For a very steep slope $s \approx 10$, the observations $y_1 = 0.9$ or $y_3 = -0.9$ again reveal only little about X — mostly just whether it was positive or negative.

The red curve below quantifies this dependence of the neuron's information transmission $\mathcal{I}(X;Y)$ on the gain parameter: As $s \to 0$, f becomes constant, and the transmitted information content approaches 0 bits. In the other extreme, when the nonlinearity approaches a step-function ($s \to \infty$), the output distribution becomes sharply bimodal, and conveys only the sign-bit of the input signal. For some intermediate *optimal slope* ($s \approx 1$), however, the neuron's output yields a maximum of about 2.5 bits of information about its current input. Modulating the gain can therefore help to mitigate the information bottleneck!

information between the channel's in- and output. This critically depends on the statistical properties of the source signals to be transmitted and the noise affecting the channel, as well as the parameters of the channel itself. If we think of the standard linear-nonlinear neuron model with an invertible activation function, then we can express the mutual information as follows:

$$I(X;Y) = I(\bar{Y};Y) = h(\bar{Y}) - h(\bar{Y}|Y),$$

where X is the neuron's input (or membrane potential), $\bar{Y} = f(X)$ is its *noiseless* output, Y is the noisy signal that is ultimately received by the next neuron, $h(\bar{Y})$ is the differential entropy of $\bar{Y}$ and $h(\bar{Y}|Y)$ is the conditional differential entropy of $\bar{Y}$ *given that* Y *has been observed*, i.e. the uncertainty of our decoding of $\bar{Y}$ from the noise-corrupted version Y.

In the absence of noise, the channel's capacity to transmit information is only limited by the *source entropy* $h(\bar{Y})$ — this is Shannon's famed first theorem (*source-coding* theorem)[113, 147]. To maximize information transmission by a noiseless channel, we therefore need to use an *"encoding"* $\bar{Y} = f(X)$ that results in a maximum entropy distribution of $\bar{Y}$.

Note: Subtleties of differential entropy

For the continuously valued case we are interested in, there is one extra caveat to consider [147]: Since the differential entropy $h(\bar{Y})$ can be arbitrarily increased by just scaling $\bar{Y}$ (in fact, $h(\alpha \cdot Y) = h(Y) + \log(|\alpha|)$), the absolute value of the differential entropy is typically meaningless, as it depends on the choice of units and scales of the variables of interest. This is fundamentally different from discrete entropy, which is invariant to *any* invertible transformation![147]. When we talk of maximizing differential entropy $h(\bar{Y})$, we therefore always include direct or indirect constraints on the scale of the random variable and focus not on the absolute value of this maximum, but only on the distribution that achieves it (this only requires comparing differences of differential entropy, in which case the scale-dependent terms cancel.).

In practice, any physical channel is subject to noise that reduces its capacity, and the neuron is of course no exception. This is the core of Shannon's even more famous second theorem (*channel-coding* theorem)[113, 147], which establishes the limit of how much information can be transmitted through the channel in the presence of noise. This upper limit can be increased by improving the *signal-to-noise* ratio, either by allocating more bandwidth to the signal (e.g. scaling up firing rates) and/or by suppressing the noise (e.g. by filtering, see also chapter 4).

Optimizing the encoding while simultaneously taking into account the statistical properties of the source signal and the characteristics of the channel is called *joint source-channel-coding* and can be quite challenging. Luckily for us, the *joint source-channel separation theorem* [147] suggests[2] that an optimal solution to this problem can be found by separately optimizing the source encoding (which only depends on the distribution of source signals) followed by a channel-specific encoding (which only depends on the characteristics of the channel). In the following, we will therefore limit ourselves to the simpler problem of source-coding, i.e. we'd like to find the distribution of neural outputs $\bar{Y}$ with the largest differential entropy $h(\bar{Y})$ under certain *metabolic constraints* imposed by the channel.

[2] The theorem assumes discrete channels, and does not perfectly translate to continuously valued signals [148].

For biological spiking neurons, these constraints could be a finite maximum firing rate of the neuron, an energy constraint on the mean firing rate, or even a constraint that depends nonlinearly on the firing rate. For neuromorphic hardware, they could be finite supply voltages or limits on the energy dissipation.

To incorporate such metabolic constraints, we need to find the output distribution P^* with the largest entropy subject to a list of equations or inequalities of the form $E_{P(y)}[g_i(y)] = c_i$ for $i \in I$ and $E_{P(y)}[g_i(y)] \geq c_j$ for $j \in J$.[3] Finding such a measure P^* might seem like a daunting task, but fortunately there is a beautiful solution to this very problem by Jaynes [149], generalizing results attributed to Ludwig Boltzmann. It states that the optimal distribution P^* is always from an exponential family with a probability density p^* that can be expressed directly in terms of the constraints:

[3] To ensure that the result is a valid probability distribution, there is always one additional equality constraint $0 \in I$ with $g_0 \equiv 1$ on the domain of P and $c_0 = 1$.

$$
\begin{aligned}
p^*(y) &= \exp\left(\sum_{i \in I \cap J} \lambda_i^* g_i(y)\right) \\
\text{where } \lambda^* &= \operatorname{argmax}_\lambda \left(\sum_{i \in I \cap J} \lambda_i c_i - \int \exp\left(\sum_i \lambda_i g_i(y)\right) dy\right) \\
\text{subject to } \forall j \in J &: \lambda_j \geq 0
\end{aligned}
$$

In contribution 6, we look at some examples of such maximum entropy distributions. The fact that the resulting distributions are all from some exponential family has a lot of interesting implications and comes in handy for the analysis. We will revisit this in chapter 6 and contrast it to a different approach, which aims to maximize *metabolic efficiency* of information transmission rather than maximizing information transmission under metabolic constraints. Here, we will just continue with the knowledge that we can in principle derive the optimal distribution of the output of a linear-nonlinear neuron under metabolic constraints according to the information bottleneck principle, and that it takes the form of some exponential family distribution.

5.3 Optimal Transport and the Monge Problem

We saw above how the neuron's activation function shapes the neuron's output distribution and thus its ability to transmit information. I then showed how metabolic constraints determine the optimal output distribution. Putting these results together, finding an activation function that produces that desirable output distribution would improve information transmission, and thus constitute a solution to the bottleneck problem. But what would this activation function look like for different input distributions? And what is the best approximation that a neuron could realize?

Again luckily for us, more general versions of this optimization problem, the *Monge-Kantorovich*, *Kantorovich-Rubinstein* or *Optimal Transport Problem* [150], have been studied extensively. It can be loosely paraphrased as the problem to find the 'best' deterministic transformation to map one given probability density onto another given probability density. What 'best' means in this context is precisely defined by a *cost* function $c(x, y) : \mathcal{X} \times \mathcal{Y} \to [0, 1]$ that penalizes the *transport* of probability mass from x to y. In the historic setting, in which this question was originally posed, c quite literally referred to the cost of moving earth from one spot x to another spot y, which is also where the alternate name *earth mover's distance* originates from [150]. The original Monge problem can be expressed in terms of this cost function as an optimization problem:

$$
f^* = \operatorname{arginf}_{f_\#(\mu)=\nu} \int c(x, f(x)) d\mu(x),
$$

where μ and ν are the source and target the probability distribution, respectively, and $f_\#(\mu)$ is the distribution onto which μ is mapped by f (i.e. the push-forward measure of μ under the function f).

In our context, we would like the neuron's activation function to deviate as little as possible from a linear function (in part because this makes the decoding simpler), so we choose $c(x, y)$ to penalize any deviation of $y = f(x)$ from x. The cost function is then some radial function $c(||x - y||)$ that only depends monotonically on the distance between x and y in some norm $\|\cdot\|$. Under generous assumptions, which our cost function satisfies, [4] it turns out that a unique optimum f^* exists and has the following simple form — regardless of the precise choice of c [150, Remark 2.30]:

$$f^* = F_{Y^*}^{-1} \circ F_X,$$

where F_X and F_{Y^*} are the cumulative probability distribution of X and Y^*, respectively.

Combining this result with the maximum entropy approach above, we therefore know the *optimal activation function* f^* that will maximize the neuron's ability to transmit information for a given input distribution! And since f^* is defined purely in terms of the distributions F_X and F_{Y^*}, any parameter of these distributions becomes a parameter of f^*. Conveniently for us, the optimal function f^* is also an acceptable candidate for an activation function of a neuron, since it is monotonically increasing and continuous if F_X is continuous and F_{Y^*} is injective. [5] In contribution 6, I derive the same solution, albeit from a very different perspective, and discuss in more detail the properties of this functional mapping; a related derivation can also be found in [151].

[4] Both p_X and p_Y must be atom-free, univariate, continuous probability distributions.

[5] For every root $p_{Y^*}(y) = 0$, f^* has a discontinuity at $x \in F_X^{-1}(\{y\})$.

Note: Kantorovich's relaxation [150]

The existence of a *deterministic one-to-one* mapping f^* as above is no longer guaranteed if we allow discontinuous probability distributions. But *even if* such a solution exists, the activation function f can become highly nonlinear. If we wish to avoid that while also maintaining the desired output distribution, we can express the optimization problem in a more general form or *relaxation* proposed by Kantorovich [150]. Rather than a deterministic one-to-one function f^*, it defines a probabilistic mapping $\gamma^* : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$, where $\gamma^*(x, y)$ is the relative amount of probability mass to be transferred from x to y:

$$\gamma^* = \operatorname{arginf}_{\gamma \in \Gamma(p_X, p_Y)} \int c(x, y) d\gamma(x, y)$$

Here $\Gamma(p_X, p_Y)$ is the set of all joint probability densities with marginal distributions p_X and p_Y, respectively. Such a map always exists (consider for example the trivial case $\gamma(x, y) = p_X(x)p_Y(y)$), but the optimal map is not (necessarily) the graph of a deterministic function from x to y, i.e. such that $\gamma(x, y) = \delta(x - f(y))$, as the stricter Monge-problem would require. In this non-deterministic case, the same input x is instead "distributed" probabilistically over possible outputs y with distribution $p_{y|x} = \gamma(x, y)$. This means that in order to enforce a desired output distribution, we could also include randomness into the neuron model itself and control it in an input dependent way. But whether this is useful or not is a separate topic I will not discuss here.

Let me summarize these results: in order to be an efficient information channel, a neuron should tune its activation function f to map its input $X \sim P_X$ to an output $Y^* = f(X)$ with maximum entropy distribution $Y^* \sim P^*$. The optimal way to achieve that is to set $f \leftarrow f^* = F_{Y^*}^{-1} \circ F_X$. This is also extremely useful if we are interested in optimizing spike-based information transmission, which we will return to in chapter 6.

5.4 Intrinsic homeostatic plasticity

These derivations all describe the mapping of a single random variable, X, onto a single random variably Y by the neuron's activation function f, but a real neuron is faced with a continuously varying input signal $X(t)$ which must be expressed as a stochastic process. [6] And what if the probability distribution of $X(t)$ were to suddenly change? In order to achieve and maintain the optimal output distribution, we'd expect the neuron to adjust to any changes in its input distribution by homeostatically regulating its nonlinear activation function in real time. This finally brings us to the main topic of this chapter, *intrinsic homeostatic plasticity*.

If we take for granted that a neuron can approximate the information-theoretically optimal input-output mapping in principle, the challenge for homeostatic plasticity is to *keep* the mapping optimal, i.e. to constantly adjust the coefficients of the activation function to changes in the environment that affect the input distribution. However, since the current probability distribution over input values is determined by external factors, it must be constantly *inferred* by the neuron from the recent history of its own input signals. In order to do this with a finite number of variables, the input distribution has to be approximated by some parameterized family of distributions, the time-varying parameters of which have to be estimated *online*, e.g. by the concentrations of some chemicals or voltage traces.

In contribution 6, I model the neuron's fast-changing membrane potential $X(t)$ by a continuous stochastic process with a given *stationary probability distribution* from some exponential family, e.g. a Gaussian. Changes due to environmental factors are assumed to occur sporadically on a much slower time-scale, which I model as sudden changes in the stationary distribution of the process. I show that despite the much more complicated mathematics involved with stochastic processes, the intuitions derived above for the probability distributions of in- and output can in fact be applied directly to the *stationary* distribution of the stochastic in- and output processes. Thus, by transforming the stochastic process that describes the neuron's membrane potential through some nonlinear function f, we can produce a stochastic process with any desired stationary distribution as the neuron's output, including the maximum entropy distribution that solves the information bottleneck problem.

By assuming a parameterized family of both the stationary input and output distributions, the optimal activation function also becomes parameterized. Since we are working with a stationary membrane potential distribution from an exponential family, these parameters are determined by the distribution's so-called sufficient statistics.

These sufficient statistics all take the form of an expected value of some nonlinear function of the process, which we can therefore estimate by filtering, e.g. with an exponentially weighted continuously running average. [7] I prove in contribution 6 that as the running estimates of the sufficient statistics approach the true values[8], the realized output distribution also approaches the desired output distribution. Since the filtering of the sufficient statistics constitutes a form of running average, the neuron will thus quickly recover from a perturbation to its input distribution! Example 2 illustrates this mechanism for the simple example of Gaussian inputs. While this description has focused on continuous linear-nonlinear models, the same arguments can be extended to spiking neurons, which are known to adapt to their input distributions, e.g. by regulating the spike-threshold [153]. Stabilizing a neuron's output by adjusting to the statistical properties of its input can also promote sparsity and might help explain the emergence of complex cell receptive fields in visual cortex [154]. This form of dynamic re-scaling has been directly observed in biological neurons in vision [155, 156], olfaction [157], audition [121], and might play an important role for neural information processing, in general [6].

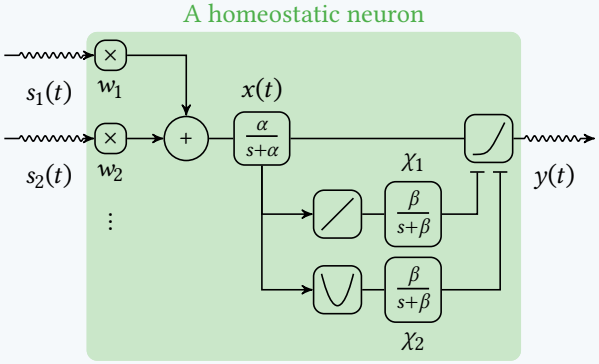
[6] I only look at *drift-diffusion processes*, which have a stationary distribution from some exponential family [152].

[7] If the mean value itself is one of these sufficient statistic (as e.g. for the Gaussian distribution), the estimation and homeostatic regulation of that parameter could be realized entirely by dendritic filtering, as discussed in chapter 4.

[8] They do this in an unbiased way, but with some residual uncertainty that depends on the estimator's time-constant. The longer the time constant is, the better the approximation becomes at the cost of a longer latency.

Example 2: A homeostatic neuron for Gaussian inputs

With the popular choice of an Ornstein-Uhlenbeck process [152] as a model of the membrane potential X, the stationary distribution is Gaussian with sufficient statistics $s_1 = \mathbb{E}[X]$ and $s_2 = \mathbb{E}[X^2]$. Two internal state variables χ_1 and χ_2 provide a running estimate of s_1 and s_2, respectively. In order to produce an output with cumulative distribution function F_Y, the neuron nonlinearly transforms its membrane potential through the function f.



The traces χ_1 and χ_2 are used to parameterize the activation function

$$f(x) := F_Y^{-1}(F_X(x)) \approx F_Y^{-1}\left(1 + \operatorname{erf}\left(\frac{x - \chi_1}{\sqrt{2\left(\chi_2 - \chi_1^2\right)}}\right)\right)$$

The neuron operates on two time-scales defined by the fast time-constant α of the membrane potential dynamics, and the slower time-constant β of the adaptation process. Any shift in mean or variance of the input distribution is counteracted by the neuron on the slower timescale — the neuron exhibits homeostatic self-regulation.

5.5 *The complex interactions of synaptic and intrinsic plasticity*

Of course, intrinsic plasticity mechanisms that adjust the neuron's response are not the only form of neural plasticity. The most critical mechanisms for learning appear to be structural and synaptic plasticity [158], which lead to the (dis-)appearance of synaptic connections (or dendritic spines) and an adjustment of the synaptic efficacy, respectively. Each of these forms of plasticity, intrinsic to the neuron or occurring within each synapse, only have access to different information and can thus influence the behavior of neurons in different ways.

Each synapse can, in principle, modulate its transmission strength (or transmission probability) based on the activity of the two neurons it connects, while each neuron, through *intrinsic plasticity*, can only adjust its nonlinearity based on the neuron's membrane potential. To see where a combination of both rules leads, we investigate the dynamics of a neuron's membrane potential and its synaptic weights under the effect of both intrinsic and synaptic plasticity in contribution 6.

5.5.1 *Principal Component Analysis*

Consider as an example a neuron with two synaptic inputs, which evolve according to a (non-linear) Hebbian rule with weight decay of the form $\frac{1}{\eta}\frac{\mathrm{d}w_{j,i}(t)}{\mathrm{d}t} = f(y_i(t))g(y_j(t)) - w_{j,i}(t)$, where f and g are increasing functions, $w_{j,i}$ is the weight of the synapse connecting neuron i

to j and y_i, y_j are the corresponding neurons' activations. In this model, if the outputs of the pre- and post-synaptic neurons stayed constant, the weight $w_{j,i}$ would approach the expected value $\mathbb{E}[f(y_i(t))g(y_j(t))]$ over time. But there is in fact a positive feedback-loop, since an increase in the synaptic weight leads to an increase in the post-synaptic activation, which in turn leads to a further increase of the synaptic weight, and so on. This could potentially lead to unstable runaway dynamics, where the weights all either converge to 0 or diverge to $\pm\infty$. Stable variations of this rule exist for that reason, such as the popular BCM rule [159], which includes a term that adjusts for the neuron's mean activity. Instead, I use the homeostatic intrinsic plasticity of the post-synaptic neuron to the same end, i.e. to maintain a fixed distribution of the neuron's output.

Note: Beyond linear Hebbian learning

The most commonly used synaptic learning rules are (bi-)linear Hebbian learning rules, where the rate of change of the weights is a product of a linear function of pre- and post-synaptic activation. But non-linear dependencies on the pre- and post-synaptic activations are of course conceivable, as well! Such non-linear Hebbian learning rules make it possible to further decouple the effects of synaptic and intrinsic plasticity, e.g. choosing an activation function purely to maximize information transmission in combination with a learning rule to realize principal or independent component analysis. Non-linear Hebbian learning rules therefore open countless more opportunities for synaptic learning rules that could be studied in this framework.

So what happens when we drive an assembly of multiple neurons with a multi-variate stationary input process, and let the synaptic connections and intrinsic parameters evolve according to these synaptic and intrinsic plasticity rules? As we show analytically in contribution 6, without any stabilizing homeostatic plasticity, the weights do in fact diverge. But under the effect of intrinsic plasticity, the weights follow a gradient field and settle in stable fixed-points. For multi-variate Gaussian inputs, these fixed-points correspond exactly to the principal component directions. More accurately, the linear Hebbian synaptic learning rule finds a projection of the multi-dimensional input space onto the one-dimensional membrane potential, for which the expected activation of the post-synaptic neuron is *maximized*, whereas intrinsic plasticity *normalizes* its expected activation.

As the input distribution (and thus its principal component directions) changes, the weight vector re-aligns itself and thus counteracts this transformation, thereby realizing a special form of homeostasis.

5.5.2 Independent component analysis

In the previous section, the combination of intrinsic and synaptic plasticity lead to the discovery of principal components, because for that choice of input distribution and activation function, the variance of the input had the largest effect on the expected output of the neuron. But what if we were to choose a different input distribution than Gaussian, where higher-order moments carry important information?

Due to the nonlinear activation function, the expected value of the neuron's output also depends on the higher-order moments of its membrane potential (see the appendix of contribution 6). For example, with a monomial activation function $f(x) = x^n$, the neuron's mean output measures the n-th moment of the input distribution. With $n = 2$, such a neuron would be sensitive to the *variance* of the input, and to the *curtosis* for $n = 3$. In general,

it depends on the Taylor expansion of the activation function, how much each moment of the input distribution influences the neuron's mean output. If we choose a different activation function or input distribution, the neuron can therefore discover other subspaces that maximize higher-order moments of the input distribution, instead. This can be used to disentangle signals that are *uncorrelated*, but not *independent*, because they do share higher-order correlations. In analogy to principal component analysis, this procedure is therefore called *independent component analysis* (ICA). [9] Our combination of intrinsic and synaptic plasticity mechanisms produces either principal or independent component analysis or a mixture thereof, depending on the input distributions and/or activation function!

[9] The original work by [160] introduced independent component analysis using a similar, neuro-inspired motivation with the activation function $f(x) = x^3$ to maximize curtosis.

5.6 Applying the information bottleneck to neural assemblies

We can generalize these ideas from individual neurons to neural assemblies: By selecting and scaling the inputs into the neurons, synaptic plasticity determines how the neurons' inputs are related to each other, whereas intrinsic plasticity independently controls the marginal distribution of each individual neuron's outputs.

This raises an interesting question: how much control over its *joint output distribution* could an assembly of neurons theoretically exert, if the only free parameters are each neuron's nonlinearity and the incoming synaptic connections? Can an assembly of neurons map an arbitrary multi-variate input distribution onto an arbitrary multi-variate output distribution? The general answer is no[10], but to make this more precise, we have to disentangle the effects of synaptic and intrinsic plasticity. In contribution 6, we do this by introducing a concept from probability theory called *copula* ([161], see also the note below).

[10] Consider e.g. that a uni-variate input signal cannot be transformed into multiple independent output signals.

Note: Copulas describe the coupling of random variables

For an assembly of N neurons with individual inputs $X_i \sim P_{X_i}$ with joint distribution P_X and outputs $Y_i \sim P_{Y_i}$, we define the intermediate random variables $U_i = F_{X_i}(X_i)$, called the *ranks* or *quantiles* of X_i, each of which is marginally uniformly distributed. The joint distributions of these rank-variables is the copula

$$C(u) := F_X(F_{X_1}^{-1}(u_1), F_{X_2}^{-1}(u_2), \dots, F_{X_N}^{-1}(u_N)),$$

a probability distribution in the N-dimensional unit cube, that captures how the random variables X_i are related — regardless of their marginal distributions! A common application of this rank-transformation in statistics is when random variables need to be compared across different scales. In that case, the correlation between the ranks of the variables can be used, which is just the correlation of the copula. The copula has many more interesting theoretical properties, e.g. [162], but most importantly for us, it is invariant under any invertible univariate transformations of the individual random variables X_i. Therefore, the copula C of the assembly's multi-variate input and its multi-variate output distribution coincide and we can just talk of the copula of the assembly. The copula is only a property of the synaptic connections and the input distribution, and unaffected by the neuron's nonlinearity.

Using the copula C, we can factorize the stationary joint probability distributions of the membrane potentials X and activations $Y = f^*(X)$ as follows:

$$F_X(x) = C(F_{X_1}(x_1), F_{X_2}(x_2), \dots, F_{X_N}(x_N))$$
$$F_Y(x) = C(F_{Y_1}(y_1), F_{Y_2}(y_2), \dots, F_{Y_N}(y_N))$$

The last factorization specifies the population's joint output distribution in terms of the desired individual marginal distributions F_{Y_i} of *each neuron's* output, which can be enforced by intrinsic plasticity, and the copula function C, which captures the co-dependency *between the neurons'* activity. The copula is invariant under element-wise invertible transformations, and therefore only depends on the synaptic connections — not the activation function. In other words: No matter what activation functions we choose, we can only modify the *marginal distributions* of each neuron's output —but not the assembly's copula— through intrinsic plasticity! Synaptic plasticity, on the other hand, can shape how different signals are combined by the individual neurons, and thus it *can* influence the assembly's copula, but not the activation function.

In general, C can be arbitrarily complex, and there is little hope that it can be fully controlled by setting the synaptic weights alone. [11] For copulas that are parameterized by more than one parameter per synapse, for example, synaptic plasticity alone is obviously insufficient to fully control the copula. But e.g. for jointly Gaussian inputs, the distribution is fully parameterized by the covariance matrix (and the mean), which *can* be shaped arbitrarily by an appropriate choice of synaptic weights (and intrinsic plasticity).

[11] Of course, it may still be possible to control the copula through synaptic weights by using multiple layers of neurons, as is done e.g. for so-called *normalizing flows* [163].

If we apply the information bottleneck argument now to an entire assembly of multiple neurons, the neurons should jointly maximize information transmission. This can be achieved if the neurons' outputs are i.i.d. with a marginal maximum entropy distribution.

It therefore seems reasonable from an information bottleneck perspective, that the individual neurons should encode different independent (or principal) components. One way to ensure this is mutual decorrelation of the neurons within an assembly by lateral inhibition in order to enforce the learning of different weights. We demonstrate that this leads to the unsupervised clustering of the MNIST handwritten digits dataset by the extraction of independent components. For a second control dataset composed of random image patches no such independent components should exist, and indeed the same setup leads to the discovery of the dominant principal components instead. The same idea, sometimes called *blind source separation*, can be generalized to other types of input signals as well, in particular to separate heavy-tailed[12] source signals (see also example 3).

[12] I use this term to refer to random variables with larger higher-order moments than a normally distributed variables with equal variance.

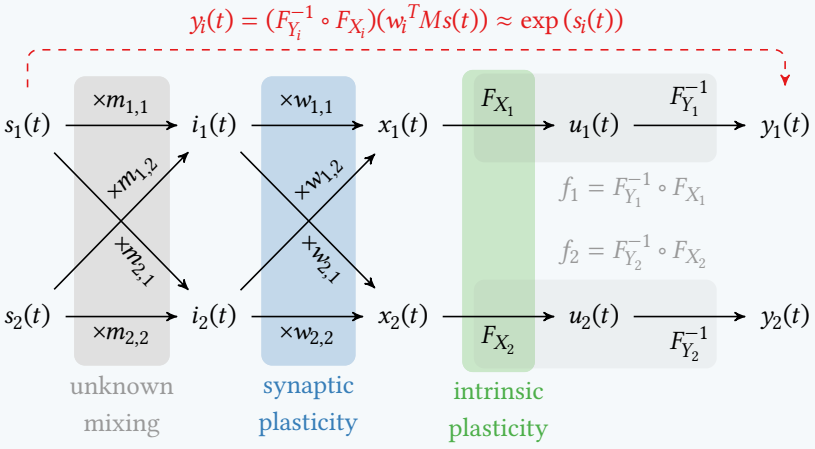
There is plenty of biological evidence to prove that such a decorrelation of signals by PCA/ICA also occurs in nature. For example, this can be observed on a very low level of the mammalian visual system (see [7, chapter 5] for a great summary of this topic). There, colors are sensed by receptors tuned to different (but overlapping) spectra of visual light, but the signals that are transmitted by ganglion cells appear to be linear combinations of these "raw signals": instead of a 'red', a 'green' and a 'blue' channel, the spike-trains transmitted over the optic nerve seem to represent a 'blue-(red+green)' difference, a 'red+green' sum and a 'red-green' difference channel! This representation decorrelates the highly correlated responses of the individual color channels, and results in a more information theoretically and metabolically efficient code. Just as in our hypothetical example, this requires an appropriate rotation of the synaptic weight vectors and an appropriate scaling of the neuron's nonlinear activation function, although that may be genetically predetermined, rather than learned, in this specific case. A similar observation can be made in the olfactory bulb of zebrafish larvae [164], where appropriate lateral inhibition decorrelates the neurons' responses.

5.7 *Plasticity is information processing*

In the field of machine learning, we think of neural networks in terms of a training phase, where the network is optimized, and an inference phase, where the trained network is used to process information. As I tried to show in this rather long chapter, this perspective completely misses the important role that plasticity mechanisms play in information processing, in

Example 3: Blind-source separation with synaptic- & intrinsic plasticity

Suppose we want to "de-mix" two independent source signals $s_1(t)$ and $s_2(t)$ from two different mixtures $i_1(t)$ and $i_2(t)$. This could be two independent sound sources that reach our two ears with different attenuation, or it could merely be two correlated outputs of neurons in a previous layer — in either case, the mixture coefficients $m_{i,j}$ are not explicitly known. These two input signals are then transmitted through synaptic connections with weights $w_{i,j}$ to the two neurons, where they are integrated into the membrane potential $x_1(t)$ or $x_2(t)$, respectively (we ignore the temporal filtering of the membrane potential here). Each neuron i then applies its activation function f_i to produce the output $y_i(t)$. We'd like each neuron to reproduce a nonlinear function $\exp(s_i(t))$ of just *one* of the input signals. This is called *blind-source separation* or independent component analysis.



The interaction of synaptic plasticity and intrinsic plasticity can solve this problem in an unsupervised manner. To ensure that $f_i(x) = (F_{Y_i}^{-1} \circ F_{X_i})(x) = \exp(x)$, we assume X_i has a probability distribution from the same family as S_i, and set $P_{Y_i} = \exp_{\#}(P_{S_i})$. The greedy mechanism of synaptic plasticity then finds a weight matrix W that inverts the unknown mixing matrix M, while intrinsic plasticity stabilizes this process and ensures that the outputs have the desired distributions F_{Y_i}. The intermediate variable $U(t) = (u_1(t)\ u_2(t))^T$ is marginally uniform, and its joint distribution is the *copula* of X_1 and X_2 or Y_1 and Y_2, respectively.

particular if we consider the dynamics of online learning that has to happen in real-time, such as homeostatic intrinsic plasticity and synaptic plasticity. Since the interaction of intrinsic and synaptic plasticity can help not just to stabilize the behavior of neurons and networks, but also to extract, compress and track relevant information like principal or independent components in high-dimensional signals, I'd consider them to be information processing mechanisms in and of themselves. While the high-level discussion above was focused on continuous linear-nonlinear neuron models, we will apply these concepts to spiking neurons, as well, in chapter 6.

References for chapter 5:

2. W. R. Ashby, *Design for a Brain: The Origin of Adaptive Behaviour (2nd Ed. Rev.)*. Chapman & Hall, 1960. DOI: 10.1037/11592-000 (cit. on pp. vii, 2, 45).

6. S. (O. N. Laughlin University Of C, *Principles of Neural Design*. 2017, ISBN: 978-0-262-53468-0 (cit. on pp. vii, 17, 34, 51, 60, 66).

7. J. V. Stone, *Principles of Neural Information Theory: Computational Neuroscience and Metabolic Efficiency*. Sebtel Press, 2018, 214 pp., ISBN: 978-0-9933679-2-2 (cit. on pp. vii, 33, 34, 46, 55, 60, 62, 66, 71, 98).

12. **J. Leugering** and G. Pipa, "A Unifying Framework of Synaptic and Intrinsic Plasticity in Neural Populations," *Neural Computation*, vol. 30, no. 4, pp. 945–986, 17, 2018, ISSN: 0899-7667. DOI: 10.1162/neco_a_01057 (cit. on pp. viii, 46).

66. N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in *2015 IEEE Information Theory Workshop (ITW)*, 2015. DOI: 10.1109/ITW.2015.7133169 (cit. on pp. 12, 46).

113. J. B. Anderson and R. Johnnesson, *Understanding Information Transmission*. John Wiley & Sons, 17, 2006, 323 pp., ISBN: 978-0-471-71119-3. Google Books: GD5GY4XyPXIC (cit. on pp. 32, 33, 46, 48, 60, 61).

121. E. C. Smith and M. S. Lewicki, "Efficient auditory coding," *Nature*, vol. 439, no. 7079, pp. 978–982, 7079 2006, ISSN: 1476-4687. DOI: 10.1038/nature04485 (cit. on pp. 34, 51).

143. S. Legg and M. Hutter. (25, 2007). "A Collection of Definitions of Intelligence." arXiv: 0706.3639 [cs], [Online]. Available: http://arxiv.org/abs/0706.3639 (visited on 08/20/2020) (cit. on p. 45).

144. N. Tishby, F. C. Pereira, and W. Bialek. (24, 2000). "The information bottleneck method." arXiv: physics/0004057, [Online]. Available: http://arxiv.org/abs/physics/0004057 (visited on 08/22/2020) (cit. on p. 46).

145. S. Lange and M. Riedmiller, "Deep auto-encoder neural networks in reinforcement learning," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, 2010. DOI: 10.1109/IJCNN.2010.5596468 (cit. on p. 46).

146. L. Buesing and W. Maass, "A Spiking Neuron as Information Bottleneck," *Neural Computation*, vol. 22, no. 8, pp. 1961–1992, 25, 2010, ISSN: 0899-7667. DOI: 10.1162/neco.2010.08-09-1084 (cit. on p. 46).

147. T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. Wiley-Interscience, 2006, 748 pp., ISBN: 978-0-471-24195-9 (cit. on pp. 46, 48).

148. T. Goblick, "Theoretical limitations on the transmission of data from analog sources," *IEEE Transactions on Information Theory*, vol. 11, no. 4, pp. 558–567, 1965, ISSN: 1557-9654. DOI: 10.1109/TIT.1965.1053821 (cit. on p. 48).

149. E. T. Jaynes, "Information Theory and Statistical Mechanics," *Physical Review*, vol. 106, no. 4, pp. 620–630, 15, 1957. DOI: 10.1103/PhysRev.106.620 (cit. on p. 49).

150. G. Peyré and M. Cuturi, "Computational Optimal Transport: With Applications to Data Science," *Foundations and Trends® in Machine Learning*, vol. 11, no. 5-6, pp. 355–607, 11, 2019, ISSN: 1935-8237, 1935-8245. DOI: 10.1561/2200000073 (cit. on pp. 49, 50).

151. A. Painsky and N. Tishby, "Gaussian lower bound for the information bottleneck limit," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 7908–7936, 1, 2017, ISSN: 1532-4435 (cit. on p. 50).

152. B. Øksendal, "Stochastic Differential Equations," in *Stochastic Differential Equations*, Springer Berlin Heidelberg, 2003, pp. 65–84, ISBN: 978-3-540-04758-2 978-3-642-14394-6. DOI: 10.1007/978-3-642-14394-6_5 (cit. on pp. 51, 52).

153. A. Azarfar, N. Calcini, C. Huang, F. Zeldenrust, and T. Celikel, "Neural coding: A single neuron's perspective," *Neuroscience & Biobehavioral Reviews*, vol. 94, pp. 238–247, 1, 2018, ISSN: 0149-7634. DOI: 10.1016/j.neubiorev.2018.09.007 (cit. on p. 51).

154. K. P. Körding, C. Kayser, W. Einhäuser, and P. König, "How Are Complex Cell Properties Adapted to the Statistics of Natural Stimuli?" *Journal of Neurophysiology*, vol. 91, no. 1, pp. 206–212, 1, 2004, ISSN: 0022-3077. DOI: 10.1152/jn.00149.2003 (cit. on p. 51).

155. N. Brenner, W. Bialek, and R. de Ruyter van Steveninck, "Adaptive Rescaling Maximizes Information Transmission," *Neuron*, vol. 26, no. 3, pp. 695–702, 1, 2000, ISSN: 0896-6273. DOI: 10.1016/S0896-6273(00)81205-2 (cit. on p. 51).

156. A. L. Fairhall, G. D. Lewen, W. Bialek, and R. R. de Ruyter van Steveninck, "Efficiency and ambiguity in an adaptive neural code," *Nature*, vol. 412, no. 6849, pp. 787–792, 6849 2001, ISSN: 1476-4687. DOI: 10.1038/35090500. pmid: 11518957 (cit. on p. 51).

157. L. Kostal, P. Lansky, and J.-P. Rospars, "Efficient Olfactory Coding in the Pheromone Receptor Neuron of a Moth," *PLOS Computational Biology*, vol. 4, no. 4, e1000053, 25, 2008, ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1000053 (cit. on p. 51).

158. A. Holtmaat and K. Svoboda, "Experience-dependent structural synaptic plasticity in the mammalian brain," *Nature Reviews Neuroscience*, vol. 10, no. 9, pp. 647–658, 9 2009, ISSN: 1471-0048. DOI: 10.1038/nrn2699 (cit. on p. 52).

159. E. L. Bienenstock, L. N. Cooper, and P. W. Munro, "Theory for the development of neuron selectivity: Orientation specificity and binocular interaction in visual cortex," *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience*, vol. 2, no. 1, pp. 32–48, 1982, ISSN: 0270-6474. pmid: 7054394 (cit. on p. 53).

160. A. Hyvärinen and E. Oja, "Independent component analysis: Algorithms and applications," *Neural Networks*, vol. 13, no. 4, pp. 411–430, 1, 2000, ISSN: 0893-6080. DOI: 10.1016/S0893-6080(00)00026-5 (cit. on p. 54).

161. P. Jaworski, F. Durante, W. K. Härdle, and T. Rychlik, eds., *Copula Theory and Its Applications: Proceedings of the Workshop Held in Warsaw, 25-26 September 2009*. Springer Berlin Heidelberg, 2010, vol. 198, ISBN: 978-3-642-12464-8 978-3-642-12465-5. DOI: 10.1007/978-3-642-12465-5 (cit. on p. 54).

162. J. Ma and Z. Sun, "Mutual Information Is Copula Entropy," *Tsinghua Science & Technology*, vol. 16, no. 1, pp. 51–54, 1, 2011, ISSN: 1007-0214. DOI: 10.1016/S1007-0214(11)70008-6 (cit. on p. 54).

163. D. J. Rezende and S. Mohamed. (14, 2016). "Variational Inference with Normalizing Flows." arXiv: 1505.05770 [cs, stat], [Online]. Available: http://arxiv.org/abs/1505.05770 (visited on 08/29/2020) (cit. on p. 55).

164. A. A. Wanner and R. W. Friedrich, "Whitening of odor representations by the wiring diagram of the olfactory bulb," *Nature Neuroscience*, vol. 23, no. 3, pp. 433–442, 3 2020, ISSN: 1546-1726. DOI: 10.1038/s41593-019-0576-z (cit. on p. 55).

Yeah, but your scientists were so preoccupied over whether or not they could that they didn't stop to think if they should.

Dr. Ian Malcolm in *Jurassic Park*

6 *Rate-coding with spiking neurons*

The neuron models we looked at so far all use real-valued, continuous output signals as a proxy of the neuron's current firing rate. But — electric gap-junctions aside — biological neurons in the human brain communicate via chemical synapses that actually have to generate individual *spikes* to communicate. This mode of communication has been known for almost two centuries [165], but to this day it plays only a minor role in machine learning models of neural networks. Why is it, that biological neurons send spikes, rather than continuous signals? And why haven't we seen more applications of spiking neural networks in machine learning?

There are two fundamentally different schools of thought on this issue. The *rate-coding* paradigm assumes, that the only relevant information conveyed by a spike-train is the time-varying rate at which the spikes are generated, while the *spike-(time-)coding* paradigm treats each individual spike as a symbol, the timings of which convey individual pieces of information. Unsurprisingly, how and how well a neuron can represent its input and whether rate-coding is a viable model for that, has been one of the oldest research questions in theoretical neuroscience [166], and has been revisited many times (see e.g. [167, 168]). Surprisingly, there is still a lack of conclusive biological evidence one way or another and the distinction between the two is not always clearly cut [169], so this apparently simple question hasn't been settled even after decades of intense debate. To better understand what rate-coding entails, I will therefore stick to an entirely theoretical view of rate-coding in the spirit of [166], and analyze its capacity to encode and transmit information for two of the most common neuron models. In chapter 7, I will then try to account for more recent biological evidence, which will lead us to spike-time coding, or rather *event-coding*.

6.1 *Why do (only) biological neurons spike?*

For proponents of rate-coding, the additional complexity of spike-based communication can be understood as a biological "implementation detail" of sorts: Pulse-based communication offers a noise-robust and energy-efficient means to approximately convey a continuous, real-valued signal (the *firing rate*) under metabolic constraints over what is essentially a binary channel (the neuron's axon with its chemical synapses). The continuous signal is then represented by the *rate* or *density* of the pulses per unit time-interval. This form of encoding is simple to implement and very reliable, which is why variations of this scheme are also used in digital electronics to transmit inherently analog signals (e.g. audio signals or servo-motor controls) over a digital connection. [1]

[1] In electronics, there is in fact a corresponding pulse-based encoding scheme for each of the spike-based communication paradigms (see also [3, chapter 3]): *pulse-density modulation* (PDM) and its variants, which go by the names *pulse-frequency modulation* (PFM) and *pulse-code modulation* (PCM) correspond to rate coding. *Pulse-position modulation* (PPM) corresponds to spike-time coding, which we will discuss in chapter 7. Asynchronous $\Delta\Sigma$ modulation is directly related to integrate-and-fire neurons and time-encoding-machines [107].

Spike-based communication also shares another benefit with digital electronics: While analog signals strongly attenuate as they propagate along the neural membrane (see also chapter 4), binary spikes can be detected over long distances and regenerated to their full amplitude by error-correcting mechanisms. In cortical neurons, this happens at distinct locations along the axon called *Ranvier nodes* [6], [2] which are separated by highly myelinated (i.e. electrically insulated) stretches of the axon. This insulation not only reduces leakage but also greatly increases conduction velocity. The resulting *saltatory propagation* of action potentials allows for an extremely fast, reliable and energy efficient communication over long distances without degradation, which is critical for coordinating motor activity in limbs far away from cortex [7, 139] and plays a crucial role in the consolidation of memory [139].

In short, from the rate-coding perspective, spike-based communication is a very useful adjustment to bio-physical constraints. But as long as a spiking neuron's firing rate can be well approximated by a nonlinear function of its input, the precise mode of communication makes little conceptual difference; the overarching framework is still function approximation by linear-nonlinear neurons, and spiking neurons are merely a hardware-efficient implementation (or approximation) thereof. In fact, as we shall see below, there is a direct correspondence between continuous linear-nonlinear neurons and simple spiking neuron models that makes it trivial to convert back-and-forth (as long as we are only concerned with mean firing rates).

[2] Again, there is a rich analogy to man-made electrical communication systems: while the strong attenuation of analog signals along wires originally made the transmission of analog signals (such as the telephone) over large distances difficult, the digital nature of the telegraph allowed for a regeneration of the signal at periodically spaced relay stations, much alike Ranvier nodes, and thus enabled fast long range communication. A nice account of this development can be found in [113].

6.2 Encoding continuous signals into rate-coded spike-trains

For the rate-coding paradigm, any benefit of spiking neurons has to come not from increased computational power, but rather from an increased *metabolic efficiency*, i.e. the amount of information transmitted per Joule of energy spent. To discuss the capabilities and limitations of rate-coding, it's therefore important to understand how and how well time-varying signals can be encoded into a series of spikes, in the first place.

Since a neuron will have to be able to decode the relevant signal again from the spike train, typical requirements for spike-based codes are as follows: The current firing rate of a neuron is only a function of its recent input [3] ; a neuron's response is time-equivariant, i.e. a temporal shift in the input results in a corresponding shift in the output; each spike is represented by a brief stereotypical pulse with identical mass[4]; and despite the *non-linear encoding*, a *linear decoder*, i.e. a filter, must be sufficient to decode the continuous rate from the spiking signal[5]. This linear 'decodability' imposes a hard constraint on the sort of spike-patterns that can be used to convey information, but there is biological evidence to support this simplifying assumption[6]. Since the spike-encoding mechanism might induce filtering effects that are irreversible, as we have seen in chapter 4, we will be satisfied if we can encode a signal $s(t)$ into a spike-train $z(t)$ and are then able to recover $(\kappa * z)(t) = f((\psi * s)(t))$ for some kernel ψ and some invertible function f from the spike-train by the linear decoder with kernel κ. These constraints still leave room for many different mechanisms to encode a continuous signal into a discrete sequence of spikes, but I will only consider three particularly interesting types of rate-based encodings.

Since each spike requires energy to generate and transmit, we can compare these different approaches by the number of spikes they tend to generate, and how well we can decode the underlying continuous signal from the spike-train.

[3] Adaptation effects as in chapter 5 are sometimes included as well, but on a much slower timescale.

[4] Typically, a Dirac $\delta(t)$ is chosen for continuous time and a Kronecker δ_t/rectangular pulse for discrete-time models.

[5] This requirement is the *first principle* of the *neural engineering framework* [5]; see also chapter 2.

[6] Pairwise correlations between spikes (which can be assessed with a linear filter) carry most of the information content in individual neurons' spike-trains [170].

6.2.1 Periodic sampling and digital transmission

To put the encoding capabilities of spiking neurons into perspective, let's compare them to a well-known reference: digital encoding schemes. In the signal processing domain,

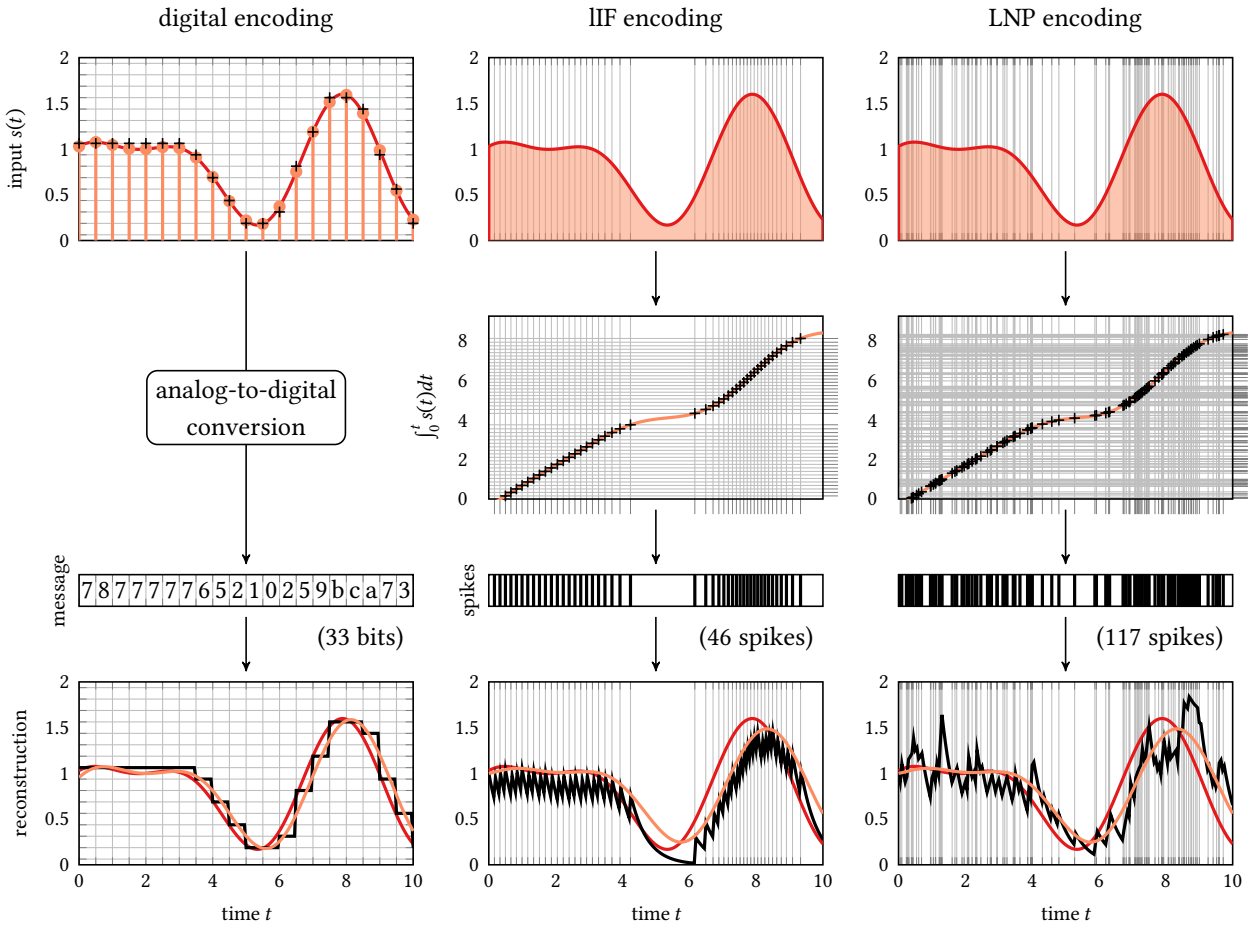


Figure 6.1. The same input signal *(top row, solid red)* is encoded by a 4-bit digital code *(left column)*, an LIF neuron *(middle column)* and an LNP neuron *(right column)* over a 10s interval. The LIF neuron produces spikes at (almost) even increments of the signal's integral, while the LNP neuron fires at uniformly distributed signal integrals *(second row)*. The digital code results in a parsimonious representation that here requires only 33 active bits. The LIF neuron requires 46 spikes for a slightly worse reconstruction *(bottom row, black line)* of the filtered input signal *(orange line)*, while the LNP neuron gives a bad approximation even for 108 spikes.

continuous real-valued signals are typically measured at periodic time-intervals, which produces a discrete sequence of real-valued samples. According to the Nyquist-Shannon sampling theorem [113], any bandwidth-limited signal can be losslessly represented this way if the *sampling rate* is sufficiently high. Each real-valued sample can then be approximated by n-bit binary number, and the active bits can be transmitted as brief pulses via n parallel wires. If we scaled each of these pulses (or bits) by the corresponding power of two, summed and filtered them appropriately, [7] we'd recover the continuous signal. While this binary encoding would hence (with a little stretch of the imagination) satisfy our requirements of a rate-based encoding, it is of course not actually a viable model of neural spike-based communication. But it does provide a theoretically optimal reference implementation for the pulse-based transmission of bandwidth-limited continuous signals, against which we can measure the performance of other, more plausible rate-coding schemes.

6.2.2 *Rate-coding with (leaky-)integrate-and-fire neurons*

Biological neurons generate spikes through a cascade of opening and closing ion channels, which modulate in- and outgoing ion currents that in turn drive the neural membrane potential. This complex biological mechanism is described by the famed Hodgkin-Huxley model [50], but much simpler models suffice if we are only interested in capturing the encoding of continuous signals into spikes. The simplest of these is the ubiquitous *integrate-and-fire* model [3, 171, 172], which operates by integrating the input signal up to a critical threshold, where it resets and fires a spike.

[7] The optimal kernel, a sinc function scaled and stretched according to the sampling frequency, is acausal, but an approximate solution can also be obtained with a causal kernel. See also chapter 4.

This can be theoretically "justified" as follows: For a continuous signal $s(t)$, the firing rate of the neuron should encode the signal in a way that can be linearly decoded by filtering (see also chapters 2 and 5). Therefore, the firing rate ought to be proportional to $s(t)$, i.e. the average number of spikes in an interval $[t_1, t_2]$ should be proportional to $\int_{t_1}^{t_2} s(t)\mathrm{d}t$.[8] For the neuron to fire exactly one more spike, the average time-interval between the previous spike at time t_1 and the new spike at time t_2 should therefore be $c\int_{t_1}^{t_2} s(t)\mathrm{d}t = 1 \Leftrightarrow S(t_2) = S(t_1) + 1/c$, where $S(t) = \int_{-\infty}^{t} s(\tau)\mathrm{d}\tau$ and c is a constant of proportionality. In other words, the neuron should fire a spike whenever $S(t)$, the integral of the signal $s(t)$, increases by more than the threshold $\theta = 1/c$ over its value at the previous spike. Of course, this is exactly the mechanism implemented by the standard integrate-and-fire neuron, which integrates its input $s(t)$ up to the critical threshold θ, where the opening of voltage-gated channels resets the neuron back to its resting potential (here chosen as 0 for the sake of simplicity), and the process begins anew. Such an encoding is also called a *send-on-delta* scheme, as a spike is emitted whenever there is a significant change ("delta") in the (integral) of the signal.

This allows us to say something about the timing of the spikes in relation to the signal: If we assume a positive input signal s, then $S(t)$ is a monotonically increasing (and hence invertible) function. Whenever $S(t) = S(t_i) + 1/c$, a new spike t_{i+1} is generated, therefore the spike times t_k satisfy $S(t_k) = k\theta$ and thus $t_k = S^{-1}(k\theta)$. This spike-train is linearly decodable by construction, and we can easily verify that this also meets our other requirements of a rate-code, since a delay in s leads to an equal delay in S and thus in t_k.

The ideal integral operator in this construction would require the membrane potential to remain constant in the absence of external inputs. Not only do inevitable leakage currents make this implausible for both biology and neuromorphic hardware, but it also has the undesirable theoretical implication, that the output of the neuron depends on a potentially infinitely long history of inputs. [9] To remedy this, the integral operator, whose impulse response is a step-function, is often replaced by a low-pass filter with an exponentially decaying impulse response. This results in the more biologically plausible and very popular *leaky integrate-and-fire* (LIF) neuron [3]. If the time-scale of the exponential filter is very short, it approaches a Dirac-δ kernel and the neuron acts like a coincidence detector of nearly simultaneous spikes, to which we will return in chapter 7. For a very long time-scale, on the other hand, the exponential filter approaches a step-function and the model converges to the pure integrate-and-fire neuron. As theoretical considerations and biological observations show (see e.g. chapter 3 of [7]), the optimal trade off between these two extremes in terms of metabolic efficiency (i.e. how many bits are transmitted per Joule spent) seems to be achieved when the filter's time-scale roughly equals the expected inter-spike interval. Other filters than the exponential could be used as well (see also the node below), but will not be further discussed here.

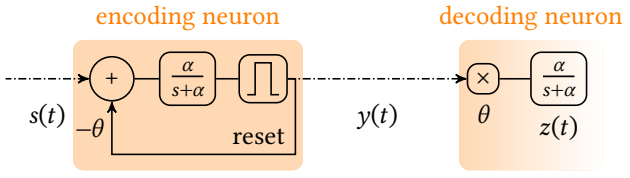


Figure 6.2. Two simple leaky integrate-and-fire neurons as used in figure 6.1, one of which receives a time-varying continuous signal $s(t)$ as its input and *encodes* it into a spike-train $y(t)$. The second neuron *decodes* the spike-train into piece-wise continuous membrane potential trace $z(t)$. Both use the same exponential kernel $\kappa_\alpha = \frac{\alpha}{s+\alpha}$ with rate α.

Let's consider the example shown in figure 6.2 of a pair of LIF neurons with the dendritic filter $\kappa_\alpha(s) = \frac{\alpha}{s+\alpha}$. Suppose we'd like to recover the filtered signal $(\kappa_\alpha * s)(t)$ from the decoding neuron's membrane potential $z(t)$. We know that the residual $(\kappa_\alpha * s)(t) - (\kappa_\alpha * y)(t)$ between the filtered input signal and the filtered spike-train is bounded between 0 and θ, because whenever the error exceeds that bound, another spike is generated, resetting the error back to 0. Under a few simplifying assumptions (see appendix B.1), this residual has a mean value of

[8] The average number of spikes in a small interval $[t - \Delta t/2, t + \Delta t/2]$ of length Δt should be approximately proportional to $s(t) \cdot \Delta t$. Partitioning the interval $[t_1, t_2]$ into strips of width Δt and calculating the Riemann sum for $\Delta t \to 0$ gives this result.

[9] This would e.g. violate the fading-memory assumption of reservoir computing [30].

$\approx \theta/2$ and a root-mean-squared error (RMSE) of $\approx \alpha\theta/\sqrt{12}$. By reducing θ (and thus increasing the firing rate) or α (and thus increasing the filter's time-constant), we can therefore reduce the error bound arbitrarily and get uniform convergence $\lim_{\theta\to 0} z = s * \kappa_\alpha$.

For a constant signal $s(t) = c \geq \theta$, the neuron's firing rate thus scales almost [10] linearly with c, whereas the expected RMSE remains constant across almost the entire input range of the neuron. No spikes at all are generated for $c \leq \theta$, so the mean firing rate as a function of the constant input c approximates the rectified-linear unit (ReLU) $f(c - \theta/2) \approx \max(0, c - \theta/2)$. See also appendix B.1 for a derivation.

[10] This approximation of the mean and RMSE fail when the mean input to the neuron goes below θ.

As figure 6.1 shows, the LIF neuron is capable of reliably encoding a time-varying signal into a single pulse-train. Because of its simplicity, a very similar mechanism is also commonly used in signal processing under the name ΣΔ or ΔΣ modulator [108] or just *integrate-and-fire* sampling [173] to convert continuous signals into pulse-trains[11].

[11] The main difference is, that a ΔΣ-modulator encodes both positive and negative changes of the signal into the rising and falling edges of a binary pulse-width-modulated signal, whereas the LIF mechanism encodes only positive changes into spikes and relies on the passive leakage for decreasing the signal.

Note: Filter-and-fire neurons

The (leaky) integrator represents only one specific type of filter that a neural dendrite could implement (see chapter 4). By substituting in various other kernels, the integrate-and-fire model can thus be generalized to a very interesting class of *filter-and-fire* models [3]. The leaky-integrate-and-fire (LIF) model with an exponential kernel shown here is a particularly popular example, since it can be motivated from biological first principles and can be implemented very efficiently by a single first-order low-pass filter. But also second-order filters like the α-kernel, a convolution of two exponential kernels, are used to model the combined effect of filtering by the chemical synapse as well as the neuron's membrane potential [3]. Such a higher-order filter could help remove the high-frequency noise otherwise introduced by the discontinuous jumps that result from filtering a spike-train with a first-order filter. Naturally, the choice of kernel has strong implications for the behavior of the neuron, and all the arguments from chapter 4 apply to spiking neurons just as well.

6.2.3 Stochastic encoding

A rather different approach to rate-coding utilizes stochasticity. A linear-nonlinear-Poisson (LNP) spiking neuron [174] fires spikes according to an inhomogeneous Poisson process [175] that uses the input signal $s(t)$ as its time-varying *rate*. The resulting spike-times are stochastic, but the expected number of spikes per time-interval $[t_1, t_2]$ is proportional to the integral $\int_{t_1}^{t_2} s(t)\mathrm{d}t$, just like for the LIF neuron above. But in contrast to the LIF neuron, the spike-times in that interval are independently and identically distributed with cumulative distribution function S, i.e. $t_k \sim S^{-1}(u_k)$ where u_k is a uniform random variable. [12] In fact, this property allows us to elucidate the key difference between the deterministic integrate-and-fire and the stochastic LNP model: While the spike-times of the IF neuron contain no information besides the signal s (i.e. the spike-times t_k are deterministic given s), the spike-times of the LNP neuron also encode noise (i.e. the spike-times t_k are randomly distributed with a cumulative distribution function proportional to S). A different way of looking at the same phenomenon is to view the LNP neuron as equivalent to an IF neuron with exponentially distributed random threshold[13], or subject to a corresponding distribution of noise on the membrane potential. All other things being equal, the LNP neuron is therefore likely to achieve a much worse signal-to-noise ratio, as we shall also see below.

[12] Recall that in (L)IF neurons, the latent variable u_k would instead be (almost) regularly spaced at $u_k = k/N$ for N spikes.

[13] Since spikes are generated at uniformly distributed levels of S, the increments from one spike to the next are exponentially distributed.

Let's look at an example of LNP neurons in figure 6.1. Just like in the case of the integrate-and-fire neuron above, the signal can be linearly decoded by filtering the spike-train with an

exponential kernel. One can show (see appendix B.2), that exponentially filtering the spike-train provides an unbiased estimate of the signal with an RMSE that approaches $\sqrt{\alpha c/(2\lambda)}$. Like for the LIF neuron, the expected firing rate response of the LNP neuron to constant input is therefore given by a rectified-linear function, but unlike the LIF neuron, the RMSE actually grows with c.

Key benefits of the stochastic approach and the main reason for its popularity are the possibility to incorporate noise and its conceptual simplicity, which allows it to be trivially extended to assemblies of multiple neurons. Deterministic LIF neurons, for contrast, can show phase-locking and other specific dynamics, or may fail to fire all together if the input is sub-threshold, whereas LNP neurons respond linearly across the entire input range. The stochasticity of the LNP neuron could thus actually *enhance* information transmission in some specific cases, but it generally comes at considerable expense in others, as we'll quantify in section 6.4.

6.3 Rate-coding neurons are linear-nonlinear neurons

We already saw above that the mean firing rate of spiking neurons can be modeled as a function of their (constant) input signals. Conveniently, this function takes the rectified-linear form for both LIF and LNP neurons, one of the most popular choices of activation function in current deep neural network architectures.[14] By making the firing rates sufficiently large, [15] an arbitrary accuracy (i.e. an arbitrarily low RMSE) can be achieved. If we also choose the dendritic filters' time-constants appropriately, a trained deep neural network can be trivially translated into a spiking neural network simply by replacing each continuous neuron with one accordingly configured spiking neuron — *et voilà*, we have a trained deep spiking neural network! A direct conversion of this sort has been shown to work even for very large, state-of-the-art network models [176, 177]. The same idea can be applied to recurrent networks and reservoir computers, as well (see also [5]). As the success of this one-to-one conversion confirms, rate-coding really is merely a different implementation of the continuous function approximation paradigm discussed in chapter 2.

[14] This is of course hardly a coincidence, since the rectified-linear activation function was in fact modeled after its (spiking) biological inspiration.

[15] For the (L)IF neuron, this can be achieved by lowering the threshold θ, for the LNP neuron by raising the gain λ.

6.4 How good is rate-coding for transmitting information?

If we adopt the rate-coding perspective, the single purpose of spike-based communication is to transmit analog signals reliably under biological constraints. But which mechanism works best? How does it fare in comparison to a purely analog implementation? And how does rate-coding with spikes compare to conventional *digital* sampling schemes that are used to simulate deep neural networks? Are rate-coding spiking neural networks a viable machine-learning alternative to conventional deep neural networks?

I'll attempt to (partially) answer these questions here, starting with the example shown in figure 6.1. There, we saw three fundamentally different coding schemes that all represent a continuous signal by discrete series of binary pulses, but they yield rather different results. Using the digital "neuron" as a reference, I'll compare, how efficiently the LIF and the LNP neuron can encode information into a spike-train. Figure 6.3 shows the RMSE of both neuron models when encoding the constant signal $s(t) = c = 0.5$. As we systematically vary the neuron's firing rate by varying the LIF neuron's threshold θ and the LNP neuron's gain λ, we can observe a consistently and substantially lower error for the LIF neuron than for the LNP neuron. Besides the fact that both the LIF and LNP neuron follow the rate-coding approach and use the same exponential kernel with rate $\alpha = 40Hz$ for decoding, the LIF neuron makes much better use of the precise timing of each spike, and encodes the continuous input signal more effectively into a spike train.

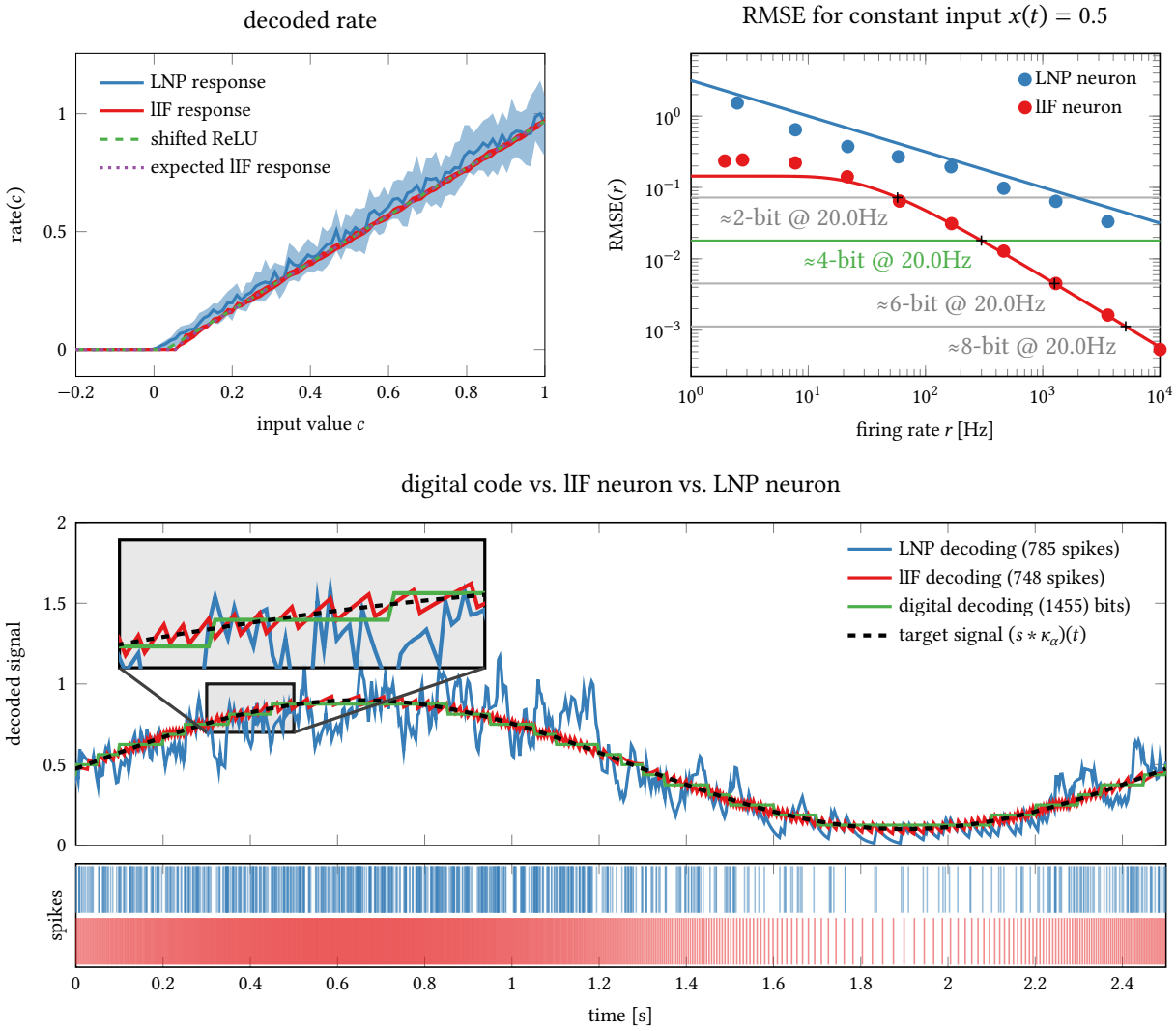


Figure 6.3. *Top left:* The normalized mean firing rate as a function of the constant input c matches the (shifted) ReLU activation function $\max(0, x - b)$ closely, where $b = 0$ for the LNP neuron and $b = \theta/2$ for the LIF neuron. The interquartile range of the LNP neuron's firing rates (in blue), estimated over 50 trials, grows with c and is much larger than for an LIF neuron (in red). *Top right:* The RMSE as a function of the firing rate for constant input $c = 0.5$ shows, that the LIF neuron achieves much lower errors at equal firing rates than the LNP neuron. *Bottom:* Spike-trains and decoded signals in response to a slow varying sine-wave (black dashed line). For reference, a 4-bit signal sampled at 20Hz is included (green).

We can quantify this more accurately with the help of information theory (see appendix B.3 for a derivation), which shows that encoding a (constant) signal with the same specified accuracy ϵ (defined by the differential entropy of the residual) will require a quadratically larger number of spikes for the LNP neuron than for the LIF neuron as we increase ϵ!

But how do these simple rate-coding spiking neural network models compare against an analog implementation or their digital counterparts from deep learning? Not too well, unfortunately: As figure 6.3 already shows, even for firing rates as high as 10,000Hz, the LIF neuron in this setup only reaches an accuracy less than that of a 10bit signal sampled at 20Hz. At a more reasonable 250Hz firing rate, the rate-coding LIF neuron barely matches the accuracy of a 4-bit signal sampled at a rate of 20Hz, resulting in an effective information content of less than a quarter bit per spike. The LNP neuron fares much worse than that.

As shown above, this disparity is in large part due to the poor scaling of the accuracy with the number of spikes, which is linear in the firing rate for the LIF neuron and only scales with the square root of the firing rate for the LNP neuron. For contrast, the accuracy of a digital code grows *exponentially* with the bit-depth of the signal! This is bad news for rate-coding with spiking neurons, since most deep learning models currently make use of the much more accurate half-precision (16-bit) or quarter-precision (8-bit) floats or integers, with only few networks quantized to precisions as low as 4-bit or below [73]. In the rate-coding context, SNNs also offer no qualitative benefits in terms of raw computational power; to the

contrary, they are merely used to approximate the behavior of DNNs and can therefore not be expected to surpass their performance. [16] Any benefit of rate-coding in biology, machine learning or neuromorphic hardware must therefore come from a more efficient *physical implementation*, rather than an information theoretical argument.

[16] It is however possible that spiking neural networks have intrinsic biases that prove beneficial, e.g. if they had a regularizing effect of sorts, but I'm not aware of any proof of that.

6.5 Optimal rate-coding under metabolic constraints

So far, we focused on the encoding accuracy of spiking neurons (measured by the RMSE) for some given constant signal. In the language of information theory, this corresponds to the problem of *channel coding*. But if we want to fully assess the neuron's ability to transmit information, we need to consider *source-coding*, as well. In section 5.2, we approached this from the information bottleneck perspective: to make the neuron's output as informative as possible, we opted to maximize its entropy (measured in $^{\text{bits}}/_{\text{second}}$) while respecting the metabolic constraints imposed on the neuron. This leads to the most powerful neuron that the energy budget allows, but in a realistic setting that may not be the best solution overall. Here, we will instead try to make the best possible use of the energy by optimizing the *metabolic efficiency* $\varepsilon := {}^{h}/_{\text{cost}}$ instead, i.e. the entropy of the neuron's output *in relation to the required power* cost (measured in $^{\text{bits}}/_{\text{Joule}}$). This alternative approach leads to a less powerful but more parsimonious neuron, which is a worthwhile trade-off if energy, rather than the number of neurons, is the most critically limited resource. In general, there is a four-way trade-off for spiking neurons between raw performance on the one hand, and firing rates ('paying with spikes'), the complexity and size of neurons, synapses and circuits ('paying with hardware'), and metabolic costs ('paying with power') on the other [7]. If we take into account, that the human brain demands almost 20% of the body's entire energy budget [178] and almost 80% of that energy is directly spent on the firing of spikes [178, 179], and scales linearly with the mean firing rate [178, 180], the enormous evolutionary benefit of increased metabolic efficiency becomes obvious. And indeed, the general tendency for biological neurons seems to be optimization of metabolic efficiency [6, 7]. The idea of such an *economy of impulses* goes back at least to [181] and was formalized by [182] for populations of neurons. We'll briefly look at what this implies for the single rate-coding neuron.

One important observation is, that under rather mild assumptions (see example 4) the metabolic efficiency is maximized for a unique optimal firing rate μ^* that strikes a good balance between the inevitable *static* power consumption of the neuron, which occurs regardless of the neuron's firing rate, and the *dynamic* power consumption due to the generation of spikes. A brief back-of-the-envelope calculation in example 4 using parameter estimates from real neurons puts this optimal rate at a surprisingly low mean firing rate of around $1.41^{\text{spikes}}/_{s}$ — much lower than what cortical neurons are capable of, but very well in line with the distribution of firing rates observed *in vivo*. [17]

[17] The mean firing rate of human cortical neurons is estimated at 1.15Hz with a range from $0.5 - 2.0$Hz; see [179] and references within.

Which ever firing rate distribution offers the best trade-off, a neuron could achieve and maintain it with an appropriate activation function and homeostatic plasticity, as I argued in chapter 5. For the LNP neuron, this can be implemented by *explicitly* making the instantaneous firing rate a nonlinear function of the membrane potential. This is more complicated for deterministic (L)IF neurons, because their effective nonlinearity is only *implicitly* defined by the neural dynamics, but the effective firing rate function can be influenced by various indirect means such as filtering (see chapter 4) or nonlinear dependencies between the membrane potential and the input [183] or between the threshold and the membrane potential [184]. In either case, a homeostatic mechanism like in chapter 5 could help achieve and maintain this optimal encoding in the face of changing or unpredictable input distributions [18].

[18] Work to apply these ideas to the design of neuromorphic hardware is currently ongoing, but not yet completed at the time of writing this thesis.

Example 4: Low firing rates optimize metabolic efficiency

Let's assume that the static power consumption of the neuron is a constant $\text{cost}_{\text{static}}$, whereas the dynamic power consumption scales linearly with the number of spikes $\mu = \mathbb{E}[Y]$ at a fixed cost of e_{spike} per spike. The total power consumption of the neuron is then $\mu e_{\text{spike}} + \text{cost}_{\text{static}}$. If we further assume that the firing-rate Y of the neuron is only subject to additive noise and exponentially distributed, which, as we saw already in chapter 5, maximizes the neuron's capacity for a certain mean firing rate μ, the neuron's capacity to transmit information is $1 + \log(c\mu) + c$, where c is a unit-dependent scaling factor to make $c\mu$ unit-free. Under these (mild) assumptions, the efficiency is a function of the mean firing rate

$$\epsilon(\mu) \propto \frac{1 + \log(c\mu) + c}{\mu e_{\text{spike}} + \text{cost}_{\text{static}}},$$

which has a unique maximum for

$$\mu = \gamma / cW(\gamma), \qquad \text{where } \gamma := \text{cost}_{\text{static}}/e_{\text{spike}} \text{ and } W \text{ is Lambert's function.}$$

If we take biological measurements from rodents [178] to estimate static and dynamic power ($e_{\text{spike}} \approx 7.12 \times 10^8 \text{ATP}$, $\text{cost}_{\text{static}} = 3.42 \times 10^8 \text{ATP/s}$, both measured in terms of the consumed number of adenosine triphosphate (ATP) molecules, $c = 1\text{s}$), we get a factor of $\gamma \approx 0.48$ and thus an optimal firing rate $\mu^* \approx 1.41\text{Hz}$.

6.6 *Rate-coding spiking neural networks and machine learning*

Spiking neuron models are certainly worth studying for both biologists and neuromorphic hardware designers, but since they are more difficult to simulate in software and rate-coding offers no apparent qualitative computational benefits over deep learning, they are currently of little relevance for machine learning. Hence, the viability of SNNs hinges on how well they can encode continuous signals into spikes and back, and how efficiently this can be physically implemented. Rather than representational power of the neuron, *metabolic efficiency* might therefore be the decisive factor behind the evolutionary success of spiking neural networks.

In the following chapter 7, we'll pursue this idea to it natural conclusion by looking at even more parsimonious, event-based alternatives to the rate-coding paradigm itself.

References for chapter 6:

3. W. Maass and C. M. Bishop, *Pulsed Neural Networks*. MIT Press, 2001, 414 pp., ISBN: 978-0-262-63221-8. Google Books: jEug7sJXP2MC (cit. on pp. vii, 32, 59, 61–63, 73).

5. C. Eliasmith and C. H. Anderson, *Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems*. MIT press, 2004 (cit. on pp. vii, 15, 60, 64).

6. S. (O. N. Laughlin University Of C, *Principles of Neural Design*. 2017, ISBN: 978-0-262-53468-0 (cit. on pp. vii, 17, 34, 51, 60, 66).

7. J. V. Stone, *Principles of Neural Information Theory: Computational Neuroscience and Metabolic Efficiency*. Sebtel Press, 2018, 214 pp., ISBN: 978-0-9933679-2-2 (cit. on pp. vii, 33, 34, 46, 55, 60, 62, 66, 71, 98).

30. H. Jaeger, W. Maass, and J. Principe, "Special issue on echo state networks and liquid state machines.," 2007 (cit. on pp. 2, 14, 62).

50. A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of Physiology*, vol. 117, no. 4, pp. 500–544, 28, 1952, ISSN: 0022-3751. pmid: 12991237 (cit. on pp. 6, 61).

73. S. Han, H. Mao, and W. J. Dally. (2015). "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." arXiv: 1510.00149 (cit. on pp. 13, 65).

107. A. A. Lazar and L. T. Tóth, "Time encoding and perfect recovery of bandlimited signals," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 6, 25, 2003 (cit. on pp. 27, 59, 73).

108. R. Gray, "Oversampled Sigma-Delta Modulation," *IEEE Transactions on Communications*, vol. 35, no. 5, pp. 481–489, 1987, ISSN: 1558-0857. DOI: 10.1109/TCOM.1987.1096814 (cit. on pp. 27, 63).

113. J. B. Anderson and R. Johnnesson, *Understanding Information Transmission*. John Wiley & Sons, 17, 2006, 323 pp., ISBN: 978-0-471-71119-3. Google Books: GD5GY4XyPXIC (cit. on pp. 32, 33, 46, 48, 60, 61).

139. R. D. Fields, "A new mechanism of nervous system plasticity: Activity-dependent myelination," *Nature reviews. Neuroscience*, vol. 16, no. 12, pp. 756–767, 2015, ISSN: 1471-003X. DOI: 10.1038/nrn4023. pmid: 26585800 (cit. on pp. 41, 60).

165. E. H. D. B. Reymond, *Vorläufiger Abriß einer Untersuchung über den sogenannten Froschstrom und über die elektromotorischen Fische*. 1843, 30 pp. Google Books: goNar3mBwJkC (cit. on p. 59).

166. D. M. MacKay and W. S. McCulloch, "The limiting information capacity of a neuronal link," *The bulletin of mathematical biophysics*, vol. 14, no. 2, pp. 127–135, 1, 1952, ISSN: 1522-9602. DOI: 10.1007/BF02477711 (cit. on pp. 59, 71, 78).

167. S. Panzeri, R. S. Petersen, S. R. Schultz, M. Lebedev, and M. E. Diamond, "The Role of Spike Timing in the Coding of Stimulus Location in Rat Somatosensory Cortex," *Neuron*, vol. 29, no. 3, pp. 769–777, 1, 2001, ISSN: 0896-6273. DOI: 10.1016/S0896-6273(01)00251-3 (cit. on p. 59).

168. F. Zeldenrust, S. de Knecht, W. J. Wadman, S. Denève, and B. Gutkin, "Estimating the Information Extracted by a Single Spiking Neuron from a Continuous Input Time Series," *Frontiers in Computational Neuroscience*, vol. 11, 2017, ISSN: 1662-5188. DOI: 10.3389/fncom.2017.00049 (cit. on p. 59).

169. R. Brette, "Philosophy of the Spike: Rate-Based vs. Spike-Based Theories of the Brain," *Frontiers in Systems Neuroscience*, vol. 9, 2015, ISSN: 1662-5137. DOI: 10.3389/fnsys.2015.00151 (cit. on p. 59).

170. A. Dettner, S. Münzberg, and T. Tchumatchenko, "Temporal pairwise spike correlations fully capture single-neuron information," *Nature Communications*, vol. 7, no. 1, pp. 1–11, 1 15, 2016, ISSN: 2041-1723. DOI: 10.1038/ncomms13805 (cit. on p. 60).

171. A. N. Burkitt, "A Review of the Integrate-and-fire Neuron Model: I. Homogeneous Synaptic Input," *Biological Cybernetics*, vol. 95, no. 1, pp. 1–19, 1, 2006, ISSN: 1432-0770. DOI: 10.1007/s00422-006-0068-6 (cit. on p. 61).

172. A. N. Burkitt, "A review of the integrate-and-fire neuron model: II. Inhomogeneous synaptic input and network properties," *Biological Cybernetics*, vol. 95, no. 2, pp. 97–112, 1, 2006, ISSN: 1432-0770. DOI: 10.1007/s00422-006-0082-8 (cit. on p. 61).

173. H. G. Feichtinger, J. C. Príncipe, J. L. Romero, A. Singh Alvarado, and G. A. Velasco, "Approximate reconstruction of bandlimited functions for the integrate and fire sampler," *Advances in Computational Mathematics*, vol. 36, no. 1, pp. 67–78, 2012, ISSN: 1019-7168, 1572-9044. DOI: 10.1007/s10444-011-9180-9 (cit. on p. 63).

174. S. Ostojic and N. Brunel, "From Spiking Neuron Models to Linear-Nonlinear Models," *PLOS Computational Biology*, vol. 7, no. 1, e1001056, 20, 2011, ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1001056 (cit. on pp. 63, 93).

175. V. Capasso and D. Bakstein, *An Introduction to Continuous-Time Stochastic Processes*. Birkhäuser Boston, 2012, ISBN: 978-0-8176-8345-0 978-0-8176-8346-7. DOI: 10.1007/978-0-8176-8346-7 (cit. on p. 63).

176. B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, and S.-C. Liu, "Conversion of Continuous-Valued Deep Networks to Efficient Event-Driven Networks for Image Classification," *Frontiers in Neuroscience*, vol. 11, 2017, ISSN: 1662-453X. DOI: 10.3389/fnins.2017.00682 (cit. on p. 64).

177. A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going Deeper in Spiking Neural Networks: VGG and Residual Architectures," *Frontiers in Neuroscience*, vol. 13, 2019, ISSN: 1662-453X. DOI: 10.3389/fnins.2019.00095 (cit. on p. 64).

178. D. Attwell and S. B. Laughlin, "An Energy Budget for Signaling in the Grey Matter of the Brain:" *Journal of Cerebral Blood Flow & Metabolism*, 31, 2016. DOI: 10.1097/00004647-200110000-00001 (cit. on pp. 66, 67).

179. Y. Yu, P. Herman, D. L. Rothman, D. Agarwal, and F. Hyder, "Evaluating the gray and white matter energy budgets of human brain function," *Journal of Cerebral Blood Flow and Metabolism: Official Journal of the International Society of Cerebral Blood Flow and Metabolism*, vol. 38, no. 8, pp. 1339–1353, 2018, ISSN: 1559-7016. DOI: 10.1177/0271678X17708691. pmid: 28589753 (cit. on p. 66).

180. G. Yi and W. M. Grill, "Average firing rate rather than temporal pattern determines metabolic cost of activity in thalamocortical relay neurons," *Scientific Reports*, vol. 9, no. 1, p. 6940, 1 6, 2019, ISSN: 2045-2322. DOI: 10.1038/s41598-019-43460-8 (cit. on p. 66).

181. H. B. Barlow, "Trigger features, adaptation and economy of impulses," in *Information Processing in the Nervous System: Proceedings of a Symposium Held at the State University of New York at Buffalo 21st–24th October, 1968*, K. N. Leibovic, ed., Springer Berlin Heidelberg, 1969, pp. 209–230, ISBN: 978-3-642-87086-6. DOI: 10.1007/978-3-642-87086-6$_1$1 (cit. on p. 66).

182. W. B. Levy and R. A. Baxter, "Energy Efficient Neural Codes," *Neural Computation*, vol. 8, no. 3, pp. 531–543, 1, 1996, ISSN: 0899-7667. DOI: 10.1162/neco.1996.8.3.531 (cit. on p. 66).

183. M. Stemmler and C. Koch, "How voltage-dependent conductances can adapt to maximize the information encoded by neuronal firing rate," *Nature Neuroscience*, vol. 2, no. 6, pp. 521–527, 1999, ISSN: 1097-6256. DOI: 10.1038/9173 (cit. on p. 66).

184. N. Fourcaud-Trocmé, D. Hansel, C. van Vreeswijk, and N. Brunel, "How Spike Generation Mechanisms Determine the Neuronal Response to Fluctuating Inputs," *Journal of Neuroscience*, vol. 23, no. 37, pp. 11 628–11 640, 17, 2003, ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.23-37-11628.2003. pmid: 14684865 (cit. on p. 66).

185. K. A. Boahen, "Communicating Neuronal Ensembles between Neuromorphic Chips," in *Neuromorphic Systems Engineering: Neural Networks in Silicon*, T. S. Lande, ed., Springer US, 1998, pp. 229–259, ISBN: 978-0-585-28001-1. DOI: 10.1007/978-0-585-28001-1_11 (cit. on p. 71).

Time represents itself.

— Boahen in [185]

What we have found is that at least a comparable information capacity is potentially available in respect of impulse timing [...] and it seems unlikely that the nervous system functions in such a way as to utilize none of this.

— MacKay and McCulloch in [166]

7 *Spike-timing and event based computation*

In chapter 6 we came to the sobering conclusion that in order to match the accuracy of linear-nonlinear neurons, rate-coding neurons would have to fire at excessive firing rates. The high metabolic cost associated with the generation of spikes makes such "naive" rate-coding theoretically unappealing for biological and artificial neurons. We addressed this issue by directly optimizing the firing rate distribution for *metabolic efficiency* instead, which resulted in a much more efficient encoding with firing rates as low as 1Hz. While such low rates are in line with experimental observations, they pose a theoretical conundrum for rate-coding: We assumed that a rate-code should be linearly decodable by the dendritic filter of another neuron i.e. on a timescale on the order of tens of milliseconds — but a rate of 1Hz is orders of magnitude *too slow* to be smoothed by a dendritic filter! One way to make sense of this is to consider that each neuron receives input from not just a single other neuron, but from thousands, and it could be their combined input that rate-codes a signal. But this explanation raises another question: If all of these neurons encode *the same* signal through their firing rates, this redundancy increases the energy cost again — destroying any gains in metabolic efficiency due to the individually lower firing rates. Wouldn't a single neuron at a higher firing rate be more efficient? [1] On the other hand, if all of these neurons encode different signals, each of these signals is represented by a too low firing rate to be interpretable in a rate-coding setting, and we're back to square one.

In this chapter, I'd therefore like to ask a more fundamental question: Is rate-coding already the best we can do, or is there a more metabolically efficient code for spike-based communication?

An early study [166] applied tools from information theory to establish limits for how much information a single spike could, in principle, transmit in a realistic setting. Their estimate put this capacity at an astonishing 9 bits per spike — orders of magnitude larger than what we saw for rate-coding neurons in chapter 6! [2] Early empirical studies have yielded more conservative estimates for the amount of information *actually* transmitted per spike *in vivo* (around 1 bit per spike, see [186] for a (dated) review), but some more recent experiments do come surprisingly close to this theoretical limit, demonstrating transmission of around $5.6 - 7$ bits per spike [7, 187]! In order to achieve such a high information content per spike, merely counting the average rate of spikes per second is not sufficient — the timing must be taken into account, as well. In the following, we'll therefore look at *spike-timing*-based codes.

[1] In chapter 6 we saw that the power consumption scales almost linearly with firing rates with a per-neuron overhead due to static power consumption. Such a redundant population code would therefore almost certainly be less efficient.

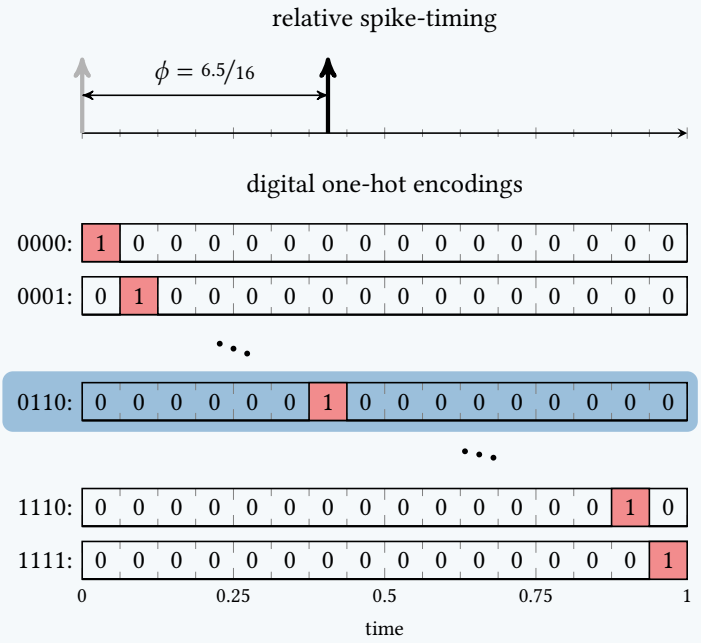
[2] For example, we saw an LIF neuron firing around 300 spikes per second to encode a signal with an accuracy and speed roughly comparable to a 4 bit signal sampled at 20Hz, i.e. at a rate of only about 0.27bits per spike.

7.1 Spike-time coding

Depending on the definition, there might be infinitely many different spike-trains with the same (time-varying) firing rate, so there is a lot of potentially relevant information encoded in the precise spike-times in addition to the mere firing rate (see also example 5). But how could a biological neuron extract such timing information?

Example 5: Phase coding is pulse-position modulation (PPM)

To understand how spike timing allows a single spike to convey multiple bits of information, we can make a simple analogy to a digital serial code that uses a "one-hot"-encoding with 2^n bits to convey n bits of information. In the case of $n = 4$, this means that within a time-interval discretized into $2^4 = 16$ time-steps, exactly one bit is active:



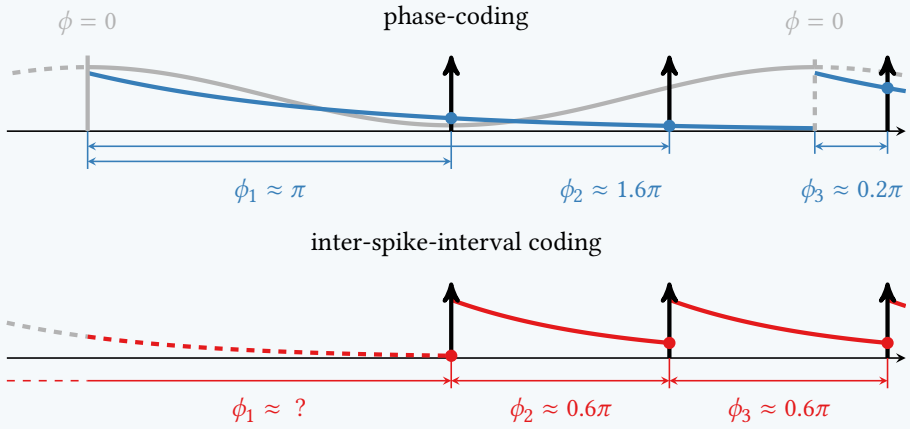
By looking at *which* bit is active within a given interval, we can thus recover 4 bits of information encoded by the corresponding sequence (here, 0110). A single *active* bit (or spike) in this example therefore transmits 4 bits of information! Note however, that this requires a reference signal to indicate the start of the interval and a precise clock signal against which the relative timing of the spike can be measured. Therefore, phase coding is just the time-continuous counterpart of *pulse-position-modulation*, a popular digital encoding scheme in signal processing !

Let's look at two popular models of spike-time coding that can be implemented by integrate-and-fire neurons (see also example 6):

The first makes use of the timing-delay between a neuron's spikes and a separate reference signal to encode a real-valued number. The reference signal could either be some particular event such as the onset of a stimulus or a saccade (then also called *"time-to-first-spike" coding* [188]), or it could be some change in the electrical potential of the neuron. If the reference signal is periodic, such as theta oscillations in hippocampus [189], the spike-train thus encodes a periodically sampled signal. Each sample is then encoded by the relative *phase-delay* between the spike and the reference signal, which is why this code is also referred

Example 6: Implementing phase and ISI-codes

Information can be encoded by the timing of spikes relative to some reference signal. Here, the reference signal could be either a spike from another neuron or a specific phase of a background oscillation (phase coding), or it could be the previous spike from the neuron itself (ISI coding).



In either case, the relative timing of each spike provides one real valued sample. Using a simple mechanism like dendritic filtering with a slowly decaying exponential filter that is triggered by the reference signal, the relative timing delay of a spike to the reference is a (nonlinear) function of the remaining trace at spike-time.

to as *phase-coding*. For a comparatively slow reference signal, such a code results in a very sparse spike-train with, in the extreme case, only a single spike per cycle that encodes a multi-bit measurement! Just like in the linear-nonlinear neuron model, synaptic weights can be used to change the timing of spikes, and the real-valued samples encoded by the relative spike times can be used for universal computation (see e.g. chapter 2 of [3]). Such a code also offers some computational advantages over the rate-codes from chapter 6: Consider, for example, an assembly of multiple neurons, each of which represents a different feature of an input signal and is decorrelated from its neighbors by inhibitory lateral connections, e.g. via some inhibitory inter-neuron. The neuron with the strongest response then fires first, and thus disables the others before they can fire. Such an assembly would compute the `maximum` operation over multiple signals with only a single spike fired! This mechanism is also extremely robust to changes in scale: If all neurons' responses were scaled down, the time of each spike might be delayed, but the order would be preserved, leading to the same result. This is consistent with the observations that cortical neurons typically fire at much lower frequencies and respond faster than the rate-coding perspective would require (see chapter 6), and that most information about a novel stimulus can often be decoded from just the first few spikes [188].

The second kind of spike-timing code is *inter-spike-interval* (ISI) coding [3], and it assumes that information is conveyed by the precise time-interval between two consecutive spikes. Just like with periodic sampling, a bandwidth-limited analog signal can in principle be encoded, transmitted and decoded without loss [107] using such an encoding[3]. A biological neuron might implement this e.g. by an exponentially decaying trace of the membrane potential or some chemical, which is reset to a fixed value by each spike. The value of the trace at any point in time then (nonlinearly) encodes the time since the previous spike, and

[3] However, optimal decoding of such a signal might require a more complex mechanism than the linear decoder we required in chapter 6.

affects the likelihood of the neuron to fire once it receives a new input. One might argue that this is quite similar to what the leaky-integrate-and-fire model already does in the limit of extremely low firing rates, and the key to explain its superior performance over the LNP model (see chapter 6).

Both of these spike-timing based encoding-mechanisms integrate nicely with the theoretical framework of *spike-timing dependent synaptic plasticity* (STDP) [190], which not only consider the simultaneous firing rates of the pre- and postsynaptic neuron (like rate-based Hebbian rules), but also the relative timing of their spikes.

7.2 Event coding

Both of the spike-timing-based approaches above rely on the same basic assumption as rate coding, that a spike-train ultimately encodes some time-varying, continuous signal or samples thereof. This provides a nice mathematical framework in which to compare various encoding schemes, and it integrates perfectly with the prevalent machine-learning perspective of neural networks as continuous function approximators (see chapter 2). But in some situations this might be an overly convoluted way of explaining a much simpler phenomenon: the neuron just fires a spike, whenever it receives a "relevant" stimulus! I'll call this simpler view the *event-coding* paradigm, in which each spike (or volley of spikes, as we shall see later) represents the occurrence of a specific *event*, rather than a real-valued sample of some continuous signal. Conceptually, this is much closer to an *interrupt-* or *event-driven* rather than a sampling-based mode of communication, which is also used in digital electronics to convey sparse signals with little latency.

A biological example of such an extremely parsimonious event-based code can be found in the fast sensory pathway of the weakly electric fish [191], where spherical neurons produce only a single individual spike in response to a prolonged stimulation. That neurons would use such an event-based style of communication also seems reasonable from an evolutionary perspective, since many of the biological mechanism used by spiking neurons predate the sort of nervous system required to even generate or interpret a rate-, ISI- or phase-coded signal. Consider for example bacteria that can form biofilm and coordinate through chemically communicated electric action potentials [192], or the rudimentary Ca^{2+} signalling mechanism already present in choanoflagellates [193] that predate animal life. Here, an event (e.g. high concentration of a chemical) triggers a specific response (e.g. release of chemicals, formation of a biofilm) — a simple mechanism that might be a precursor to spiking neurons and is best understood from this event-driven perspective. In another evolutionary stage, *nerve-nets* [91], action potentials often induce some synchronized behavior throughout the body of an animal. For example, pacemaker neurons of the jellyfish generate periodic action potentials that trigger a nerve-net of motor neurons to drive synchronized contraction of swimming muscles [91]. The output of these pacemaker neurons can be best understood as a form of event-coding. The spike-based transmission of information might therefore originate in some form of event-coded sensory or motor signals. By triggering these event detectors at a stimulus-dependent rate, it is conceivable that rate-based codes could also have emerged from such a simpler event-based code.

7.3 Detecting events in spike-trains

But what exactly constitutes an *event*, and how can such an event be detected? While this may be clear for a sensory neuron, we need to specify what *event* means in the context of cortical neurons that only receive spiking input from other neurons. I will give two different definitions, *fixed spatio-temporal patterns*, and *ordered* but variable *sequences* of such patterns.

Most of these ideas apply to continuous signals as well, but in both cases I will focus only on spike events.

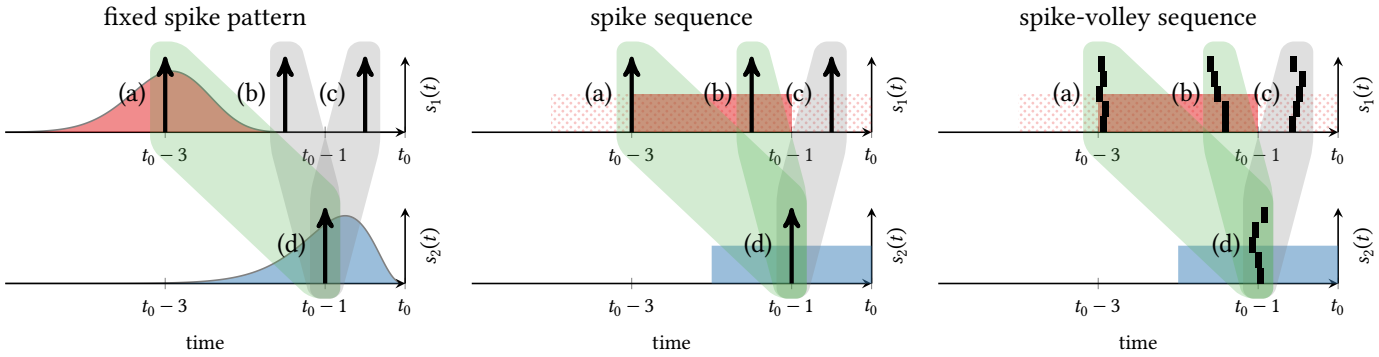


Figure 7.1. **Left:** A pattern-detector for one spike each from two incoming spike-trains. The kernel for each input filter is shown in red and blue, respectively. Only the spike pair $(a) \to (d)$ is an accepted pattern. **Middle:** A detector for a sequence of one spike in input 1 that precedes another spike in input 2 by some bounded time-interval. Both patterns $(a) \to (d)$ and $(b) \to (d)$ are accepted. **Right:** The same detector for a sequence of spike-volleys, rather than individual spikes.

7.3.1 *Fixed spatio-temporal patterns*

The most obvious definition of an *event* would be a stereotypical signal that last for a brief time-interval and always follows the same time-course. For spiking signals, that would be a fixed *pattern* of spikes over time, distributed across one or more neurons. We can define this as follows:

A *spike pattern event* $P = \{\tau_{i,j} : i \in \{1, \dots, n\}, j \in J_i\}$ that occurs at time τ produces the n-dimensional signal $P_i(t) = \sum_{j \in J_i} \delta(t - \tau - \tau_{i,j})$, where $\tau_{i,j}$ are called the *spike-times* of the pattern. J_i is the index set of spikes belonging to neuron i in this pattern.

To detect such a pattern, a neuron could make use of a dendritic filter that implements the matched filter of the pattern, i.e. a kernel $\kappa(t) = P(T - t)$ for some T (see chapter 4). To allow for small jitter in the timing of the spikes, we can additionally smoothen the dendritic filter by convolution with some other kernel g.[4] Additional spikes *not* belonging to the pattern at all can also affect the filter response, so an appropriately high threshold needs to be chosen to allow a reliable distinction between pattern and noise. The detection of such fixed patterns therefore reduces to dendritic filtering and thresholding, which e.g. the Gamma-neuron from chapter 4 with appropriate number of filter taps can approximate very well.

[4] This corresponds to a Janossy distance metric over spike-trains [194] and could be similarly derived from optimal transport theory [195].

While the ability to detect such stereotypical patterns is certainly useful, this kind of event-detector suffers from two draw-backs. First, biological parameters determine the time-scale of dendritic integration, which limits the length of patterns that can be detected by this mechanism and may prove to be too short for many interesting patterns. Second and more importantly, this definition of a spike pattern is extremely rigid, as it prescribes the exact time of each spike in the pattern with little room for variability. This is fine for detecting relatively short patterns, such as volleys of (nearly) synchronous spikes or rapid successions of spikes produced by a "hard-wired" cell assembly or motive. The high timing precision of some cortical neurons [196, 197] shows that such well-timed spike patterns are certainly possible to generate. But for longer lasting patterns, in particular if they are driven by external inputs that can vary in length, we'd expect some variability in the timing of the individual spikes.

7.3.2 *Ordered (but variable) sequences of spikes and spike patterns*

Instead of prescribing the actual spike times as above, we might only be interested in the order in which certain spikes arrive. For example, a spike from neuron A followed by a spike from neuron B would constitute a noteworthy event regardless of the precise timing

of either spike (as long as B fires within some time interval after A). This would be a very parsimonious code, as well, but it relies on the ability of an individual spike to reliably encode the occurrence of some event. Conversely, a single erroneously generated spike could trigger such a neuron and lead to a false detection.

To improve the reliability of such an event-based code, we can extend this concept to *ordered sequences of spike patterns*, e.g. sufficiently large *volleys* of spikes from some assembly of neurons rather than individual spikes. In the notation from above, such a spike volley corresponds to a pattern with a single spike per neuron (i.e. $J_i = \{1\}$), all of which are set to occur at roughly the same time (i.e. $t_{i,j} = 0$). This can be easily detected by a dendritic filter, e.g. the fast exponential filter of the leaky-integrate-and-fire neuron model or a brief rectangular filter would work. Since multiple synchronous spikes are required to elicit a response, such an encoding would be extremely robust to noise, while having very low latency and requiring only relatively few spikes to signal a noteworthy event.

7.4 Active dendritic sequence processing

The ability to detect sequences of spike-volleys as discussed above would be a very useful property for spiking neurons to have, but it requires more sophisticated biological mechanisms than just passive dendritic integration. Over the last century, a lot of research has gone into studying the electrical properties of cortical neurons, but only in the last two decades has the vastly improved technology in neuroimaging and electrophysiology allowed a deeper investigation of one rather fundamental property of cortical neurons: Neural dendrites are not just the passive cables we considered in chapter 4, but they can produce localized long-lasting depolarization, i.e. dendritic NMDA or calcium spikes or, as I will collectively call them, plateau potentials [198]. These actively generated effects have been shown to play an important role in cortical UP-states [199], the generation of spikes and bursts, synaptic plasticity and learning, non-linear dendritic computation, and more [198]. In an apparent case of convergent evolution, physiologically different but functionally similar mechanisms exist not just in cortical pyramidal neurons, but also in other cell types such as Purkinje cells [200].

London and Häusser [201] suggested that such localized processes would endow a single dendrite with countless *functional subunits*, which might be the key to understanding a neuron's computational capabilities. Given the importance and ubiquity of this phenomenon, it is surprising how few models in theoretical neuroscience and machine learning currently incorporate active dendritic processes or offer an explanation of their contribution. This may in part be due to inconclusive and sometimes even contradictory biological evidence[5], which makes it difficult for theoreticians to decide, which phenomena are fundamental, and which are merely "quirks of nature". Two very interesting models by Hawkins and Ahmad [204] and Brea, Gaál, Urbanczik, and Senn [205] include such active dendritic processes to explain the emergence of a predictive UP-state in the neuron's somatic membrane potential, which allows the individual neuron to predict (and learn) "state-transitions" and a network of such neurons to (learn to) detect long-lasting sequences of input! We build on these ideas and derive a much more general model of this process, which we call *active dendritic sequences processing*.

[5] For example, it has been reported that individual spike-inputs at apical dendrites might have no measurable impact on somatic membrane potentials due to strong signal attenuation [202] just as it has been reported that this effect might be completely compensated for by synaptic scaling [141]. Strongly nonlinear interactions between localized dendritic membrane potentials have been demonstrated [142], but other results show a nearly linear integration through the entire dendrite [203]. The list goes on.

The proposed model and its derivation from basic biological principles and observations is detailed in contribution 7. It uses passive dendritic filtering (i.e. the integration of EPSPs) in individual, electrically isolated dendritic compartments to detect volleys of coincident spikes originating from some populations of neurons. Upon detecting such an event, an *active* process generates a localized, long-lasting depolarization (a *plateau potential*), which enables a nearby dendrite segment to detect the next volley event in the sequence. If the second

Note: Stochastic population codes with event-based spiking neurons

Just like a single spike, a spike volley is a well-timed event that can be used for an event-based code. But unlike the single spike, it conveys an additional piece of information besides the timing of the event: the magnitude of the volley, i.e. the number of participating spikes. This could carry either of two different interpretations: it could signify the magnitude of the corresponding event, just like an earth-quake event has a time and magnitude, or it could signify the probability with which a binary event has occurred, encoding the detector's uncertainty. The latter interpretation is very useful in the context of detecting sequences, as it allows a population of sequence detectors to encode their uncertainty. It also offers a simple interpretation for the unreliable transmission of spikes by stochastic synapses: With deterministic synapses, a spike volley of a given magnitude will either always or never suffice to trigger a plateau potential. But with stochastic synapses, only a random subset of those spikes will be transmitted. Therefore, the *probability*, that the *effective size* of the spike volley is sufficient to trigger detection depends on the magnitude of the volley. This turns the individual ADSP neuron into a probabilistic detector, which responds to a sequence of input patterns with a probability that reflects the uncertainty encoded in the input signals. For example, a sequence where each required event has occurred with high certainty will lead to a sequence of large spike volleys, which will be detected with high probability. However, if any of the events only occurred with reduced probability, the corresponding spike volleys will be smaller, and hence the detector is more likely to not respond. By combining multiple such detectors into an assembly, we again produce a code of spike-volleys, where the magnitude of the volley is the number of triggered detectors, and thus encodes the probability of the sequence having occurred. This is highly beneficial if we want the population to quickly produce a graded response to a spike-based input [206]! Therefore, ADSP neurons with stochastic synapses can be combined into assemblies or populations that communicate via a stochastic, event-based *population code* (see also chapter 6 of [115])!

pattern actually occurs during this plateau, a new plateau is generated in that segment, which in turn activates another dendrite segment, and so on. This procession can make its way to the soma, where it then triggers a spike, if and only if the entire sequence of events has occurred in the correct order. Importantly, the precise timing of the individual volleys doesn't matter here, as long as they happen in the correct order (and within the specified time-intervals). This allows individual neurons to detect ordered sequences of incoming spike volleys that can last hundreds of milliseconds! Not only does this mechanism allow an individual neuron to detect sequential inputs, but it also provides a simple yet reliable mechanism to do non-linear computations with spike-volleys in continuous time. This also resolves the important question, how the fast (passive) neural membrane potential dynamics can contribute to the detection of patterns on a much slower, behaviorally relevant timescale: Our model only uses the fast dendritic filter to detect brief volleys of coincident spikes, rather than complex temporally extended patterns as in chapter 4. The further integration of that information on a slower time-scale is then due to long-lasting plateau potentials. This is a more realistic interpretation of biological evidence [207]. Since this mechanism is invariant to changes in the precise timing of the spikes (or spike volleys), it would allow the detection of such sequences across multiple time-scales, which might be relevant e.g. for *reactivation*, *replay* or *preplay* of hippocampal place-cell activity [208–210]. If we once

again draw a comparison to concepts from computer science and electronics, this behavior is better described by a *state machine* or *timed automaton* [211, 212], rather than the logic gate we saw in chapter 3.

Contribution 7: Event-based pattern detection in active dendrites

In this manuscript, we derive a simple yet powerful mechanisms of dendritic computation in single neurons from first biological principles. Our model makes use of actively generated dendritic plateau potentials, which provide the neuron with distributed processing elements and memory traces that collectively allow a single neuron's dendritic tree to process information in nonlinear ways and on timescales that exceed the typical timescales of membrane potentials by orders of magnitude. We show how this event-based mechanism can be used to reproduce well known nonlinear computations when viewed from a rate-coding perspective, but also how it goes much further than that by detecting specific long-lasting sequences of spike volleys and integrating information from a vast number of inputs over comparatively long time-scales. A pre-print of this paper is publicly available, and a revised version of the same manuscript is currently still under review.

Reference (see also appendix C, page 187ff for the full text):

J. Leugering, P. Nieters, and G. Pipa, "Event-based pattern detection in active dendrites," *bioRxiv*, p. 690 792, 17, 2020. DOI: 10.1101/690792.

To better understand how such a neuron can process information through the interaction of localized process that are distributed throughout the dendrite, an analogy can be made to decision trees, which rely on a similar hierarchical structure to classify high-dimensional inputs. A modified learning rule for decision trees can therefore even be used to train this biologically motivated neuron model! I investigated this perspective in contribution 8.

The event-based detection of long sequences results in a highly parsimonious code, which offers potentially large savings in energy consumption for biological neurons and neuromorphic hardware alike, which is why we also filed a patent for a digital neuromorphic circuit model of an ADSP neuron that can be implemented in a fully digital electronic circuit (see contribution 9).

7.5 Rate-, phase-, ISI-, or event-coding?

So which code do spiking neurons *actually* use: rate-, phase-, ISI-, or event-coding? This question goes back almost 70 years to [166], who used information theory to analyze the maximum capacity of a synapse under various assumed codes. But the answer is context dependent and differentiating between these paradigms can be surprisingly difficult.

To illustrate how much the answer to this question depends on context, consider for the sake of argument a hypothetical neuron that can detect the presence of a specific odor of a predator and fires a single spike (or a burst of spikes) whenever it detects a few of its molecules, which happens once every couple of milliseconds.

Looking at the exact same spike-train, a proponent of rate-coding could rightly argue: *"The more molecules there are, the higher the firing rate of the neuron, hence the neuron uses a rate-coding approach to encode the concentration of the molecules."*.

Contribution 8: Making spiking neurons more succinct with multi-compartment models

In this conference paper, which accompanies a full-length presentation (which was postponed due to the ongoing SARS-Cov-2 pandemic and is now to be held in March, 2021), I analyze the computational properties of the biologically motivated multi-compartment neuron model of contribution 7 from a machine-learning perspective. By transferring and adapting concepts and learning rules developed for decision trees to this neuron model, I give an intuition for how such a hierarchical structure like a neural dendrite can be useful for computation, and how simple, local learning rules might be enough to optimize such models.

Reference (see also appendix C, page 204ff for the full text):

J. Leugering, "Making spiking neurons more succinct with multi-compartment models," in *Proceedings of the Neuro-Inspired Computational Elements Workshop*, 17, 2020, ISBN: 978-1-4503-7718-8. DOI: 10.1145/3381755.3381763.

A proponent of ISI-coding could argue with equal justification: *"Since the incoming events are essentially Poisson-distributed with time-varying rate, two detections in short succession are a good indicator of a high concentration, so the neuron uses an ISI-coding approach."*.

A proponent of Phase-coding could say for much the same reason: *"The waiting time between phase zero of some reference oscillation and the first detection of a molecule gives an estimate of the concentration, so the neuron uses a phase code."*.

From an event-coding perspective, I would argue: *"The neuron merely signals each event, i.e. the detection of a molecule, with a spike."*.

Neither of these explanations is wrong, but the way this thought experiment was set up, they are not equally *useful*: In order for some downstream neuron to make the decision whether the animal should stay or run away, the rate-coding perspective would require passing the spike train through a low-pass filter with a long enough time-constant to combine the effects of multiple spikes (see chapter 6), which introduces an inevitable and irreversible delay into the signal (see chapter 4). To reach some specific accuracy in the decoded signal, the filter must be longer and hence the response must slower the fewer spikes there are. This is obvious a problem in our thought-experiment, since the animal wouldn't have the luxury of waiting that long! The phase coding approach would require some reference signal, the rate of which limits the response time of the animal and the phase of which introduces an independent random variable. ISI coding would require multiple spikes in order to assess their relative timing. In the extreme case, where a decision must be made based on a single spike, event-coding thus seems to be the only viable explanation of the neuron's code.

But in a slightly different situation, e.g. if a decision to stay or run away is not based on the detection of individual molecules, but rather on whether the average concentration exceeds some higher threshold for some period of time, a rate-coding view might very well offer the better explanation! In fact, two different "decoders" downstream from the neuron might even simultaneously decode the same spike-train for different purposes by ways that can be explained by different paradigms. A crucial take-away of this thought experiment is therefore, that the coding paradigm we use to explain neural firing *depends as much on the receiver* as it does on the transmitter!

Contribution 9: "Neuromorpher Musterdetektor und neuromorphe Schaltkreisanordnung hiermit" (German patent filing)

Based on the insights derived from the neuron model of contribution 7, we designed a digital neuromorphic circuit that can efficiently realize the computation required for active dendritic sequence processing without the need for any general purpose processing elements like arithmetic-logic-units or micro-processors. It implements a processor for temporal patterns and sequences in each hierarchically structured neuron through a combination of pulses of different lengths, just like its biological counterpart. Homogeneous assemblies of multiple such neurons then communicate with each other through a code that serializes and transmits multiple spike-trains over a single binary connection.

Reference (see also appendix C, page 210ff for the full text):

J. Leugering, P. Nieters, and G. Pipa, "Neuromorpher Musterdetektor und neuromorphe Schaltkreisanordnung hiermit," pat. pending.

But one obvious benefit of event-based communication is that it allows for a maximally sparse code, where each event of interest is represented by just a single spike. A neural network thus becomes a network of interconnected pattern detectors, where neurons close to the periphery detect patterns in the input stimuli, while neurons deeper within the nervous system can be thought of as detecting "patterns of patterns". This argument is very appealing for both computational neuroscience and for implementations of spiking neural networks in the context of machine learning and neuromorphic hardware. However, this requires individual neurons to be able to detect relevant patterns in the first place. This requires an extension of our spiking neuron models, which incorporates well-known but often neglected active processes that occur within the dendrites. Due to the solid foundation of this mechanism on biological evidence and the powerful computation it enables, I believe event-coding to be a fundamental, if not the primary, mode of spike-based communication. Consequently, I have come to view rate- and phase-coding as modifications or refinements thereof, which become relevant when some type of event occurs often enough to admit a notion of *rate*, or when its timing is only relevant in relation to some reference signal.

References for chapter 7:

3. W. Maass and C. M. Bishop, *Pulsed Neural Networks*. MIT Press, 2001, 414 pp., ISBN: 978-0-262-63221-8. Google Books: jEug7sJXP2MC (cit. on pp. vii, 32, 59, 61–63, 73).

7. J. V. Stone, *Principles of Neural Information Theory: Computational Neuroscience and Metabolic Efficiency*. Sebtel Press, 2018, 214 pp., ISBN: 978-0-9933679-2-2 (cit. on pp. vii, 33, 34, 46, 55, 60, 62, 66, 71, 98).

13. **J. Leugering**, P. Nieters, and G. Pipa, "Event-based pattern detection in active dendrites," *bioRxiv*, p. 690 792, 17, 2020. DOI: 10.1101/690792 (cit. on pp. viii, 78).

15. **J. Leugering**, "Making spiking neurons more succinct with multi-compartment models," in *Proceedings of the Neuro-Inspired Computational Elements Workshop*, 17, 2020, ISBN: 978-1-4503-7718-8. DOI: 10.1145/3381755.3381763 (cit. on pp. viii, 79).

17. **J. Leugering**, P. Nieters, and G. Pipa, "Neuromorpher Musterdetektor und neuromorphe Schaltkreisanordnung hiermit," pat. pending (cit. on pp. viii, 80).

91. T. Katsuki and R. J. Greenspan, "Jellyfish nervous systems," *Current Biology*, vol. 23, no. 14, R592–R594, 2013, ISSN: 09609822. DOI: 10.1016/j.cub.2013.03.057 (cit. on pp. 16, 74).

107. A. A. Lazar and L. T. Tóth, "Time encoding and perfect recovery of bandlimited signals," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 6, 25, 2003 (cit. on pp. 27, 59, 73).

115. K. Doya, S. Ishii, A. Pouget, and R. P. N. Rao, *Bayesian Brain: Probabilistic Approaches to Neural Coding*. MIT Press, 2007, 341 pp., ISBN: 978-0-262-04238-3. Google Books: bsQMWXXHzrYC (cit. on pp. 32, 77).

141. C. C. Rumsey and L. F. Abbott, "Synaptic Democracy in Active Dendrites," *Journal of Neurophysiology*, vol. 96, no. 5, pp. 2307–2318, 1, 2006, ISSN: 0022-3077. DOI: 10.1152/jn.00149.2006 (cit. on pp. 41, 76).

142. A. Polsky, B. W. Mel, and J. Schiller, "Computational subunits in thin dendrites of pyramidal cells," *Nature Neuroscience*, vol. 7, no. 6, pp. 621–627, 6 2004, ISSN: 1546-1726. DOI: 10.1038/nn1253 (cit. on pp. 41, 76).

166. D. M. MacKay and W. S. McCulloch, "The limiting information capacity of a neuronal link," *The bulletin of mathematical biophysics*, vol. 14, no. 2, pp. 127–135, 1, 1952, ISSN: 1522-9602. DOI: 10.1007/BF02477711 (cit. on pp. 59, 71, 78).

186. A. Borst and F. E. Theunissen, "Information theory and neural coding," *Nature Neuroscience*, vol. 2, no. 11, pp. 947–957, 11 1999, ISSN: 1546-1726. DOI: 10.1038/14731 (cit. on p. 71).

187. P. J. Simmons and R. R. de Ruyter van Steveninck, "Sparse but specific temporal coding by spikes in an insect sensory-motor ocellar pathway," *Journal of Experimental Biology*, vol. 213, no. 15, pp. 2629–2639, 1, 2010, ISSN: 0022-0949, 1477-9145. DOI: 10.1242/jeb.043547 (cit. on p. 71).

188. R. VanRullen, R. Guyonneau, and S. J. Thorpe, "Spike times make sense," *Trends in Neurosciences*, vol. 28, no. 1, pp. 1–4, 1, 2005, ISSN: 0166-2236. DOI: 10.1016/j.tins.2004.10.010 (cit. on pp. 72, 73).

189. G. Buzsáki, "Theta Oscillations in the Hippocampus," *Neuron*, vol. 33, no. 3, pp. 325–340, 31, 2002, ISSN: 0896-6273. DOI: 10.1016/S0896-6273(02)00586-X (cit. on p. 72).

190. Y. Dan and M.-m. Poo, "Spike Timing-Dependent Plasticity of Neural Circuits," *Neuron*, vol. 44, no. 1, pp. 23–30, 30, 2004, ISSN: 0896-6273. DOI: 10.1016/j.neuron.2004.09.007 (cit. on p. 74).

191. J. Nogueira, M. E. Castelló, and A. A. Caputi, "The role of single spiking spherical neurons in a fast sensory pathway," *Journal of Experimental Biology*, vol. 209, no. 6, pp. 1122–1134, 15, 2006, ISSN: 0022-0949, 1477-9145. DOI: 10.1242/jeb.02080. pmid: 16513939 (cit. on p. 74).

192. A. Prindle, J. Liu, M. Asally, S. Ly, J. Garcia-Ojalvo, and G. M. Süel, "Ion channels enable electrical communication in bacterial communities," *Nature*, vol. 527, no. 7576, pp. 59–63, 2015, ISSN: 1476-4687. DOI: 10.1038/nature15709 (cit. on p. 74).

193. X. Cai, "Unicellular ca2+ signaling 'toolkit' at the origin of metazoa," *Molecular Biology and Evolution*, vol. 25, no. 7, pp. 1357–1361, 2008, ISSN: 0737-4038. DOI: 10.1093/molbev/msn077 (cit. on p. 74).

194. D. J. Daley and D. Vere-Jones, *An Introduction to the Theory of Point Processes*, 2nd ed. Springer, 2003, 2 pp., ISBN: 978-0-387-95541-4 978-0-387-21337-8 978-0-387-49835-5 (cit. on p. 75).

195. B. Sotomayor-Gómez, F. P. Battaglia, and M. Vinck, "A geometry of spike sequences: Fast, unsupervised discovery of high-dimensional neural spiking patterns based on optimal transport theory," Neuroscience, preprint, 4, 2020. DOI: 10.1101/2020.06.03.131573 (cit. on p. 75).

196. K. H. Srivastava, C. M. Holmes, M. Vellema, A. R. Pack, C. P. H. Elemans, I. Nemenman, and S. J. Sober, "Motor control by precisely timed spike patterns," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 114, no. 5, pp. 1171–1176, 31, 2017, ISSN: 0027-8424. DOI: 10.1073/pnas.1611734114. pmid: 28100491 (cit. on p. 75).

197. J. Putney, R. Conn, and S. Sponberg, "Precise timing is ubiquitous, consistent, and coordinated across a comprehensive, spike-resolved flight motor program," *Proceedings of the National Academy of Sciences*, vol. 116, no. 52, pp. 26 951–26 960, 26, 2019, ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.1907513116. pmid: 31843904 (cit. on p. 75).

198. G. J. Stuart and N. Spruston, "Dendritic integration: 60 years of progress," *Nature Neuroscience*, vol. 18, no. 12, pp. 1713–1721, 12 2015, ISSN: 1546-1726. DOI: 10.1038/nn.4157 (cit. on p. 76).

199. C. Wilson and Y. Kawaguchi, "The origins of two-state spontaneous membrane potential fluctuations of neostriatal spiny neurons," *The Journal of Neuroscience*, vol. 16, no. 7, pp. 2397–2410, 1, 1996, ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.16-07-02397.1996 (cit. on p. 76).

200. E. A. Rancz, "Dendritic Calcium Spikes Are Tunable Triggers of Cannabinoid Release and Short-Term Synaptic Plasticity in Cerebellar Purkinje Neurons," *Journal of Neuroscience*, vol. 26, no. 20, pp. 5428–5437, 17, 2006, ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.5284-05.2006 (cit. on p. 76).

201. M. London and M. Häusser, "Dendritic Computation," *Annual Review of Neuroscience*, vol. 28, no. 1, pp. 503–532, 2005. DOI: 10.1146/annurev.neuro.28.061604.135703. pmid: 16033324 (cit. on p. 76).

202. G. Stuart and N. Spruston, "Determinants of Voltage Attenuation in Neocortical Pyramidal Neuron Dendrites," *The Journal of Neuroscience*, vol. 18, no. 10, pp. 3501–3510, 15, 1998, ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.18-10-03501.1998 (cit. on p. 76).

203. S. Cash and R. Yuste, "Linear Summation of Excitatory Inputs by CA1 Pyramidal Neurons," *Neuron*, vol. 22, no. 2, pp. 383–394, 1, 1999, ISSN: 0896-6273. DOI: 10.1016/S0896-6273(00)81098-3 (cit. on p. 76).

204. J. Hawkins and S. Ahmad, "Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex," *Frontiers in Neural Circuits*, vol. 10, 2016, ISSN: 1662-5110. DOI: 10.3389/fncir.2016.00023 (cit. on p. 76).

205. J. Brea, A. T. Gaál, R. Urbanczik, and W. Senn, "Prospective Coding by Spiking Neurons," *PLOS Computational Biology*, vol. 12, no. 6, e1005003, 24, 2016, ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1005003 (cit. on p. 76).

206. W. Maass and T. Natschläger, "A Model for Fast Analog Computation Based on Unreliable Synapses," *Neural Computation*, vol. 12, no. 7, pp. 1679–1704, 1, 2000, ISSN: 0899-7667. DOI: 10.1162/089976600300015303 (cit. on p. 77).

207. P. König, A. K. Engel, and W. Singer, "Integrator or coincidence detector? The role of the cortical neuron revisited," *Trends in Neurosciences*, vol. 19, no. 4, pp. 130–137, 1, 1996, ISSN: 0166-2236. DOI: 10.1016/S0166-2236(96)80019-1 (cit. on p. 77).

208. M. Wilson and B. McNaughton, "Reactivation of hippocampal ensemble memories during sleep," *Science*, vol. 265, no. 5172, pp. 676–679, 1994. DOI: 10.1126/science.8036517 (cit. on p. 77).

209. L. Buhry, A. H. Azizi, and S. Cheng. (13, 2011). "Reactivation, Replay, and Preplay: How It Might All Fit Together," [Online]. Available: https://www.hindawi.com/journals/np/2011/203462/ (visited on 12/06/2020) (cit. on p. 77).

210. J.-B. Eichenlaub, B. Jarosiewicz, J. Saab, B. Franco, J. Kelemen, E. Halgren, L. R. Hochberg, and S. S. Cash, "Replay of Learned Neural Firing Sequences during Rest in Human Motor Cortex," *Cell Reports*, vol. 31, no. 5, p. 107 581, 5, 2020, ISSN: 2211-1247. DOI: 10.1016/j.celrep.2020.107581 (cit. on p. 77).

211. R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, no. 2, pp. 183–235, 1994, ISSN: 03043975. DOI: 10.1016/0304-3975(94)90010-8 (cit. on p. 78).

212. C. Sloth and R. Wisniewski, "Complete abstractions of dynamical systems by timed automata," *Nonlinear Analysis: Hybrid Systems*, vol. 7, no. 1, pp. 80–100, 2013, ISSN: 1751570X. DOI: 10.1016/j.nahs.2012.05.003 (cit. on p. 78).

Perfection is finally attained not when there is no longer anything to add, but when there is no longer anything to take away.

— *Terre des hommes*, Antoine de Saint-Exupéry

I feel like I'm just gluing oranges to hair-dryers!

— Anonymous

It's the most powerful words in the world. No argument or eloquence can stand a chance against it. [...] It's: "So what?"

— Dusty Attenborough, *Ginga Eiyū Densetsu*

8 *Conclusion*

As we have seen in chapter 1, the study of artificial neural networks started with an attempt to capture the basic mechanism by which biological neurons process information, and to distill it into an abstract mathematical model. And still today, such artificial neural networks are used as the dominant metaphor to explain "how the brain works", i.e. how it is, that sensory information is processed, decisions are made and actions are taken. The success of deep learning has left many with the illusion that we have finally "cracked the code" of neural information processing, and in a (surprisingly unsurprising) twist, the answer appears to be the same function approximation framework that was already proposed by cyberneticists in the 1960s, and then again by Connectionists in the 1980s. Of course, the capability of recent (deep and/or recurrent) artificial neural networks to solve all sorts of machine learning problems has improved to an impressive degree, and demonstrates the enormous potential of neural networks much more effectively than either cyberneticist or theoretical neuroscientists could. However, these models, which we briefly looked at in chapters 2 and 3, are primarily designed with machine learning applications in mind and hence provide an extremely simplistic, sometimes even misleading, perspective on information processing in the *brain*. This does not discredit deep neural networks in the least, but it shows that despite their common origin, neuroscience and deep learning have fundamentally different objectives, and caution is required when transferring intuitions from one to the other. Goodfellow, Bengio, and Courville [39] summarized this clearly:

> [O]ne should not view deep learning as an attempt to simulate the brain. [...] It is worth noting that the effort to understand how the brain works on an algorithmic level is alive and well. This endeavor is primarily known as "computational neuroscience" and is a separate field of study from deep learning. It is common for researchers to move back and forth between both fields. The field of deep learning is primarily concerned with how to build computer systems that are able to successfully solve tasks requiring intelligence, while the field of computational neuroscience is primarily concerned with building more accurate models of how the brain actually works.

But this is not to say that machine learning and theoretical neuroscience couldn't benefit from each other. Quite to the contrary, I believe that neuroscientists could benefit greatly from the analysis tools developed in machine learning, electrical engineering and computer science, whereas computer scientists and engineers interested in machine learning would do well to take more inspiration from the biological mechanisms analyzed in neuroscience!

Throughout this entire thesis, I have therefore attempted to discuss several inherently biological phenomena that defy this framework in the language of engineering, i.e. dendritic filtering in chapter 4, homeostatic plasticity mechanisms in chapter 5, spike-based

communication in chapter 6 and finally event-based mechanisms of neural computation in chapter 7.

To decide whether these additions are *actually* instrumental for information processing or merely abstract descriptions of a needlessly complicated biological mechanism, we will have to put them to the test. I have come to believe that the best way to do that is by looking across the domain boundaries between neuroscience and adjacent disciplines, in order to find inspiration and to test ideas in a less forgiving environment outside one's own control. Particularly the *embodiment* of concepts from theoretical neuroscience in neuromorphic hardware appeals to me as a tough, but honest benchmark that makes it possible to evaluate the merit of many theoretical models of neural computation "in the real world". I therefore believe that many future innovations in theoretical neuroscience will originate in or be driven by such application oriented fields, which also motivated my own transition towards neuromorphic hardware.

References for chapter 8:

39. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT press Cambridge, 2016, vol. 1 (cit. on pp. 4, 5, 9, 12, 85).

A Appendix for chapter 4

Note:

Some derivations in this appendix can also be found in similar form in the original work on the Gamma neuron [125, 126], others come from (unpublished) work in our lab. I have decided to re-derive and compile them here for conciseness and completeness. At some points, we have also slightly extended the Gamma neuron beyond its original definition or taken it out of its original context, but decided to keep the established name for our more general extension.

A.1 Equivalence between filtering and continuous delays

Filter and (continuous) delay operators are very closely related.

To show the first direction of this relationship, consider a continuous signal $s(t)$. Any operator D that delays s by some fixed delay $\Delta t \geq 0$ can be expressed as a causal filter with a shifted Dirac-distribution as its kernel:

$$Ds(t) = s(t - \Delta t) = \int_{-\infty}^{t} s(\tau)\kappa(t - \tau)\mathrm{d}\tau$$
$$\kappa(t) := \delta(t - \Delta t)$$

Conversely, for a continuous signal $s(t)$ and a causal filter with (piece-wise) continuous kernel $\kappa(t)$, we can use the Riemann integral to approximate the effect of the kernel κ by a linear combination $\bar{\kappa}$ of shifted and scaled δ-pulses:

$$
\begin{aligned}
(s * \kappa)(t) &= \int_{-\infty}^{t} s(\tau)\kappa(t-\tau)\mathrm{d}\tau \\
&= \lim_{\Delta t \to 0} \Delta t \sum_{k=0}^{\infty} s(t - k \cdot \Delta t)\kappa(k \cdot \Delta t) \\
&= \lim_{\Delta t \to 0} \Delta t \sum_{k=0}^{\infty} \left(\int_{-\infty}^{t} s(\theta)\delta(t - \theta - k \cdot \Delta t)\mathrm{d}\theta \right) \kappa(k \cdot \Delta t) \\
&= \int_{-\infty}^{t} s(\theta) \lim_{\Delta t \to 0} \Delta t \sum_{k=0}^{\infty} \kappa(k \cdot \Delta t)\delta(t - \theta - k \cdot \Delta t)\mathrm{d}\theta \\
&= (s * \bar{\kappa})(t) \qquad \text{where } \bar{\kappa}(t) = \lim_{\Delta t \to 0} \Delta t \sum_{k=0}^{\infty} \kappa(k \cdot \Delta t)\delta(t - k \cdot \Delta t)
\end{aligned}
$$

A.2 Transfer function of the Gamma neuron

Consider the Gamma neuron as described in section 4.4. We can analyze the linear system realized by its dendritic filter in the Laplace domain. Let us denote the Laplace transform of the j^{th} input signal $s(t)$ with $S_j(s)$, and the Laplace transform of the output signal of the i^{th} tap of the dendritic filter with $X_i(s)$, where $X_1(s)$ is the output of the filter tap closes to the soma. To simplify the analysis, we separate the feed-forward and the feed-back paths of the model by defining the *open-loop impulse-response* to the j^{th} *external* input signal (denoted κ_j^{fwd}) as well as the open-loop impulse-response of the neuron to its *own output* y (denoted κ^{fb}).

In the absence of feedback, an impulse arriving at the i^{th} tap from the soma has to traverse i filters on its way to the soma, and hence produces (in the Laplace domain) a filter response $\left(\frac{\alpha}{s+\alpha}\right)^i$ — which corresponds to the probability density function of a Gamma distribution with coefficients i and α.

For a neuron with n taps and m input signals and time-constant α, these individual impulse responses of the taps are scaled by the forward and feedback weights $w_{i,j}$ and v_i, respectively, and linearly combined to yield the effective dendritic filter kernels:

$$
\begin{aligned}
\kappa_j^{\text{fwd}} &= \sum_{i=1}^{n} \left(\frac{\alpha}{s+\alpha}\right)^i w_{i,j} \\
\kappa^{\text{fb}} &= \sum_{i=1}^{n} \left(\frac{\alpha}{s+\alpha}\right)^i v_i
\end{aligned}
$$

We can then derive the transfer function:

$$
\begin{aligned}
X_1(s) &= \kappa^{\text{fb}} X_1(s) + \sum_{j=1}^{m} \kappa_j^{\text{fwd}} S_j(s) \\
\Rightarrow X_1(s) &= \sum_{j=1}^{m} \frac{\kappa_j^{\text{fwd}}}{\mathbb{1} - \kappa^{\text{fb}}} S_j(s) \\
&= \sum_{j=1}^{m} \frac{\sum_{i=1}^{n} \left(\frac{\alpha}{s+\alpha}\right)^i w_{i,j}}{\mathbb{1} - \sum_{i=1}^{n} \left(\frac{\alpha}{s+\alpha}\right)^i v_i} S_j(s) \\
&= -\sum_{j=1}^{m} \frac{\sum_{i=1}^{n} \alpha^i (s+\alpha)^{n-i} w_{i,j}}{\sum_{i=0}^{n} \alpha^i (s+\alpha)^{n-i} v_i} S_j(s) \\
&= -\sum_{j=1}^{m} \frac{\sum_{i=1}^{n} \sum_{k=0}^{n-i} \binom{n-i}{k} s^k \alpha^{n-k} w_{i,j}}{\sum_{i=0}^{n} \sum_{k=0}^{n-i} \binom{n-i}{k} s^k \alpha^{n-k} v_i} S_j(s)
\end{aligned}
$$

The last expression can be simplified using matrix-vector multiplications:

$$(w^j)_i \coloneqq w_{i,j}, \quad w^j \in \mathbb{R}^n$$
$$(M)_{k,i} \coloneqq \begin{cases} \alpha^{n-k}\binom{n-i}{k} & \text{if } k \leq i \\ 0 & \text{otherwise} \end{cases}, \quad M \in \mathbb{R}^{n \times n}$$

The matrix M above is an invertible matrix, therefore the coefficient vectors Mw^j and Mv of this rational transfer function can be freely determined by an appropriate choice of the feed-forward and feedback weights $w_{k,j}$ and v_k, respectively. The resulting transfer function of the dendritic filter simplifies to:

$$X_1(s) = \sum_{j=1}^{m} \frac{\sum_{k=1}^{n} s^{k-1}(Mw^j)_k}{\mathbb{1} - \sum_{k=1}^{n} s^k (Mv)_k} S_j(s).$$

The special case of a Gamma neuron without feedback occurs when setting the feedback weights v to zero, which results in the simpler form:

$$X_1(s) = \sum_{j=1}^{m} \left(\sum_{k=1}^{n} s^{k-1}(Mw^j)_k \right) S_j(s)$$

The Gamma neuron with n filter taps and linear feedback can thus be used to implement a dendritic filter with arbitrary *proper rational* transfer function (for a single input signal) with degrees up to $n-1$ in the numerator and n in the denominator. For multiple inputs, the numerator of this transfer function can be individually chosen for each input, whereas the denominator (determined by the feedback coefficients) is shared among all inputs.

The ability to freely place zeros and poles (i.e. zeros of the denominator) of the transfer function makes this type of filter bank with feedback extremely versatile. It is capable of implementing a wide range of practically relevant filters, such as higher-order Butterworth, Chebyshev and Elliptic filters in low- and band-pass form. High-pass and band-stop filter can similarly be implemented, but some limitations apply. For an in-depth look at continuous filter design, see e.g. chapter 7 of [124].

Derivatives of Gamma filters are Gamma filters

The time-derivative of the open-loop impulse-response of tap $k \geq 2$ in the Gamma neuron's filter can be expressed simply in terms of just two neighboring taps (see also [125]):

$$\begin{aligned} \kappa_k(t) &= \frac{\alpha^k}{\Gamma(k)} t^{k-1} \exp(-\alpha t) \\ \kappa_k'(t) &= \frac{\alpha^k}{\Gamma(k)} \left(t^{k-2} \exp(-\alpha t) - \alpha t^{k-1} \exp(-\alpha t) \right) \\ &= \frac{\alpha}{k-1} \kappa_{k-1}(t) - \alpha \kappa_k(t) \end{aligned}$$

Therefore, the time derivative of *any* filter constructed without feedback by linear combination of the taps $k \geq 2$ can be implemented by the same filter bank, as well. The same argument applies to the feedback path, and hence the time-derivative can be implemented for (most) filters constructed with feedback, as well.

In short, this means that the Gamma neuron model can combine filtering and differentiation in the linear operator implemented by its dendrite. The Gamma neuron therefore posses all the capabilities required of a *PID-controller*: proportional input (i.e. a δ-impulse response, e.g. approximated by a quickly decaying exponential filter), integration of input

(e.g. approximated by a slowly decaying exponential filter) and differentiation of input. See also [129] for more information about the design of PID-controllers.

A.3 The ring of Gamma filters

As shown in appendix A.2, the Gamma neuron's dendritic filter has a proper rational transfer function with order $\leq n$ in the denominator. While rational functions form a *field*, *proper* rational functions only form a *ring*, since the multiplicative inverse of a proper rational function would not necessarily be proper. This has an analog physical interpretation: The sum of two causal filters or the concatenation of two causal filters (with rational transfer functions) is again a causal filter (with rational transfer function), but a causal filter cannot be inverted by application of a causal filter (consider as a simple counter-example, that inverting a delay would require an acausal advance of the signal). This ring is commutative, but it doesn't contain a multiplicative identity (this would correspond to a Dirac-δ in the time-domain), and could therefore be called a *pseudo-ring*.

While multiplicative inverses don't exist in this ring, we can approximate a solution of $\kappa_2 = \kappa^\dagger \kappa_1$ for $\kappa^\dagger$ by Euclidean division with remainder r:

$$\exists! \kappa^\dagger, r \,:\, \kappa_2 = \kappa^\dagger \kappa_1 + r.$$

Now consider a signal s and its filtered version $\tilde{s} := \kappa_1 * s$. We then have

$$\kappa^\dagger * \tilde{s} = (\kappa_2 - r) * s$$

Therefore, while we cannot implement *de*convolution, we can use filtering to "replace" the effect of the filter κ_1 with the effect of the filter $\kappa_2 - r$, an approximation of κ_2. If $\kappa_2(s)$ is now chosen e.g. to approximate the delay $\delta(t - T)$ for some large T, then this *re*convolution with the filter $\kappa^\dagger$ approximates a delayed *de*convolution, i.e. it approximately recovers the delayed original signal $s(t - T)$!

B Appendix for chapter 6

Note:

Many of the derivations in this appendix are not new results, and can hence be found across standard literature like [105, 174, 213]. I have nevertheless decided to re-derive them here from scratch and compile them in order to provide a more concise summary and to allow for a direct comparison between the different models using a common language.

B.1 Rate-coding with (L)IF neurons

Activation function of integrate-and-fire neurons

The pure integrate-and-fire neuron integrates its input up until it hits a threshold θ, at which point it resets and the process begins anew. For an incoming signal $s(t)$ with integral $S(t) = \int_0^t s(t)$, the integrate-and-fire neuron thus produces spikes at the times $t_k = S^{-1}(k\theta), k \in \mathbb{N}$. If we filter the resulting spike train $\chi(t) = \sum_{t_k} \delta(t - t_k)$ by a filter $\kappa(t)$, we get the decoded signal z_{IF}:

$$z_{\text{IF}}(t) = \theta \cdot (\kappa * \chi)(t) \tag{B.1}$$

$$= \theta \sum_{t_k \leq t} \int_{-\infty}^{t} \kappa(t - \tau)\delta(\tau - t_k)\mathrm{d}\tau \tag{B.2}$$

$$= \theta \sum_{t_k \leq t} \kappa(t - t_k) \tag{B.3}$$

For a constant signal $s(t) = c \geq 0$, we get $S(t) = ct$ and hence $t_k = {}^{k\theta}/_{c}$. To simplify notation, we can introduce $K_t := \max\{k : t_k \leq t\} = \lfloor {}^{ct}/_{\theta} \rfloor$, which is the index of the last spike before time t. If we choose the exponential kernel $\kappa(t) = H(t)\alpha \exp(-t\alpha)$, where H is the Heaviside

step-function and $\alpha > 0$ sets the time-scale of the filter, we can compute z_{IF}:

$$z_{\text{IF}}(t) = \sum_{k=-\infty}^{K_t} \alpha\theta \exp(-\alpha(t - k\theta/c)) \tag{B.4}$$

$$= \frac{\alpha\theta}{1 - \exp(-\alpha\theta/c)} \exp(\alpha(K_t\theta/c - t)) \tag{B.5}$$

$$= \frac{\alpha\theta}{1 - \exp(-\alpha\theta/c)} \exp(-\alpha\Delta t) \quad \text{where } \Delta t \coloneqq t - t_{K_t} \tag{B.6}$$

This expression depends only on the relative time Δt since the previous spike, and repeats after every spike. Therefore, the "decoded" spike-train is a periodic signal of discontinuous jumps after every spike with period $t_{k+1} - t_k = \theta/c$, followed by exponential decay. We can thus compute the mean signal by averaging it between two successive spikes:

$$\bar{z}_{\text{IF}}(c) = \frac{1}{\theta/c} \int_0^{\theta/c} z_{\text{IF}}(t) \mathrm{d}\Delta t \tag{B.7}$$

$$= \frac{\alpha\theta}{\theta/c} \cdot \frac{1}{1 - \exp(-\alpha\theta/c)} \int_0^{\theta/c} \exp(-\alpha(\Delta t)) \mathrm{d}\Delta t \tag{B.8}$$

$$= c \cdot \frac{1 - \exp(-\alpha\theta/c)}{1 - \exp(-\alpha\theta/c)} \tag{B.9}$$

$$= c \tag{B.10}$$

Therefore, the (linearly decoded) output of the integrate-and-fire neuron is given by a rectified linear function $\bar{z}_{\text{IF}}(c) = \max(0, c)$ of the constant input c.

Decoding error of integrate-and-fire spike-trains

Given $z_{\text{IF}}(t)$ and $\bar{z}_{\text{IF}}(c)$, we can similarly calculate the expected root-mean-squared error (RMSE) of the IF neuron.

$$\text{MSE}_{\text{IF}}(c) = \frac{1}{\theta/c} \int_0^{\theta/c} (z_{\text{IF}}(\Delta t) - \bar{z}_{\text{IF}}(c))^2 \mathrm{d}\Delta t \tag{B.11}$$

$$= \frac{c}{\theta} \int_0^{\theta/c} z_{\text{IF}}(\Delta t)^2 \mathrm{d}\Delta t - (\bar{z}_{\text{IF}}(c))^2 \tag{B.12}$$

$$= \frac{c\alpha^2\theta}{(1 - \exp(-\frac{\alpha\theta}{c}))^2} \int_0^{\theta/c} \exp(-2\alpha\Delta t) \mathrm{d}\Delta t - c^2 \tag{B.13}$$

$$= \frac{c\alpha\theta(1 + \exp(-\frac{\alpha\theta}{c}))}{2(1 - \exp(-\frac{\alpha\theta}{c}))} - c^2 \tag{B.14}$$

$$= \frac{c\alpha\theta}{2} \coth(\frac{\alpha\theta}{2c}) - c^2 \tag{B.15}$$

$$\text{RMSE}_{\text{IF}}(c) = \sqrt{\frac{c\alpha\theta}{2} \coth(\frac{\alpha\theta}{2c}) - c^2} \tag{B.16}$$

$$\lim_{c\to\infty} \text{MSE}_{\text{IF}}(c) = \frac{\alpha^2\theta^2}{12} \quad \Rightarrow \quad \lim_{c\to\infty} \text{RMSE}_{\text{IF}}(c) \approx \frac{\alpha\theta}{\sqrt{12}} \tag{B.17}$$

The last equation shows, that in the limit of relatively high firing rates (which we typically assume when talking about rate coding) the RMSE only depends on the product $\alpha\theta$ and becomes independent of c. To reduce the error, we have to either use a slower kernel with smaller α or lower the threshold θ.

A different way to interpret this result is to consider that in the steady-state, the filtered spike-train makes a jump of fixed magnitude $\alpha\theta$ after each spike and then relaxes back to the same initial value z_0 before spiking again. The higher the firing rate is, the more this exponential relaxation looks like a linear decrease, and the more the filtered spike-train looks like a saw-tooth wave. If we were to estimate the stationary distribution of the error that results from sampling the filtered signal z_{IF} at a random point in time, the residual r would then be uniformly distributed in the range $[c - \alpha\theta/2, c + \alpha\theta/2]$ with mean value c and standard deviation $\alpha\theta/\sqrt{12}$. We will make use of this probabilistic perspective for comparing the information content of various encodings in appendix B.3.

Activation function of leaky integrate-and-fire neurons

Leaky integrate-and-fire (LIF) neurons are very similar to integrate-and-fire neurons, but the integrator is replaced by a first-order low-pass filter with leak-rate α. We will choose the same α for both the LIF neuron and the decoder (which would be another LIF neuron in a spiking neural network, anyway). LIF neurons also fire periodically in response to constant inputs, albeit with a lower firing rate that depends *non-linearly* on the input and the leak-rate α. We can therefore model the LIF neuron as an IF neuron, whose input signal $c' = \nu(c)$ has been nonlinearly transformed. This nonlinear distortion ν can be characterized as follows:

For constant input c and following a reset at time 0, the LIF neuron's membrane potential follows the trajectory $c/\alpha(1 - \exp(-\alpha t))$, i.e. it exponentially approaches c/α rather than growing at constant rate c like in the IF neuron. The threshold θ is reached at time $t = -1/\alpha \log(1 - \alpha\theta/c)$. An IF neuron would produce the same periodic firing for a different constant input c':

$$-1/\alpha \log(1 - \alpha\theta/c) = \theta/c' \tag{B.18}$$

$$\Leftrightarrow \nu(c) \coloneqq c' = -\frac{\alpha\theta}{\log(1 - \alpha\theta/c)} \tag{B.19}$$

For $c \gg \theta$, this has the asymptote $c'(c) \approx c + \alpha\theta/2$.

This implies, that the LIF neuron has the transfer function is

$$\bar{z}_{\text{LIF}}(c) = \bar{z}_{\text{IF}}(\nu(c)) = \max(0, -\frac{\alpha\theta}{\log(1 - \alpha\theta/c)}).$$

If we use the asymptotic approximation, this reduces further to simply

$$\bar{z}_{\text{LIF}}(c) \approx \max(0, c + \alpha\theta/2).$$

Decoding error of leaky integrate-and-fire spike-trains

Using the same trick of substituting in $c' = \nu(c)$ for the input of an IF neuron, we can also calculate the RMSE of the LIF neuron. But since we are mostly interested in the high firing-rate regime, where we saw that the RMSE of the IF neuron does not depend on c at all, we get the same approximation for the LIF neuron as well:

$$\text{RMSE}_{\text{LIF}}(c) \approx \frac{\alpha\theta}{\sqrt{12}}.$$

B.2 Rate-coding with linear-nonlinear-Poisson neurons

To allow for a direct comparison between the LNP neuron and the (L)IF neuron from appendix B.1, let's now imagine that we filter the spiking output of an LNP neuron in response to a (piece-wise) constant input. The linear-nonlinear-Poisson spiking neuron produces

stochastic spikes at the (time-varying) rate $s(t)$ by an inhomogeneous Poisson process, i.e. the number N of spikes in the time-interval $[0, t]$ is a Poisson random variable with expected value $\mathbb{E}[N] = \int_0^t \lambda s(\tau)\mathrm{d}\tau = \lambda(S(t) - S(0))$. Here, λ is a firing-rate gain parameter that plays the same role as $1/\theta$ does in the LIF neuron. When decoding the spike-train, we therefore weigh each spike by $1/\lambda$ instead of θ to compensate for the gain. In contrast to the (L)IF neuron, the spike-times are *not* periodically spaced for the LNP neuron, so we have to follow a slightly different approach to derive the mean and RMSE of the decoded spike-train.

For each of these N spikes, the spike times $t_k \sim \mathrm{Uniform}(0, t)$ are independent and identically distributed random variables. If we use the same constant signal $s(t) = c$ and exponential filter κ as in appendix B.1, we can compute the effect γ_k of the individual spikes on the decoded signal, the combined effect Γ of all spikes since time $t = 0$, as well as a couple of expectations that will be useful later:

$$\gamma_k(t) = \frac{\alpha}{\lambda}\exp(-\alpha(t - t_k)) \tag{B.20}$$

$$\Gamma(t) = \sum_{0 \le t_k \le t} \gamma_k(t) \tag{B.21}$$

$$\mathbb{E}[N] = \lambda \int_0^t c\mathrm{d}\tau = \lambda t c \tag{B.22}$$

$$\mathbb{E}[\gamma_k(t)] = \frac{1}{t}\int_0^t \frac{\alpha}{\lambda}\exp(-\alpha(t-\tau))\mathrm{d}\tau = \frac{1}{\lambda t}(1 - \exp(-\alpha t)) \tag{B.23}$$

$$\mathbb{E}[\gamma_k(t)^2] = \frac{\alpha^2}{t\lambda^2}\int_0^t \exp(-\alpha(t-\tau))^2\mathrm{d}\tau = \frac{\alpha}{2t\lambda^2}(1 - \exp(-2\alpha t)) \tag{B.24}$$

$$\mathbb{E}[\Gamma(t)|N] = N\mathbb{E}[\gamma_i(t)] = \frac{N}{\lambda t}(1 - \exp(-\alpha t)) \tag{B.25}$$

$$\mathbb{E}[\Gamma(t)] = \mathbb{E}[N]\mathbb{E}[\gamma_i(t)] = c(1 - \exp(-\alpha t)) \tag{B.26}$$

With these results, we can finally derive the decoded signal and its expected value:

$$z_{\mathrm{LNP}}(t) = 1/\lambda(\kappa * \chi)(t) \tag{B.27}$$

$$= \sum_{-\infty \le t_k \le t} \gamma_k(t) \tag{B.28}$$

$$= z_{\mathrm{LNP}}(0)\exp(-\alpha t) + \Gamma(t) \tag{B.29}$$

$$\mathbb{E}[z_{\mathrm{LNP}}(t)|N] = \mathbb{E}[z_{\mathrm{LNP}}(0)]\exp(-\alpha t) + \mathbb{E}[\Gamma(t)|N] \tag{B.30}$$

$$\mathbb{E}[z_{\mathrm{LNP}}(t)] = \mathbb{E}[z_{\mathrm{LNP}}(0)]\exp(-\alpha t) + \mathbb{E}[\Gamma(t)] \tag{B.31}$$

$$= \mathbb{E}[z_{\mathrm{LNP}}(0)]\exp(-\alpha t) + c(1 - \exp(-\alpha t)) \tag{B.32}$$

We can solve the last equation easily by using the fact that the filtered signal (and hence its expectations) must be time-shift-invariant, i.e. $\mathbb{E}[z_{\mathrm{LNP}}(t)] = \mathbb{E}[z_{\mathrm{LNP}}(0)]$. For the mean value $\bar{z}_{\mathrm{LNP}}(c)$ of the decoded signal in response to constant input $s(t) = c$, we thus get:

$$\bar{z}_{\mathrm{LNP}}(c) = \mathbb{E}[z_{\mathrm{LNP}}(t)] = c \tag{B.33}$$

Filtering an LNP neuron's spike-train thus also produces an unbiased estimate of its (constant) input.

Calculating the RMSE for the LNP neuron is more difficult, because it involves a random number of spikes N as well as the random times of each individual spike. But we can use the *law of total variance* and apply the same trick as above, i.e. enforcing that the MSE must be

time-shift-invariant:

$$\begin{aligned}
\mathrm{Var}[z_{\mathrm{LNP}}(t)|N] &= \mathrm{Var}[z_{\mathrm{LNP}}(0)\exp(-\alpha t) + \Gamma(t)] && \text{(B.34)}\\
&= \mathrm{Var}[z_{\mathrm{LNP}}(0)\exp(-\alpha t)] + \mathrm{Var}[\Gamma(t)] && \text{(B.35)}\\
&= \mathrm{Var}[z_{\mathrm{LNP}}(0)]\exp(-2\alpha t) + N\mathrm{Var}[\gamma_i(t)] && \text{(B.36)}\\
\mathrm{MSE}_{\mathrm{LNP}}(c) &= \mathrm{Var}[z_{\mathrm{LNP}}(t)] && \text{(B.37)}\\
&= \mathbb{E}(\mathrm{Var}[z_{\mathrm{LNP}}(t)|N]) + \mathrm{Var}(\mathbb{E}[z_{\mathrm{LNP}}(t)|N]) && \text{(B.38)}\\
&= \mathbb{E}(\mathrm{Var}[z_{\mathrm{LNP}}(0)]\exp(-2\alpha t) + N\mathrm{Var}[\gamma_i(t)]) + \mathrm{Var}(\mathbb{E}[z_{\mathrm{LNP}}(0)]\exp(-\alpha t) + N\mathbb{E}[\gamma_i(t)]) && \text{(B.39)}\\
&= \mathrm{Var}[z_{\mathrm{LNP}}(0)]\exp(-2\alpha t) + \mathbb{E}[N]\mathbb{E}[\gamma_i(t)^2] + (\mathrm{Var}(N) - \mathbb{E}[N])\mathbb{E}[\gamma_i(t)]^2 && \text{(B.40)}\\
&= \mathrm{Var}[z_{\mathrm{LNP}}(0)]\exp(-2\alpha t) + \mathbb{E}[N]\mathbb{E}[\gamma_i(t)^2] && \text{(B.41)}\\
\mathrm{Var}[z_{\mathrm{LNP}}(0)] = \mathrm{Var}[z_{\mathrm{LNP}}(t)] &\Leftrightarrow \mathrm{MSE}_{\mathrm{LNP}}(c) = \frac{\alpha c}{2\lambda} && \text{(B.42)}\\
\mathrm{RMSE}_{\mathrm{LNP}}(c) &= \sqrt{\frac{\alpha c}{2\lambda}} && \text{(B.43)}
\end{aligned}$$

Since the filtered spike-train of the LNP neuron can be viewed as a sum of independent and identically distributed random variables, we can assume that the distribution of the membrane potential (for a sufficiently high firing rate) approaches a normal distribution due to the central limit theorem. In contrast to the (L)IF neuron, the RMSE hence increases proportionally to $\sqrt{c}$! We will make use of this in appendix B.3.

B.3 The entropy of LIF and LNP encoding

Let's assume as in appendices B.1 and B.2 that $c \gg 0$, so that we may assume a Gaussian distribution of errors when decoding the LNP neuron's output and a uniform distribution for the LIF neuron. The entropy of a Gaussian distribution with standard-deviation $\mathrm{RMSE}_{\mathrm{LNP}}(c) = \sqrt{\frac{\alpha c}{2\lambda}}$ is $E^{\sigma}_{\mathrm{Gauss}} = 1/2\log(\pi e\frac{\alpha c}{\lambda})$ and the entropy of a uniformly distributed random variable on the interval $[0, \alpha\theta]$ is $E^{\theta}_{\mathrm{Uniform}} = \log(\alpha\theta)$. Therefore, measurements of the two neurons' firing rates are similarly informative if $E^{\sigma}_{\mathrm{Gauss}} = E^{\theta}_{\mathrm{Uniform}}$, i.e. if $\theta = \sqrt{\frac{\pi e c}{\alpha\lambda}}$. Since for a constant signal, the mean firing rate r_{LNP} of the LNP neuron scales linearly with λ whereas the mean firing rate r_{LIF} of the LIF neuron scales linearly with $1/\theta$, we can see that in order to reach similar performance, we have to have $r_{\mathrm{LIF}} \propto \sqrt{\frac{\alpha r_{\mathrm{LNP}}}{\pi e c}}$.

B.4 Spike-coding under metabolic constraints

B.4.1 Maximizing information-rate under metabolic constraints

Following the procedure already outlined in section 5.2, we can find the optimal distribution of firing rates Y that maximizes the information-rate under certain metabolic constraints. For example, we might want to keep the expected (RMS) error below some limit θ_{error}, and the energy cost below the limit θ_{cost}. The optimal firing rate distribution must then be of the form:

$$p(y) = \exp(\lambda_0 \mathbb{1}_R(y) + \lambda_1 \mathrm{cost}(y) + \lambda_2 \mathrm{error}(y)),$$

and the optimal coefficients of the distribution $\lambda^* = \begin{pmatrix}\lambda_0^* & \lambda_1^* & \lambda_2^*\end{pmatrix}^T$ can be found by optimization of:

$$\lambda^* = \arg\max_{\lambda}\left(\lambda^T\theta - \int_R p(y)\mathrm{d}y\right) \quad \text{subject to} \quad \lambda_1, \lambda_2 \geq 0,$$

where $R = [0, R_{\max}]$ is the range of admissible firing rates and $\theta = \begin{pmatrix} 1 & -\theta_{\text{cost}} & -\theta_{\text{error}} \end{pmatrix}$ holds the constraints we wish to enforce. [1] In general, it might be difficult to derive a closed-form solution for this, but if we are for instance willing to assume that (a) the firing rate is bounded, (b) the cost increases linearly with the rate and (c) RMSE is (almost) independent of y (as is the case for the (L)IF neuron), then the equation above reduces to a truncated exponential distribution! [2]

[1] The first term 1 and corresponding coefficient λ_0 originate from the constraint that the distribution of Y has to be normalized over R, the other two from the metabolic constraints.

[2] If the cost constraint is redundant, this simplifies further to a uniform distribution.

B.4.2 Maximizing metabolic efficiency

While optimizing the through-put of a neuron seems reasonable from an information bottleneck perspective, there is convincing evidence that (some) biological neurons appear to be optimized for metabolic efficiency instead, firing at rates as low as two spikes per second and with an information content as high as 5.6 bits per spike (see e.g. chapter 4 of [7]). This may be much less than the maximum bit-rate that a single neuron could deliver in principle, but it appears to make optimal use of the invested energy. To find the optimal firing rate distribution P_Y^*, we can proceed as follows: If we again use $\text{cost}(y)$ to denote the power required for maintaining a firing rate y, the cost associated with the firing rate distribution P_Y is $\text{cost}_{P_Y} = \mathbb{E}_{P_Y}[\text{cost}(y)]$. We can then express the metabolic efficiency of the neuron $\varepsilon = \frac{I_{P_Y}}{\text{cost}_{P_Y}}$ where I_{P_Y} denotes the information-rate of the neuron for firing rate distribution P_Y. If the cost was independent of the firing rate distribution (i.e. if generating spikes required no additional energy), then most the "powerful" encoding from above would also be the most metabolically efficient. But if we make the more realistic assumption, that each spike costs a finite amount of energy e_{spike} in addition to the *static power cost*$_{\text{static}}$ required to keep the neuron operational, the metabolic efficiency is optimized by the distribution

$$P_Y^* = \arg\max_{P_Y} \frac{I_{P_Y}}{\mathbb{E}[Y] \cdot e_{\text{spike}} + \text{cost}_{\text{static}}}.$$

If we only consider exponential distributions and assume a one-to-one deterministic encoding by the neuron, then the information rate (which is then proportional to the entropy of Y) is just a function of the expected firing rate $I_{P_Y} \propto \log_2(\mathbb{E}[Y]) + 1$, and hence P_Y^* becomes a function of just the mean firing rate, as well. In this case, we see that the metabolic efficiency would be maximized for

$$\mathbb{E}[Y] = \frac{\gamma}{\gamma W(2\exp(-1))}, \qquad \text{where } \gamma := {}^{\text{cost}_{\text{static}}}/_{e_{\text{spike}}} \text{ and } W \text{ is Lambert's function.}$$

Note that this gives us a finite, optimal firing rate for the neuron, even if we impose no hard limits on either the energy constraint or the maximum firing rate.

C *Full-text sources of own contributions*

In the following, I have compiled relevant publications for the consideration by the PhD-Committee. The documents are included "as-is" in the exact form they were published or submitted for publishing (except, of course, for the added page numbers and a disclaimer in the margins). The documents are included for evaluation and archiving purposes only and are not meant for further distribution.

Bistable Perception in Conceptor Networks

Felix Meyer zu Driehausen[1(✉)], Rüdiger Busche[1,2], Johannes Leugering[1], and Gordon Pipa[1]

[1] Institute of Cognitive Science, Osnabrück University, Wachsbleiche 27, 49076 Osnabrück, Germany
{fmeyerzudrie,rbusche,jleugeri,gpipa}@uni-osnabrueck.de
[2] AIM Agile IT Management GmbH, Habichtshorststraße 1, 30655 Hannover, Germany

Abstract. Bistable perception describes the phenomenon of perception alternating between stable states when a subject is presented two incompatible stimuli. Besides intensive research in the last century many open questions remain. As a phenomenon occurring across different perceptual domains, understanding bistable perception can help to reveal properties of information processing in the human brain. It becomes apparent that bistable perception involves multiple distributed processes and several layers in the hierarchy of sensory processing. This observation directs research towards general models of perceptual inference and to the question whether these models can account for the spontaneous subjective changes in percepts that subjects experience when shown rivalling stimuli. We implemented a recurrent generative model based on hierarchical conceptors to investigate its behaviour when fed an ambiguous signal as input. With this model we can show that (1) it is possible to obtain precise predictions about the properties of bistable perception using a general model for perceptual inference, (2) hierarchical processes allow for reduction in prediction error, (3) random switches in the percept of the network are due to noise in the input and (4) dominance times exhibit a gamma distribution of stimulus dominance times compatible with experimental findings in psychophysics. Code for the experiments is available at https://github.com/felixmzd/Conceptors.

Keywords: Bistable perception · Predictive coding · Conceptors

1 Introduction

Bistable perception arises when a sensual modality is presented with a stimulus that is too ambiguous to be resolved by a unique interpretation. While there are many examples of visual stimuli that evoke bistable perception such as the Neckar cube [14] or binocular rivalry [15], the phenomenon has also been observed for other sensual modalities such as olfaction [16], audition [7] and the tactile sense [5].

I. V. Tetko et al. (Eds.): ICANN 2019, LNCS 11731, pp. 24–34, 2019.
https://doi.org/10.1007/978-3-030-30493-5_3

As such a general phenomenon, bistable perception seems to be a direct result of properties inherent in the information processing of the human brain.

The common characteristic of bistable perception across sensual modalities are spontaneous alternations in percept between the interpretations of the stimulus while the presented stimulus itself remains constant. The exact timepoints of change of the percept can not be predicted and are apparently random. However, the distribution of dominance time durations was shown to be relatively constant across examples of bistable stimuli and resembles a gamma like or right-skewed normal distribution [3].

Several models have been proposed that attempt to account for the established results on the timing of dominance intervals. Some also take more recent evidence on the distributed neural processing of rivalling stimuli into account [1]. A model for the condition of binocular rivalry by Freeman consists of four parallel visual channels, two driven by the left eye and two by the right. Therein, the succession of cortical levels is represented by several consecutive processing stages for each channel [8]. Dayan describes a model wherein the alternation between the percepts can be generated by competition between top-down cortical explanations for the inputs instead of direct competition between the inputs [6]. In a similar spirit, Hohwy et al. offer an explanation of the binocular rivalry condition in terms of predictive coding [10].

While computational models accurately predict the properties of bistable perception the often lack applicability to other perceptual processes [4]. Here, we present a model based on the hierarchical random feature conceptor architecture proposed by Jaeger [12]. Hierarchical random feature conceptors have successfully been applied to denoising tasks, which presents a core function of general perception. Conceptors in general have been proposed as a solution to the neuro-symbolic integration problem by implementing a filter mechanism on the hidden state dynamics of echo state networks [11]. A conceptor is inserted in the state update rule of the echo state network. It suppresses activity in atypical directions in the network dynamics while activity in typical directions remains unaffected. Typicality of the directions can in this setting be determined by observed activity during training under the same input pattern.

2 Experimental Setup

In order to simulate the condition of bistable perception, we utilized the hierarchical random feature conceptor network, as it was presented in “Controlling recurrent neural networks by conceptors”, Chap. 3.16, by Jaeger in 2014. The network consists of three identical echo state networks arranged in a hierarchy of three levels. This echo state network was chosen to consist of 100 neurons with a feature space consisting of 700 neurons.

2.1 Learning of Prototype Patterns

In preparation to the experimental condition, the echo state network is presented with the clean signals of two sine waves with periods 13.190045 and 4.8342522

sampled at integer t, henceforth referred to as sine 1 and sine 2. After the prototype conceptors were learned for the clean signals, a bistable signal composed out of a superposition of these signals and noise is presented to the network. For each of the sine waves, the system is run through three periods:

1. For a washout period of 200 timesteps, during which the networks response starts to be correlated with the driver, no network responses are collected. Then the system is run in the conceptor adaptation mode for 2000 timesteps, wherein the prototype conceptor for that pattern is learned. Finally the system is run for 600 timesteps with the adapted conceptor in the network state update loop, and the network's response is collected.
2. In the following, two learning steps are performed.
   (a) The output weights W_{out} are computed by ridge regression with all collected reservoir states as arguments and the corresponding prototype patterns as targets. The normalized root mean squared deviation (NRMSD) between the output of the system, utilizing the calculated output weights, and the prototype pattern is computed.
   (b) In the second learning step, the loading, an input simulation matrix D is obtained. This is done by ridge regression, with the objective to reproduce the same network activations as they were elicited by the driver, but in absence of the driver.
3. Subsequently, the success of the learning steps was tested by a recall period. For every pattern the trained system was run under the respective conceptor for 200 washout steps. This allows for the adaptation of the network dynamics to the control of the current conceptor. Afterwards the output of the system was collected for 200 timesteps and compared to the original prototype pattern.

This describes the setup of one module of the random feature conceptor architecture with two sine waves of different periods learned. In the hierarchical random feature conceptor system, three layers of this architecture are bidirectionally connected. Weight matrices are learned beforehand and are then shared between layers. Conception weights and inputs evolve independently for each layer.

Bottom-up and top-down processing is mediated by trust variables that are adapted based on discrepancies between predicted and actual input in each layer. The input to each layer is a mixture of predicted input and the signal from the lower layer, mediated by the trust variable.

There exists a bottom-up as well as a top-down flow of information. The output of the lower layer is fed to the higher layer, constituting the bottom-up flow. The top-down flow is the influence of conception weights of a higher module on a lower one. Both are mediated by the trust variables.

The top-down pathway influences the conception weights in each layer of the hierarchy. The top-layer hypothesis is passed downwards in the hierarchy. In each of the lower levels 2 and 1 an autoconceptor adaptation process is taking place, yielding layer internal conception weight vectors. These are linearly mixed with

the conception weight vector from the next higher layer, using the trust variables as mixing coefficients. The topmost layer is a special case, as there is no layer above which can have any influence on its conceptor. By design its conceptor is constrained to be a disjunction of prototype conceptors of the two sine waves.

The bottom-up pathway influences the input to the higher levels 2 and 3. These levels have a self generated input simulation signal. Additionally to this, they receive the output from the next lower layer. Again, the trust variables determine how much influence the bottom-up pathway has against the self generated input simulation signal by serving as mixing coefficients.

2.2 Experiments with the Bistable Stimulus

In addition to the hierarchical random feature architecture we introduced a feedback loop from the top level hypothesis to the input of the system. This feedback loop suppresses those parts of the input signal that can be explained or predicted under the current hypothesis of the system.

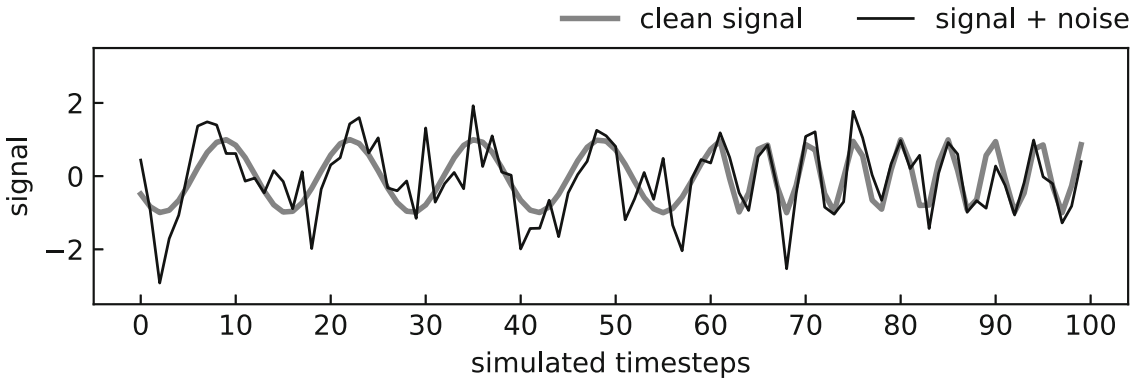


Fig. 1. A sample of the effective, ambiguous input, with influence from the feedback loop. Up to timepoint 62 the signal consists of sine wave 1 and noise, thereafter of sine wave 2 plus noise. The hypothesis that sine wave 2 is the source in the signal was winning until timestep 62. Therefore the signal of sine wave 1, which is not predictable under this hypothesis, remained in the input signal. From timestep 62 on the same reasoning holds, with hypothesis 1 being the winning hypothesis and sine wave 2 remaining as unpredicted residuum in the input signal.

The input to the hierarchical architecture at the lowest level is the sum of the two irrational sine patterns and normally distributed noise, with the signal to noise ratio of 1 with respect to the clean sine wave input. The noise was found to be necessary to push the system into an oscillating regime [2]. When the system settles on a hypothesis on the highest level of the hierarchy, the part of the input signal that can be explained by that hypothesis is subtracted from the input to the lowest level of the hierarchy. Importantly, we defined the winning hypothesis by the procedure of 'the winner takes it all'. This affects the input drastically as the complete clear signal that belongs to the winning hypothesis is

subtracted from the input. Thereby the effective input to the system is usually a composition of noise and one signal source. A sample of this effective input is shown in Fig. 1.

The system is run for 50.000 timesteps. Over the course of this simulation the hypothesis of the system about the source of the driver is collected on all three levels of the hierarchy. Moreover the dynamics of the trust variables that operate between the levels are saved. Experiments were recorded using the `Sacred` library [9] to ensure reproducibility.

3 Results

Initial learning of the prototype conceptors was performed successfully. The NRMSD for computing the output weights was 0.0027. The NRMSD per neuron between the input driven network response and the network response elicited by D was 0.0005 on average per neuron. Both learned sine wave patterns were recalled. After the correction of an inevitable phase shift, the NRMSD for the sine 1 was 0.025 and the NRMSD for sine 2 was 0.059.

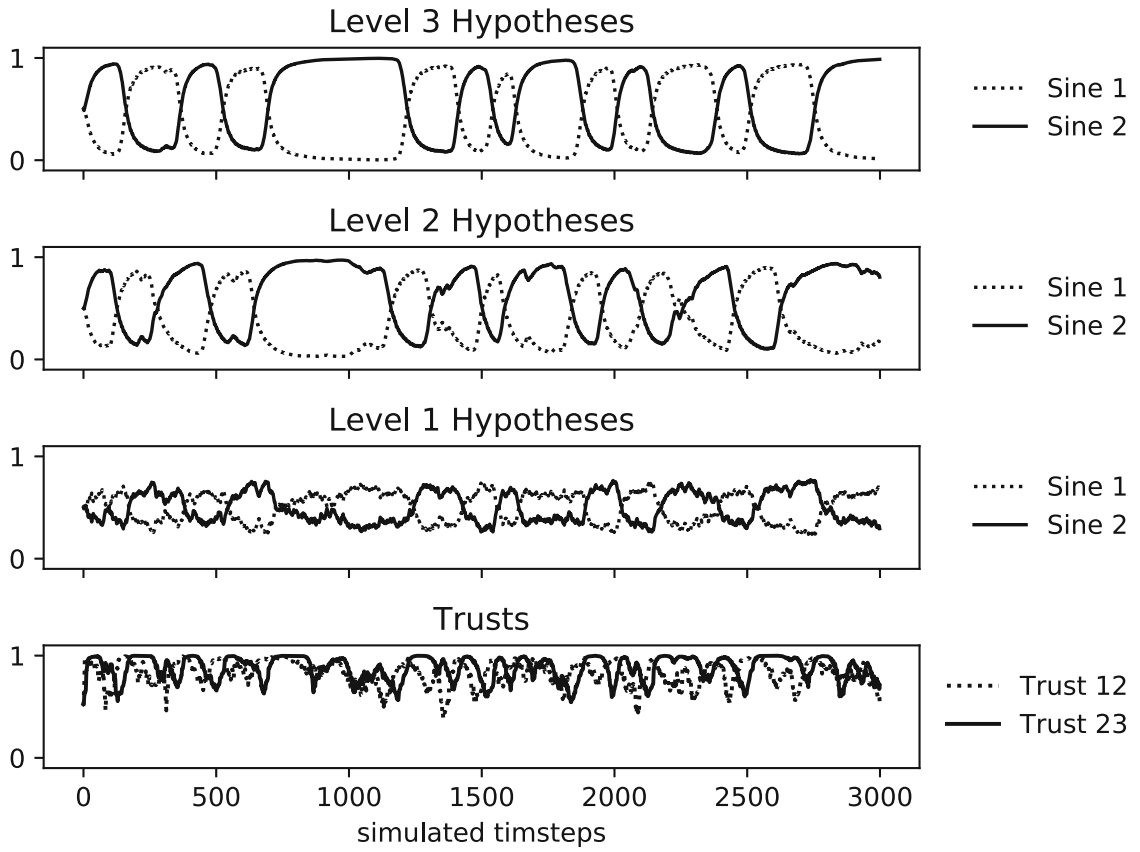


Fig. 2. Developments of hypotheses and trusts in the network. Displayed are the first 3000 simulated timesteps. The three topmost plots show the evolution the of hypothesis vectors for the three levels of the hierarchy. The bottom-most plot shows the trust variables operating at the intersection of the levels of the hierarchy. For details see the Results section.

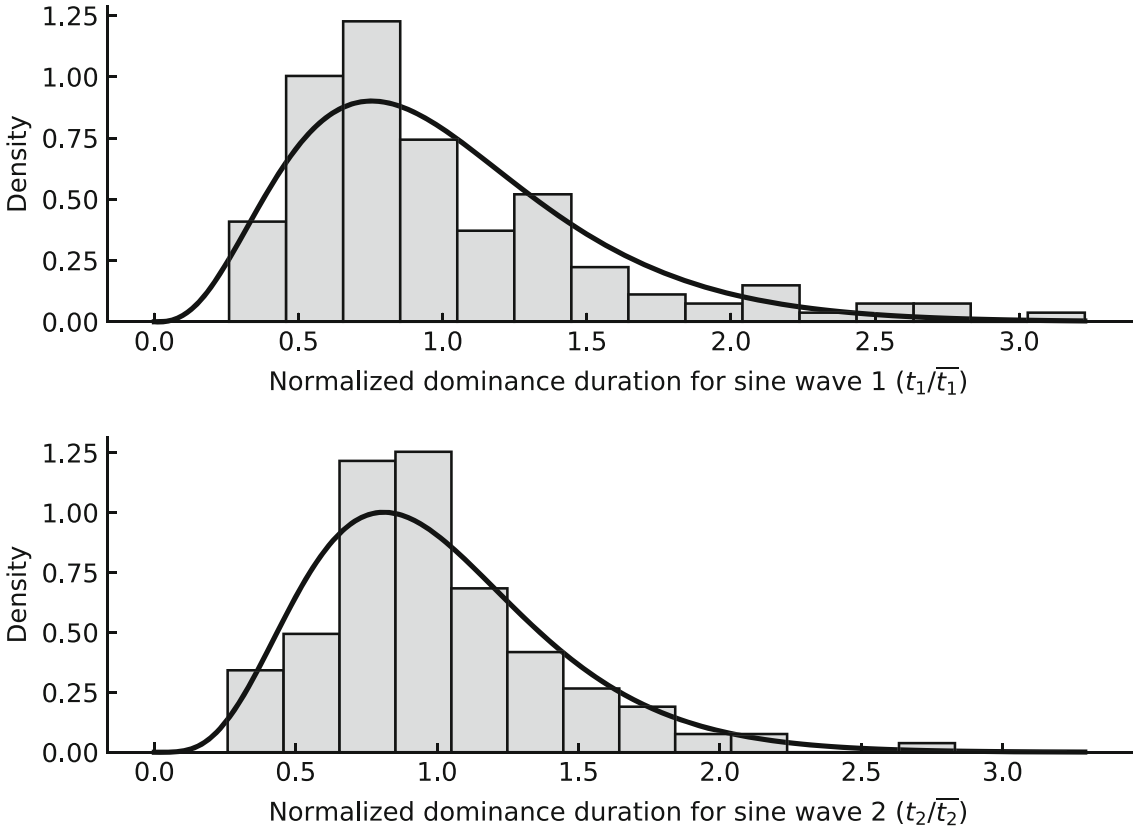


Fig. 3. Distribution of dominance times, separately for each sine wave. Both histograms were fit to a gamma distribution function (black line). The distribution of dominance times in the simulation is similar to data acquired from experiments in humans, when they were viewing rivalling stimuli.

Figure 2 shows the results of then presenting the combined signal for the first 3000 out of the total of 50000 simulated timesteps. A few observations can be made: On level 1 the hypotheses are not yet really differentiated with relatively long periods where both hypotheses are almost equally likely. On level 2 this differentiation is far better, surpassed only by a little in layer 3. Moreover, a small delay in the processing of the system can be observed. Comparing level 2 and level 3 hypotheses, it can be seen that level 3 reacts similar but has a delay on the order of 100 to 300 timesteps with regard to level 2. Between level 1 and level 2 this is less obvious, but can also be observed. It is also far more difficult to see, because on level 1 the structure of the hypothesis peaks is still very different compared to the higher levels. Most importantly an oscillation between the hypotheses can be observed on all levels. The top level hypothesis vector can be interpreted as the perception of the system, switching from one sine wave to the other, back to the first one, and so on. This resembles the perception human observers have when they are viewing rivalling stimuli. The bottommost plot displays the trust variables that operate between the levels. They both stay at a high level during the stimulation, indicating that the system is confident to generate the correct pattern most of the time. Especially for the trust variable between level 1 and 2 several small dips can be observed. These can correspond

to a switch in the input signal due to a change in hypothesis on the top level. The system realizes that its prediction does not match the input pattern as much as it would, if it were to change its hypothesis and conceptor. It therefore operates shortly in an input driven manner to find the optimal input matching hypothesis and settles again, only to be tempted to change again as soon as the new hypothesis affects its input.

We calculated the distribution of dominance times on the data of the third level hypothesis vector. We measured normalized dominance times in terms of simulated time steps t, dividing all dominance times by the mean dominance time $\bar{t}$. We in particular calculated the normalized dominance time distribution for each sine wave separately, to not get a mixed distribution that is skewed to either side because the patterns have different signal strengths. Both distributions are plotted in Fig. 3. The distributions show similarity to the results of Levelt [13]. As Levelt did, we fit a gamma distribution to the data. The gamma pdf can be parameterized with shape k and scale θ as

$$f(t;k,\theta) = \frac{t^{k-1}e^{-\frac{t}{\theta}}}{\theta^k \Gamma(k)}$$

We estimated the parameters k and scale θ of the distribution, which yields the following equation of the fit for sine 1

$$f(t;3.179,0.315) = \frac{t^{3.179-1}e^{-\frac{t}{0.315}}}{0.315^{3.179}\Gamma(3.179)}$$

and for sine 2 respectively

$$f(t;7.237,0.138) = \frac{t^{7.237-1}e^{-\frac{t}{0.138}}}{0.138^{7.237}\Gamma(7.237)}$$

3.1 Stability

We find the phenomenon of bistable perception in our architecture to occur across different input patterns. We tested our architecture with different combination of sine waves and a combination of random periodic patterns and sine waves. We found the hypotheses at the highest level to consistently switch between the two presented patterns for all tested pattern combinations. We also found the distributions of dominance times to consistently form gamma-like right-skewed distributions, that resemble the distributions observed in psychophysics experiments.

However, the exhibition of the phenomenon is dependant on the right interplay between input pattern, aperture and initialization.

4 Discussion

We used the hierarchical random feature conceptor as a model for bistable perception, thereby providing an application of the conceptor architecture in a cognitive modeling task.

4.1 Comparing Dominance Times to Levelt's Work

The distribution of dominance times that we obtained from the simulation is remarkably similar to the dominance time distribution of Levelt's work in the 60 s, which is shown in Fig. 4. Across different combinations of input pattern, we especially find that normalized dominance times concentrate in the range between zero and three. Also, we consistently observe a right skewed distribution of dominance time durations, which is a well established result in research on bistable perception across modalities. These results encourage further investigations in how far the hierarchical random feature conceptor architecture is a suitable model for general human perception.

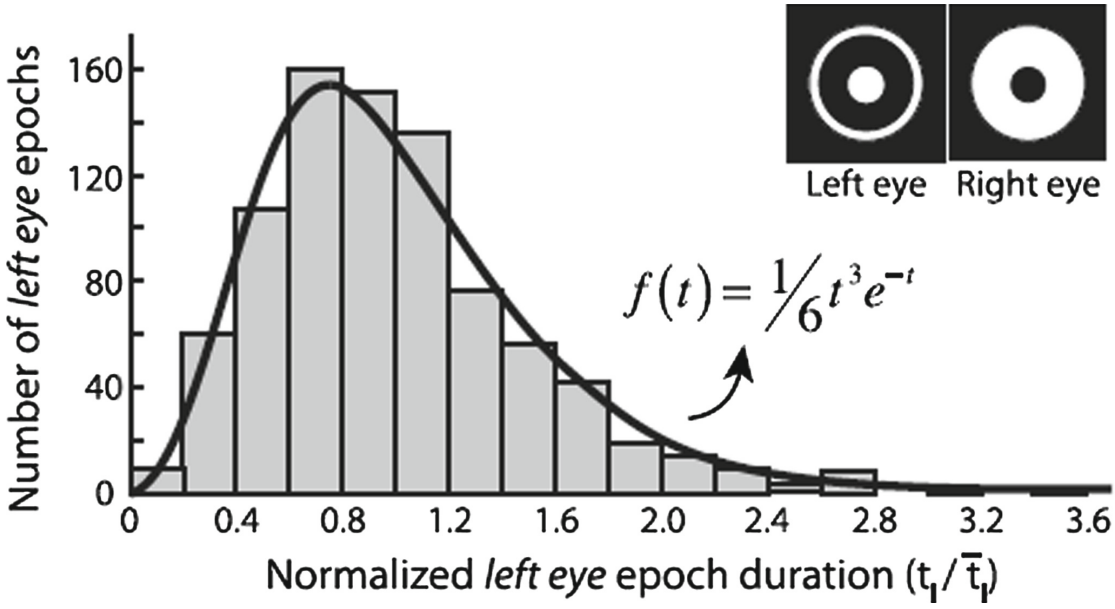


Fig. 4. Distribution of dominance times for binocular rivalry as reported by Levelt [13]. This figure is reproduced from Brascamp et al. [4].

4.2 Bistable Perception

In our simulation the system has learned two prototype patterns. These two patterns are "the world" for the system. Besides the driving input itself, its internal representation of the prototype patterns is the only information it has access to during runtime. As the system is also not adapting or learning any new patterns during the course of the simulation, the only hypothesis it can make up involve the two prototype patterns. The simulation of the bistable perception shows that the system adapts its hypothesis about the current input in accordance to the input. On the level of the hypotheses it shows an alternating behaviour, just as it is the key observation in bistable perception in humans. Insofar we have a working example of a challenging situation for a perceptual system. The system has only the option to make up hypotheses from the two prototype pattern it knows. It can, however, settle on a mixture of these, maintaining for example the

hypothesis that a mixture of the prototype patterns causes the current sensory input. This is in fact the case, if the system is run without the effect of the feedback loop. In many situations this is highly desirable and it is a research project in its own right in how far conceptor combinations really are able to combine concepts. Nevertheless in the special case of humans viewing rivalling stimuli, the hypothesis of a mixture of both stimuli is a priori highly unlikely. For the concrete example of binocular rivalry, face-house compounds do usually not appear in the world. We reflect this low prior probability for the compound hypothesis by subtracting the winning hypothesis from the input. This design choice is supported by a strong effect of the prediction of the system on the actual perception. The predicted signal is completely explained and therefore can be subtracted from the input signal. In the original approach we tried to take the bare prediction of the system on the top layer and subtract that from the input. This turned out to be not suitable for our attempt, as reservoir systems as we use them produce inevitable phase shifts of the generated signal versus the input signal. Moreover we faced the above mentioned problem of the system believing that the current input is a mix of both signals.

4.3 Relation to Predictive Error Minimization

Hohwy, Roepstorff and Friston [10] utilize the predictive error minimization theory (PEM) to explain the phenomenology of a specific instance of bistable perception, namely binocular rivalry. Here, we present their analysis of the binocular rivalry condition in terms of the predictive error minimization scheme and relate it to our model.

According to PEM theory, the brain tries to find the best matching hypothesis that could be the cause for the observed data. When the human brain is exposed to a binocular rivalry condition with a picture of a house and a picture of a face presented to separate eyes at the same time the brain of a subject might settle on the hypothesis that a house caused the visual stimulation. Under this hypothesis, the brain, as a hierarchical generative model, would predict some features of a house which will match with parts of the sensory data. The sensory drive that is generated by the face would remain as a residuum and as a prediction error that is not accounted for by the prediction of the brain. This error is on about the same order of magnitude as the explained data, namely the part of the stimulus that belongs to the house. Due to this balance of information content between both parts of the stimulus and due to noise, the hypothesis that the face generated the sensory drive would overtake. This oscillation describes the alternation between different percepts that is observed when humans view rivalling stimuli.

The hierarchical random feature architecture tries to minimize prediction error by selecting the best hypothesis in order to predict the incoming sensory data. The residuum of the incoming data which can not be explained is called the prediction error. In contrast to PEM theory, our proposed architecture does not signal the prediction error upwards in the hierarchy, but a denoised version of the sensory input. 'Denoised' means in this context that parts of the signal which are not predictable under the current hypothesis are regarded as noise and

are suppressed. This, in fact, leads to less prediction error on higher layers of the hierarchy, as the prediction error is suppressed by each layer. This mechanism is therefore actually minimizing prediction error, but in a slightly different fashion than the usually in predictive coding proposed upwards signalling of the residual signals or prediction errors. Minimizing prediction error just by suppressing all signals that can not be predicted on its own does not seem very useful. But this process is aided by a general assessment of fit of all prototype patterns to the input signal. This is inherent in the conceptor mechanism. Therefore the mechanism for prediction error minimization is different in the hierarchical random feature conceptor as compared to the usual notion in predictive coding. This issue is still in debate, also for the predictive coding research community, as we are not aware of any clear cut evidence in favour of and against other possible realizations of error signalling.

5 Conclusion

We implemented a recurrent generative model based on hierarchical conceptors to investigate its behaviour with regards to bistable perception. We were able to show that the network exhibits random switches in its percepts. The distribution of dominance durations furthermore resemble well established findings from psychophysical experiments on bistable perception in humans. Moreover, the hierarchical organization and the message passing between the levels of the hierarchy allows for noise suppression and prediction error minimization. Therefore, we were able to construct an accurate model for bistable perception that is based on a model for general perception and is applicable to other tasks. Overall, we conclude that the hierarchical random feature conceptor architecture is a promising model for general human perception. Further work has to be done in order to investigate whether the architecture can account for more perceptual phenomena.

References

1. Alais, D., Blake, R.: Binocular Rivalry. MIT Press, Cambridge (2005)
2. Brascamp, J.W., Van Ee, R., Noest, A.J., Jacobs, R.H., van den Berg, A.V.: The time course of binocular rivalry reveals a fundamental role of noise. J. Vis. **6**(11), 8 (2006)
3. Brascamp, J.W., Van Ee, R., Pestman, W.R., Van Den Berg, A.V.: Distributions of alternation rates in various forms of bistable perception. J. Vis. **5**(4), 1 (2005)
4. Brascamp, J., Klink, P., Levelt, W.J.: The 'laws' of binocular rivalry: 50 years of levelt's propositions. Vis. Res. **109**, 20–37 (2015)
5. Carter, O., Konkle, T., Wang, Q., Hayward, V., Moore, C.: Tactile rivalry demonstrated with an ambiguous apparent-motion quartet. Curr. Biol. **18**(14), 1050–1054 (2008)
6. Dayan, P.: A hierarchical model of binocular rivalry. Neural Comput. **10**(5), 1119–1135 (1998)
7. Deutsch, D.: An auditory illusion. Nature **251**(5473), 307 (1974)

8. Freeman, A.W.: A multi-stage model for binocular rivalry. J. Neurophysiol. **94**, 4412–4420 (2005)
9. Greff, K., Klein, A., Chovanec, M., Hutter, F., Schmidhuber, J.: The Sacred infrastructure for computational research. In: Proceedings of the 16th Python in Science Conference, pp. 49–56. SciPy (2017)
10. Hohwy, J., Roepstorff, A., Friston, K.: Predictive coding explains binocular rivalry: an epistemological review. Cognition **108**(3), 687–701 (2008)
11. Jaeger, H.: The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. Bonn, Germany: German National Research Center for Information Technology GMD Technical report **148**, 34 (2001)
12. Jaeger, H.: Controlling recurrent neural networks by conceptors. arXiv preprint arXiv:1403.3369 (2014)
13. Levelt, W.J.: On binocular rivalry. Ph.D. thesis, Van Gorcum Assen (1965)
14. Necker, L.A.: LXI. observations on some remarkable optical phænomena seen in switzerland; and on an optical phænomenon which occurs on viewing a figure of a crystal or geometrical solid. The London, Edinburgh, and Dublin Philos. Mag. J. Sci. **1**(5), 329–337 (1832)
15. Wheatstone, C.: Contributions to the physiology of vision.-part the first. on some remarkable, and hitherto unobserved, phenomena of binocular vision. Philos. Trans. Roy. Soc. Lond. **128**, 371–394 (1838)
16. Zhou, W., Chen, D.: Binaral rivalry between the nostrils and in the cortex. Curr. Biol. **19**(18), 1561–1565 (2009)

The Neocortex

Edited by Singer, Wolf, Terrence J. Sejnowski, and Pasko Rakic

MITCogNet
From the MIT Press

Series Editor: J. R. Lupp
Editorial Assistance: M. Turner, A. Ducey-Gessner, C. Stephen
Photographs: N. Miguletz
Lektorat: BerlinScienceWorks

The book was set in TimesNewRoman and Arial.
Printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data

Names: Singer, W. (Wolf), editor.
Title: The neocortex / edited by Wolf Singer, Terrence J. Sejnowski, and Pasko Rakic.
Description: Cambridge, MA : The MIT Press, [2019] | Series: Strüngmann forum reports | Includes bibliographical references and index.
Identifiers: LCCN 2019019582 | ISBN 9780262043243 (hardcover : alk. paper)
Subjects: LCSH: Neocortex.
Classification: LCC QP383.12 .N44 2019 | DDC 612.8/25--dc23 LC record available at https://lccn.loc.gov/2019019582

10 9 8 7 6 5 4 3 2 1

11

Computational Elements of Circuits

Johannes Leugering, Pascal Nieters, and Gordon Pipa

Abstract

Information processing in the brain is implemented across several temporal and spatial scales by populations of neurons. This chapter addresses how single neurons, small network motifs, and larger networks, in which emergent dynamics are largely shaped by the connectivity of the system, contribute to this processing of information. Computation is defined as a semantic mapping; that is, it is the process by which representations of external (e.g., stimulus-driven) or internal (e.g., memories) information change. A feature specific to neuronal computation is that mappings are mostly local, constrained by connectivity patterns between neurons. This implies that complex mappings from local information onto representations that are highly relational and abstracted, and which rely on information between distant parts of the system, require mechanisms that can bridge, bind, and integrate pieces of information across large scales. An overview of this process in the nervous system is delineated: Local information processing is described at the level of individual neurons and small motifs. Emergent phenomena are addressed that implement information processing across large recurrent neuronal populations. Finally, an omnipresent but mostly ignored feature of neuronal systems, delay-coupled computation, is described.

Information Processing in Single Neurons and Populations

An understanding of how information is processed in neural systems begins with a consideration of how an individual neuron perceives and processes information, before extending this scope gradually to larger systems. Our goal in this chapter is to present a concise, abstract view of computation in neural systems, understood to be key to a meaningful change in the representation of information. In the interest of brevity, the biological complexity of neurons and networks (e.g., the role of specific ion channels or the potential influence of glia cells and neuromodulators) will not per se be addressed.

A Stochastic Process Linear-Nonlinear Neuron Model

From the perspective of the linear-nonlinear (LN) model, a neuron is a computational unit that receives a multivariate time-varying input signal through its synaptic inputs and generates a univariate time-varying output signal. This mapping from input to output signals is near instantaneous (at least time-invariant), as the neuron itself is assumed to have, at most, a very limited internal memory[1] and be subject to noise.

In the mathematical framework of stochastic processes, a neuron can thus be concisely described as a nonlinear, causal, time-invariant operator that maps a multivariate stochastic process onto a univariate stochastic process. We make several simplifying assumptions that result in a convenient class of neuron models (Ostojic and Brunel 2011; see also Figure 11.1):

- The neuron's operation can be modeled as a leaky integrator or, even simpler, an instantaneous input-output mapping.
- It is composed of a linear operator, which reduces the multivariate input arriving at different synapses along the dendritic tree to a univariate input to the neuron's soma, followed by a nonlinear transformation.
- The linear operation is parameterized by synaptic weights, which can be positive or negative.
- The nonlinear transformation, which we refer to as the activation function or just nonlinearity, is a monotonically increasing, (locally) differentiable and bounded function.

While the activation could be further used in a spike generation process as an instantaneous firing rate, we treat it here as the neuron's continuous state or output. Each neuron in a population independently processes its own input (which may be correlated to other neurons' inputs), and its state provides one component of the entire population's multivariate state. The computation carried out by a population of neurons, mapping a multivariate input signal onto a multivariate state, must thus arise component-wise from the computations realized in the individual neurons. Each neuron, however, is limited to those operations which can be realized by a LN model under the above constraints.

To better understand the capabilities and limitations of this class of models, it helps to analyze them from a machine learning perspective, where such models commonly appear under different guises and names.

[1] The exception to this rule is found in plasticity mechanisms, which we assume to operate on a much slower, separate timescale than that of the output signal, and thus they can be treated as virtually constant in this context. The commonly made assumption of near instantaneous operation of the neuron further presumes that slower active dendritic processes do not substantially contribute to computation, which can be called into question and may turn out to be an overly simplistic perspective.

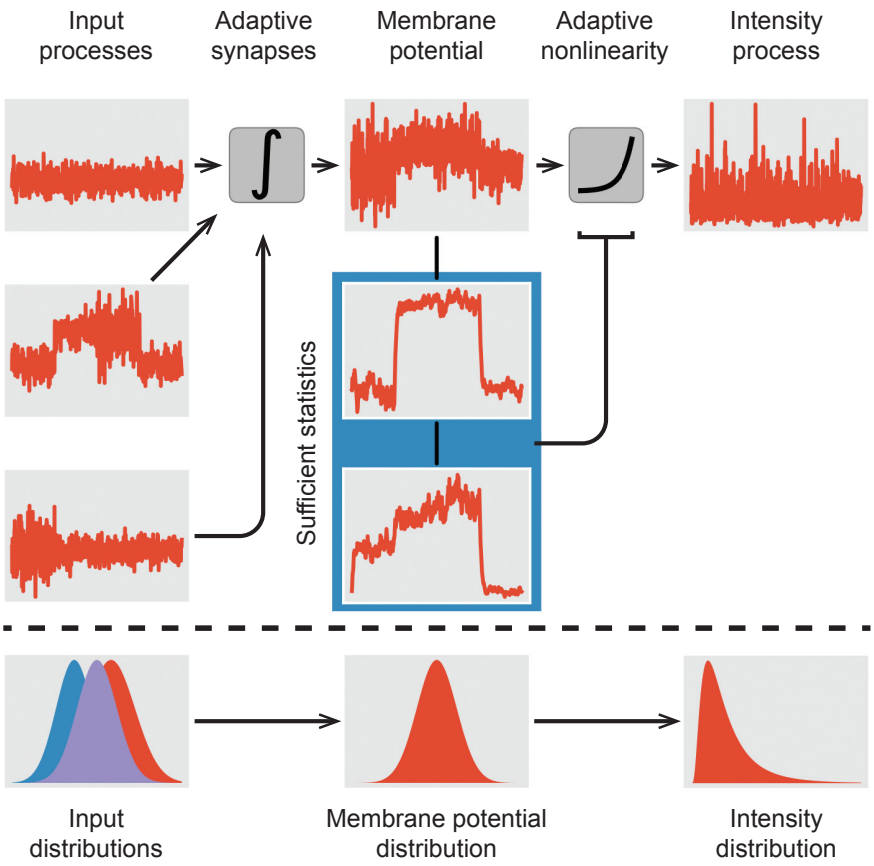


Figure 11.1 A model neuron receives a linear combination of multiple time-varying stochastic processes that are scaled by adaptive, synaptic weights and integrated into the neuron's membrane potential. By sensing some sufficient statistics of the membrane potential, the neuron's nonlinearity can be adjusted to achieve an activation (or intensity) with desirable statistical properties. Assuming stationarity of the input processes, the neuron's nonlinearity can be determined by the desired mapping from the univariate membrane potential distribution to an intensity distribution. Adapted from Leugering and Pipa (2018).

LN Models in Machine Learning

Using a Heaviside function for the nonlinearity, LN models appear in machine learning in the form of linear hard-margin classifiers, such as the classical perceptron (Rosenblatt 1958), linear support vector machines (Hearst et al. 1998), or depth-one decision trees (so-called "decision stumps"; Criminisi et al. 2012). With continuous nonlinearities, such as the logistic function, these models can be used as soft-margin classifiers and regressors, as in generalized linear models (GLMs) (McCullagh and Nelder 1989), where the nonlinearity is used to relate a linear combination of input features to the expected value of the (task-specific) label associated with the data. To improve performance, multiple instances of such models can be combined laterally to form an ensemble, used in a boosting procedure or stacked hierarchically, like the layers of an artificial neural network (Hopfield 1988) or the levels of a decision tree.

Computation in this context simply refers to the ability of the model to encode specific task-relevant information about its inputs into its output. The same claim has been made for individual biological neurons, as well as whole layers of neurons in deep networks under the "information bottleneck" principle. The deceptively simple argument is that each neuron (or each layer of a network, respectively) is presented with a high-dimensional input signal that carries task-relevant, as well as irrelevant, information, and, in a noisy environment with limited capacity to transmit information, ought to transform it into an informative low-dimensional output signal (Becker 1996).

Supervised Learning

In a supervised setting, where the desired output of the model is known at all times, the extraction and transmission of task-relevant information with simultaneous suppression of task-irrelevant "noise" represents a form of lossy compression. In multilayer networks, backpropagation can provide a supervised error signal for each layer and ultimately each neuron, thus allowing it to locally solve a lossy compression problem, which has been hypothesized as the theoretical mechanism underlying the surprising success of deep neural networks (Shwartz-Ziv and Tishby 2017).

Unsupervised Learning

The concept and potential mechanisms of error backpropagation in biological neural networks, however, are controversial, and the existence of supervised target signals may be called into question altogether. In the absence of supervision, the information bottleneck principle can be restated as the objective for each neuron to simply encode its inputs into its output in the most informative way possible, since it cannot distinguish task-relevant from irrelevant information.

The information encoding of the output signal is reflected in the firing statistics, with heavy-tailed firing rate distributions corresponding to sparse spiking codes, and narrowly peaked distributions corresponding to tonic firing or bursting codes. By driving synaptic plasticity, this can, in turn, shape the topology of synaptic connections and lead to the formation of specific motifs, thus allowing a population of neurons to implement task-relevant computation without supervision.

Under the biological constraints imposed on the neuron (e.g., bounded firing rates, energy limitations), the mutual information between input and output is bounded by the entropy attainable by the output distribution. A common objective is thus for the neuron to enforce a maximum entropy distribution of its outputs by appropriately adjusting its nonlinearity, while simultaneously tuning its synaptic connection weights to project the multidimensional input signal onto the most informative subspace. Equivalently, for a population of

neurons, the information bottleneck objective is to realize a maximum entropy joint distribution, such that each marginal distribution of an individual neuron's output satisfies the biological constraints.

Unsupervised Learning Application: Independent Component Analysis

As it turns out, this objective fully determines a unique optimal choice of nonlinearity for a given family of input distributions and a desired output distribution. It also implies that the linear subspaces selected by the neurons' respective synaptic input weights should correspond to the main independent components. Consequently, this problem is also referred to as independent component analysis (ICA), a generalization of principle component analysis which can no longer be solved by linear methods (Hyvärinen and Oja 1998; Triesch 2007).

This intuition transfers seamlessly to a framework of stochastic processes (Leugering and Pipa 2018), where a population is tasked with mapping its (stationary) multivariate input process onto a multivariate output process, with a joint distribution composed of independent components with given marginal distributions. By factoring the population's joint distribution into its marginal distributions and a copula function, it becomes apparent that this objective can be achieved through the interaction of two distinct mechanisms:

1. The copula function captures all of the dependency structure present in the joint distribution and depends only on the choice of synaptic input weights of the population; thus it can be adjusted by synaptic plasticity.
2. The marginal distribution of each neuron's output can be enforced purely by an appropriate choice of nonlinearity; thus it can be adjusted by intrinsic plasticity.

Since all of the information required to solve the ICA problem is available locally to the neurons or their synapses, it can be solved by the LN model discussed above using simple, biologically plausible mechanisms of intrinsic and synaptic plasticity in a time-continuous, noisy setting.

Using motifs of several laterally inhibiting neurons, different independent components can be found, leading to a highly informative, multivariate output signal. As shown in Figure 11.2, such a structure can be used to learn, in an unsupervised fashion, to classify MNIST images with just a handful of neurons. For an in-depth discussion of this result, see Leugering and Pipa (2018).

Computation in Networks Using Emergent Properties

The cerebral cortex is a highly distributed system with reciprocal connections that shape neuronal activity through self-organizing and that can create

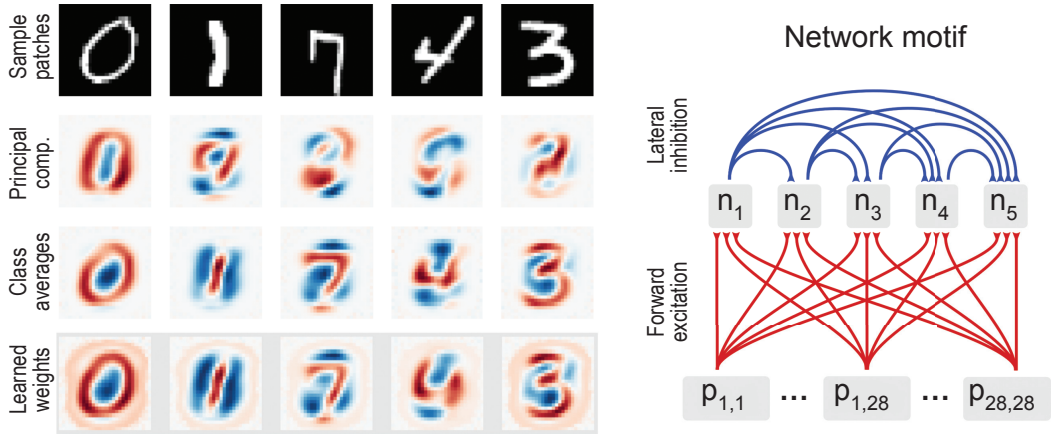


Figure 11.2 A small motif of five neurons receives feedforward excitation from 28 × 28 neurons, representing pixels of visual inputs. Images from the MNIST database are presented successively, while the synaptic weights and each neuron's nonlinearity are adjusted by local synaptic and intrinsic plasticity, respectively. Only the combination of both learning mechanisms leads to the (unsupervised) discovery of independent components in the input space, corresponding to the average input for each class. Lateral inhibition learns to decorrelate the neurons and ensures that different components are discovered, reducing redundancy and thus maximizing the information content of the motif's output. Results from Leugering and Pipa (2018).

coherent states able to encode representations of sensory objects, decisions, and programs for motor acts (Uhlhaas et al. 2009). The topology of the connectivity shares properties with small world networks having no singular center where all information converges (Gerhard et al. 2011). This raises questions of how the numerous computations on the level of single neurons are coordinated and bound together to give rise to coherent percepts and actions, and how relations between simultaneously represented contents can be encoded. One option is that neuronal synchrony can implement both features. Several mechanisms have been discussed—for instance mediated by inhibitory synapses, enhanced via gap junctions, induced by motifs of neuronal connectivity (Vicente et al. 2008; Pérez et al. 2011; Messé et al. 2018)—that can induce neuronal synchronous firing even despite long conduction delays. However, one of the central challenges that has not been sufficiently addressed is that the mechanism needs to enable the neurons to synchronize and desynchronize in a stimulus-specific fashion, and thereby to encode relationships. Noise-induced coherence is one such mechanism that was recently demonstrated to produce fast, stimulus-specific, and biologically plausible synchronization patterns.

First discussed in complex and excitable systems (Pikovsky and Kurths 1997), noise-induced coherence is a process that can structure and synchronize the activity of the system based on noisy or even unstructured input. The nature of noise-induced coherence is that the complex system (the dynamical elements, e.g., neurons, together with the network topology) defines patterns that exhibit enhanced coherence if the system is driven by a corresponding motif, neuron-specific optimal amplitude of unstructured noise. In other words, and in

respect to neuronal networks, noise translates a pattern of neuron-specific firing rates into patterns of coherent and synchronized population responses (i.e., translation of a neuron-specific rate code to a population-based sync code). Importantly, this translation is network specific, which opens the possibility that the expression of synchronous events is not only driven by the stimulus-specific rate pattern but also by the network, and its structure is shaped by neuronal plasticity.

Transformation of Spike Rate Coding to Coherent Population Codes via Noise-Induced Coherence

To illustrate the mechanism and encoding based on noise-induced coherence, let us consider an example for the visual cortex V1. In general, it is known that network structure of cortical networks is at least partially shaped by the experience of past activation mediated by neuronal plasticity. For V1, this implies that the connection strength horizontal connections in V1 reflect the aggregate statistics of natural visual scenes (Onat et al. 2013); that is, V1 cells with nearby receptive fields are preferentially connected, and specifically when they select for similar visual stimuli. Figure 11.3a shows a network simplified to such a V1 prototypic connectivity pattern. The system receives stimulus-specific input described by neuron-specific retinal coordinates which match their cortical position (retinotopy) and have a particular angle (orientation tuning) presynaptic spike rates (i.e., uncorrelated and rate-modulated Poisson firing). To illustrate the effect of noise-induced coherence, we use two kinds of stimuli: one that is open and composed of two shorts blocks, and one that is closed and composed of a longer bar. Given the retinotopic mapping, this implies that the activation pattern, in comparison to the underlying network, results in different shortest path lengths between stimulus-driven cells. Only few of these cells will have direct connections, since horizontal connections preferably connect cells with nearby receptive fields. More generally, the network connectivity implies a metric for possible stimulus patterns. Given this metric, for V1, the shortest path between any two responding cells will likely be longer, on average, for a scattered stimulus than for a more compact stimulus. As a result, the same cells which receive a presynaptic input pattern matching the connectivity of the network (here, cells that are part of a continuous patch) exhibit stronger noise-induced coherence than others. Such mechanisms can be generalized to more complex encoding schemes, depending on the connectivity patterns of the network. For example, the well-known orientation tuning of cells in V1, in combination, and network motifs described by preferred connectivity across cells with similar orientation will result in enhanced coherence of neurons that encode chains of shorter line segments (see Figure 11.4). In general, noise-induced coherence is a mechanism that can measure the similarity between the network connectivity and the stimulus-induced spike rate pattern (Korndörfer et

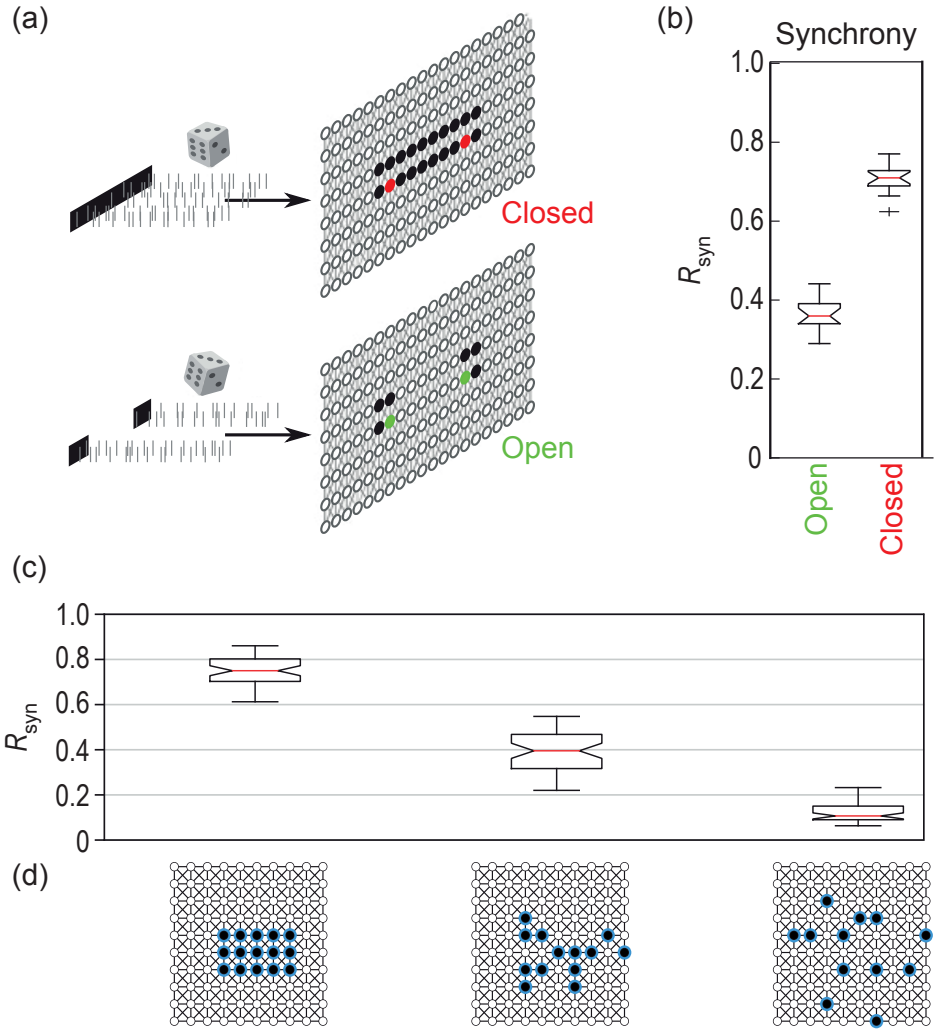


Figure 11.3 (a) Noise-induced coherence for two alternative presynaptic stimuli (red, a closed line; green, an open line segment). Coherence is measured between the green- or red-highlighted neurons. The only difference is the context given by the stimulus drive, which itself is composed of unstructured Poisson noise. The network topology is defined by nearest neighbor connection matches, and stimuli are matched using retinotopic mapping. (b) Synchrony measure of the pairs of neurons shown in (a) and for the two stimulus conditions. Coherence is higher for the compact closed line, since the shortest path length between stimulus-driven neurons is smaller for the closed contour. (d) This feature is generalized to the amount of scattering of a stimulus; that is, the greater the scattering, the larger the shortest path length between neurons, given the metric of the underlying network connectivity. The resulting stimulus-induced coherence (c) is the largest for the most compact, and the lowest, for the most scattered stimulus. Adapted from Korndörfer et al. (2017).

al. 2017). It can therefore be a measure of how well the stimulus matches a prior learned by neuronal plasticity and encoded in the network's connectivity. Here the stimulus-induced spike rate reflects a classical labeled line code. Thus, spike synchrony generated by noise-induced coherence carries synergistic information that reflects to which degree the current stimulus encoded by the spike rate is expected, given past stimulus experiences. Such a signal could be used early after input onset in a feedforward fashion, for instance,

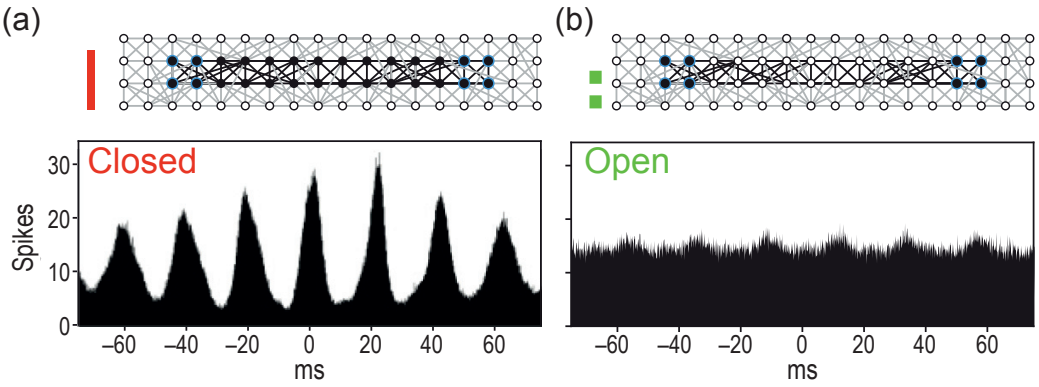


Figure 11.4 Average cross correlation of noise-induced coherence between two sets of neurons marked in blue for two different stimulus conditions: (a) closed line segment and (b) open line segment. The noise-induced coherence is stronger for the closed, compared to the open, condition. In the original publication (Korndörfer et al. 2017), it is shown that this increased coherence can be decoded as closed contour as early as a few spikes after stimulus onset (70 ms).

to guide attention toward stimuli composed of plausible parts. In contrast to many other types of synchronization, it also does not require, but can be improved by, inhibitory cells (Korndörfer et al. 2017), and it produces firing patterns that closely resemble *in vivo* recorded patterns (e.g., Gray et al. 1989; Uhlhaas et al. 2009).

Reservoir Computing

Most mechanisms discussed over the past decades for neuronal information processing require highly structured networks, specific types of dynamical processes, and very specific encoding schemes of information (e.g., rate code versus population spike codes). A frequently used feature of computational models is that they rely on attractor dynamics, which can be trained to implement specific computational features, such as associated memory in Hopfield networks (Hopfield 1982) or the winner-takes-all mechanism (Maass 2006) for decision making, for example.

Like the ICA network discussed above, all of these computational models implement a clearly defined information processing principle and rely on a very specific type of implementation, in terms of connectivity and dynamical elements. This is a strong advantage, since it allows us to study principle and well-defined behavior, and to reduce the computation to a minimal set of required properties. At the same time, this reductionism also renders the models biologically implausible, since biological systems are subject to noise on pretty much any property, such that neuronal networks are mostly random with some statistical preferences for certain motifs, and neurons are diverse in type and morphology.

Therefore, a strikingly different model for neuronal computation is *reservoir computing*, originally introduced as liquid state machines by Maass

et al. (2002) or echo-state networks by Jaeger and Haas (2004). In contrast to most other computational principles, the recurrent network of a reservoir computer can be unstructured and random. This surprising property results from the simple insight that the distance between random mappings of states is growing fast, with increasing dimensionality of the mapping. In other words, implementing a certain computation does not require a dedicated network with specific connectivity tailored for the given task but, in principle, only a random network that implements a sufficiently high-dimensional random mapping. In the field of machine learning, this is known as feature expansion or kernel machines (Schölkopf and Smola 2002). Further, reservoir computing makes explicit use of the recurrence of neuronal networks to maintain an echo (i.e., memory capacity) of past inputs. This echo is mediated by reverberating activity, generated by the recurrent connectivity. Together, feature expansion and memory of the system can render a reservoir computer a universal computer (Buonomano and Maass 2009). The only task-specific element in reservoir computing is a task-specific mapping that can be learned by supervised, semi-supervised (Toutounji and Pipa 2014), or reinforcement learning algorithms (Aswolinskiy and Pipa 2015).

The remarkable insight of reservoir computing is that random recurrent networks can implement, in principle, any kind of computation if the networks are sufficiently complex. From a biological point of view, this implies that initially unstructured networks can bootstrap themselves, based on neuronal plasticity, to improve performance. Importantly, it can operate initially even without any structure.

Computation in Delay-Coupled Systems

When describing computation in the nervous system from the perspective of abstract single neurons or recurrent networks which show emergent behavior as a collective, a simplifying assumption is often made: interactions between neurons are instantaneous and not delayed. This is simply because delays in differential equations complicate the analysis of such systems significantly, and deriving theoretical results is a lot harder.

In biophysical reality, however, the brain is a network of nodes and wires that must be subject to transmission delays. For instance, conduction delays of tens of milliseconds occur in axonal transmission of spikes (Ringo et al. 1994). Interspike intervals, indicative of the timescales on which neurons compute outputs, have been found on the same scale in the motor system (Calvin and Stevens 1968) or in retinal ganglion cells (Levine and Shefner 1977). It is thus clear that delays play a role in the dynamics and computational properties of neural networks. A long-established example, where this role is well understood, is audio processing: transmission delays on delay lines are used to

distinguish left ear input from right ear input, and interpolate the location of a sound source (London and Häusser 2005).

In cortical structures, network motifs or microcircuits have been found that circumvent transmission delays and lead to zero time lag synchronization (Vicente et al. 2008). On the other hand, in microcircuits where transmission delays are modeled, only very specific topologies allow for coherent spiking activity, which delays control phase differences between oscillatory neurons (Pérez et al. 2011). So, locally, transmission delays control phase transitions between in-phase and out-of-phase response, whereas, globally, axonal delays can stabilize coherent response-important phenomena in neural computation.

Even still, these examples only describe how delays can negatively impact behavior of microcircuits or stabilize existing behavior. Future work should investigate the degree to which the added complexity of delay-coupled systems can be exploited for computation.

A Single Node with Delayed Feedback

Stabilizing emergent phenomena may not be the only mechanism by which delays can aid computation in the nervous system. Instead, the benefit of delayed interactions can be illustrated theoretically by examining a single computational node with delayed feedback. This very simple setup is described by a delay differential equation:

$$dx(t) = f\left(x(t),\, x(t-\tau)\right)dt. \tag{11.1}$$

The equation can be solved by a trick known as the method of steps (Guo and Wu 2013), which is both intuitive and illustrative of the complexity of delayed interactions: Assume that the solution to Equation 11.1 on some interval, $[t_0 - \tau, t_0]$, is known and denote that solution ϕ_0. For the subsequent overlapping interval, $[t_0, t_0 + \tau]$, Equation 11.1 can then be rewritten as

$$dx(t) = f\left(x(t),\, \phi_0(t-\tau)\right)dt, \tag{11.2}$$

since for all $t \in [t_0, t_0 + \tau]$, it holds that $t - \tau \in [t_0 - \tau, t_0]$, where ϕ_0 is the solution. This is now an ordinary differential equation and can be solved using traditional methods. However, the starting condition for the new solution is now a tuple $(\phi_0, \phi_0(t_0))$ of a function, and the function is evaluated at t_0. Further, the solution on the interval $[t_0, t_0 + \tau]$ is again a function; let that function be ϕ_1. In the method of steps, this procedure is iterated with this new starting value for the next interval of length τ. In general, if $t_i = t_0 + i\tau$, then ϕ_i is the solution on the interval $[t_{i-1}, t_i]$.

Even though Equation 11.1 is a differential equation of a single, scalar variable, solving it involves mapping functions onto functions for each τ interval

or cycle (Figure 11.5a) and is therefore infinitely dimensional. Delay differential equations are a subclass of partial differential equations whose state is described by functions, instead of finite-dimensional state vectors.

By introducing one simple, delayed feedback to a dynamical system, we elevate the complexity from one to infinitely many dimensions. This complexity

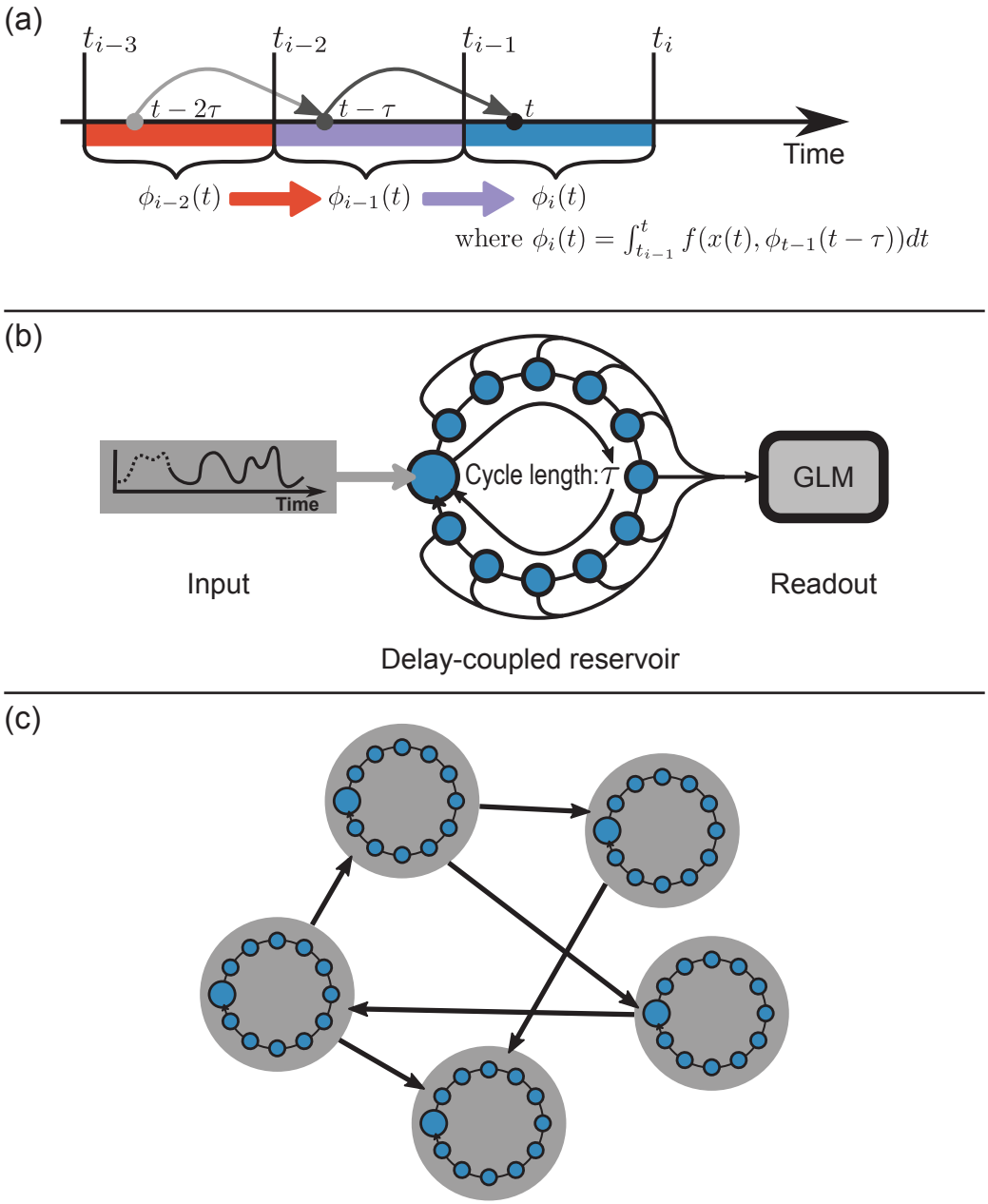


Figure 11.5 (a) Schema of solving delay differential equations with the method of steps. Functional solutions ϕ_i are mapped onto solutions ϕ_{i+1} via an integral over the original delay differential equations, where the delay dependency is replaced by a dependency on the last solution. (b) A delay-coupled reservoir utilizes the complexity of delay differential equations for computation by creating an input-driven dynamical system and feeding sampled activity during one τ cycle into a GLM readout trained to solve a specific task. (c) Networks of delay-coupled nodes can be understood as small recurrent systems inside a larger recurrent system. This model may be used to model the complexity of recurrence and delay coupling at the same time.

not only makes solving the mathematical problem more difficult, it also leads to dynamics that can be used for computation and inference.

The Delay-Coupled Reservoir

Introduced by Appeltant et al. (2011), the delay-coupled reservoir is a system described by a delay differential equation, such as Equation 11.1, but driven by an input $u(t)$:

$$dx(t) = -ax(t) + g\big(x(t-\tau), u(t)\big), \tag{11.3}$$

where g is a nonlinear function. The key insight from this work is that the activity in/of this simple one-node recurrent system can be sampled N-times during one delay cycle of length τ, and this sampled activity can be treated as the N-dimensional of a reservoir computer with N nodes. The input $u(t)$ is adapted to also change on the timescale of one τ-cycle, such that each τ-cycle associates one N-dimensional vector of activations with one input value. Following the reservoir computing procedure, this activation vector can then be used in a linear readout to learn a time-invariant, fading-memory function on the input (Figure 11.2).

The on-the-surface simplicity of delay differential equations leads to straightforward hardware implementations, where some nonlinear element is driven by input and self-coupled via a delay line. These simple building blocks have led to implementation based on standard electronic building blocks, but they also allow for the exploration of new computing devices, as in using delay-coupled lasers and photonics (Larger et al. 2012).

The hidden complexity of the system, however, allows it to be used in time-series forecasting, speech recognition, and even volatility prediction for financial markets (Appeltant et al. 2011; Grigoryeva et al. 2014).

This complexity, and the process of obtaining a vector of activity, can also be looked at theoretically using the method of steps. The iterative solution of the ordinary differential equation for subsequent τ-cycles, or intervals of length τ, can then be approximated analytically and written as a vector update equation for the N-relevant sample points directly (Schumacher et al. 2013). Thus, for computation within a reservoir computing setup, the infinite dimensionality of space of solutions to the delay differential equation reduces to N dimensions, a free parameter of the model. One can therefore profit from the potentially infinite dimensionality of a functional state. In practice, with a chosen decay rate α and a specific nonlinearity g, choosing arbitrarily large N does not benefit specific machine learning tasks above a task-dependent soft threshold. Nevertheless, researchers have seen benefits in expanding the dynamics of this simple, nonlinear, and delayed feedback-coupled node into an $N = 50$ up to an $N = 800$ dimensional state vector, as input to the linear regressor.

Instead of sampling the activity of the delay-coupled reservoir at N evenly spaced points, one can optimize the placement of these readout points. Here, it is useful to treat the N sampling points as a network of virtual nodes with a very particular connectivity structure: a lower diagonal exponential decay matrix. The distance from one sampling point, or virtual node, to the next can then be fine-tuned according to a homeostatic plasticity rule. It presupposes that good spatiotemporal computational performance is achieved when different virtual nodes are both sensitive to their inputs and as diverse as possible (Toutounji et al. 2015). The experiments in the study show that this rule does indeed lead to increased performance. From the point of view of the readout, the result permits a crude biological interpretation: a linear-nonlinear output neuron optimizes the locations along an axon, where it "reads" the activity of another neuron with complex time-dependent dynamics. Clearly, this interpretation is somewhat bold, but it highlights the potential of future research that uses delayed feedback models to encode and then decode information in temporal dynamics.

The delay-coupled reservoir can also serve as a model system to investigate how two different delays might interact. In a previous study, Nieters et al. (2017) highlighted strange dependencies that arise if Equation 11.3 is expanded to

$$dx(t) = -ax(t) + g\left(x(t-\tau_1), x(t-\tau_2), u(t)\right). \tag{11.4}$$

Delays that are close to simple rational, or even integer multiples of each other, lead to a poorly performing reservoir computer—how close is too close is controlled by the decay rate α of the exponential decay in the system. A too strong dependency of a sampling point onto its own history—the effect of choosing the $\tau_2 = 2\tau_1$—is detrimental. This delicate sensitivity to the choice of a second delayed feedback is reminiscent of the sensitivity to different delays in microcircuits mentioned earlier but is, of course, also an artifact of the discretized system used to model the activity at N sampling point with an analytic approximation and discretization. Future work must focus on a more realistic setting, where delays are distributed and continuous to investigate whether sharp transitions between well- and badly performing models also occur.

The takeaway from previous investigations into delay-coupled computation is that the added complexity can induce a complex temporal dynamics readout by an appropriate readout mechanism, which can benefit computation significantly. Reservoir computing is a compatible concept that embeds models, such as the delay-coupled reservoir, in the context of neural computation. More work is needed to connect the observed effects of delay coupling more closely with biological reality. Studies also highlight how complex neural networks may actually be that are subject to multiple delayed interaction effects. A possible perspective to study such systems abstractly is to connect single nodes with distributed delays recurrently in a reservoir, in the sense of a classical recurrent

network. In such a network of networks, each node itself can be regarded as a simple recurrent system (Figure 11.3).

Discussion

In this chapter, we have discussed several computational principles at the level of individual neurons and networks of neurons, and addressed the implications of delayed communication. These principles ranged from specifically tuning single neurons to implement well-defined computational tasks (i.e., independent component analysis) to reservoir computing to implement computing based on randomly connected networks and random feature expansion. This diversity and wide range of functions can be viewed as either an overwhelming complexity that might just hide a key underlying unifying principle not yet uncovered, or a rich diversity used by the evolution as a large reservoir of tools and tricks to implement efficient computational circuits. If the latter is true, then the simple question of which computational principle do we discard is not sufficient. Instead, we need to address efficiency in terms of performance and the use of resources, robustness to noise and structural changes, and generalizability of the computational principles for different tasks. The ultimate question, however, remains essentially open: How does the cortex, or the brain, compute information?

embeddedworld2020
Exhibition&Conference
... it's a smarter world

A visit to the neuromorphic zoo

Johannes Leugering
Institute for Integrated Circuits
Fraunhofer Gesellschaft e.V.
Erlangen, Germany
johannes.leugering@iis.fraunhofer.de

***Abstract*— Over the course of the last decade, neural networks have finally found their way from mostly academic research into commercial applications. So far, this transition has taken place primarily behind closed doors at high-performance computing centers – but with ever more powerful mobile devices and a growing interest in the Internet of Things, a similar revolution is ahead of us in the embedded device market. One technology that takes center stage in these developments is specialized *neuromorphic* hardware, custom designed for executing neural network applications. In this paper, we would like to provide some background information on this fascinating branch of hardware development for interested readers from other disciplines, compare different approaches and provide an overview of the current state of the field.**

Keywords—neuromorphic hardware; embedded AI; neural networks; deep learning; accelerators

I. THE DEEP LEARNING REVOLUTION

The last decade has brought with it a remarkable transformation of the field of artificial intelligence. Starting in 2012, when a Deep Neural Network now known as AlexNet [1] beat all competing approaches in a highly competitive computer vision challenge for the first time, Deep Learning has enjoyed a stellar rise in popularity - both in academia and industry alike. Today, it has become a ubiquitous and indispensable tool for a broad range of applications: Unlocking a mobile phone through facial recognition [2], controlling it through voice [3, 4], or having it translate a website [5] likely relies on a deep neural network in the background to bear the brunt of the work. On the other end of the spectrum, powerful servers and clusters employ deep neural networks for the automated analysis of big medical data sets [6], economic forecasts [7] or epidemiological predictions [8] etc., and ever-more complex neural networks are being developed to tackle increasingly harder problems.

In the wake of this "Deep Learning revolution" and as we mourn the gradual decline of Moore's law [9], there is a growing demand for innovative hardware solutions to sustain this development – even as we approach the limits of established technologies. In particular *neuromorphic hardware*, which is often subsumed with other approaches under the generic umbrella terms *next generation computing* (NGC) or *non-von-Neumann computing*, promises to deliver critical performance benefits that pave the way for further market adoption of machine learning and artificial intelligence. But what exactly *is* neuromorphic hardware, and how does it work? Why is it becoming increasingly relevant today, and where might it lead us in the future? We'll discuss these questions in the following, starting with a little bit of background.

II. WHAT EXACTLY *IS* NEUROMORPHIC HARDWARE?

The term "neuromorphic" is obviously a portmanteau of "neuro-" and "-morphic", and it describes hardware that is in some way inspired by the *morphology* of biological *neural* systems. Since biological inspiration can take many forms and can be taken to different levels, the label “neuromorphic hardware” is applied rather loosely to an entire research field that shares the common objective to *implement (some) neural network models efficiently in hardware*. The best way to achieve this varies, depending on the type of neural network to accelerate, design constraints and the performance criteria that have to be optimized, e.g. power, latency, noise robustness etc. There are many degrees of freedom in the design process, among them:

- What *type of networks* should be used, e.g. neurons with continuously valued states that change continuously with time, or with discrete states that are updated at discrete time-steps, or spiking neurons that communicate asynchronously?
- Should these computations be realized by *analog or digital* circuitry?
- Should the *topology of the networks* be restricted to some specific structure, e.g. to feed-forward, recurrent or convolutional networks?
- Which components of the network should be directly implemented by *dedicated hardware components*, and which, if any, should be emulated with a more *conventional processor design*?
- Can the design be decomposed into *functional modules*, and if so, how should these modules communicate with each other?

Any combination of these (and many more) design choices leads to a different species in zoo of neuromorphic hardware designs. In practice, this results in a broad continuum from specialized multi-processor designs on the one end, all the way to full analog instantiations of biological neural network models

on the other end, where each neuron and synapse of the neural network model has a corresponding dedicated electrical counterpart. Neuromorphic hardware therefore encompasses several diverse technologies, rather than any one in particular, so an exact technical definition of the term is difficult. Instead, we can identify some characteristic features, or *design principles*, that distinguish (most) neuromorphic hardware designs from more conventional approaches:

- *highly parallel* computing instead of to the sequential operation of a single central processing unit (CPU)
- the use of *distributed and decentralized* memory instead of a central dedicated storage
- a system design specifically *optimized to implement neural networks* of some form.

III. How can/does neuromorphic hardware work?

Since neuromorphic hardware is designed to accelerate neural networks, its merits can only be explained in the context of neural networks. Luckily, the mathematical model underlying most machine-learning applications of neural networks is actually very simple. To get everyone on the same page, what follows is a (hopelessly incomplete) high-level glance at the theory of neural networks, focusing only on those aspects that are relevant for neuromorphic hardware designers. For a more complete introduction, we refer to [10] and references therein. Everyone who is familiar with the theory already can safely skip right ahead to the next section, where we then discuss different approaches to implementing such network models in hardware.

A. *Excursion: How do (deep) neural networks work?*

In the machine-learning context[1], a neural network is a graph structure composed of *neurons* that are connected by *synapses*. Each synapse can multiplicatively scale its input, which may be the output of some neuron or an external source, by some number called the synapse's *weight*, and transmit the result to another neuron. Each neuron linearly combines all of the signals from its incoming synaptic connections (scaled by their respective weights) into a single signal. This signal is then non-linearly transformed to produce the neuron's output.

Mathematically [10], the neuron j's output $y_j(t)$ at time t is therefore a function of the input signals $x_i(t)$, and has the simple form $y_j(t) = f\left(\sum_i w_{j,i}\, x_i(t) + b_j\right)$, where $w_{j,i}$ represents the scalar *weight* of a synaptic connection from input x_i to neuron j, and b_j is a neuron specific offset or *bias* term. We can group all the weights and bias terms in a single weight matrix W and the bias vector $\vec{b}$, which results in the simple matrix equation $\vec{y}(t) = f\left(W\vec{x}(t) + \vec{b}\right)$. From this equation, it should become clear, that the multiply-accumulate operations (MACs) required for matrix-vector multiplication constitute the main computational cost for simulating neural networks.

One major exception to this story are *spiking neural networks* (SNNs, [11]), which encode a neuron's output instead by a sequence of brief, stereotypical pulses (*spikes*). In these networks, much like pulse code modulate (PCM, [12]) in digital signal processing, it is the *number of spikes per unit time* that conveys the magnitude of a signal, not the amplitude. The corresponding mathematical model requires an explicit representation of time and is hence best suited for the processing of time-series.

Despite the simple mathematical formalism, many types of neural networks can be realized by different classes of weight matrices. To give a few examples, a (block)-triangular weight matrix represents a (layered) feed-forward network. Diagonal blocks represent recurrently connected layers (i.e. groups of mutually connected neurons), and all blocks on the second or higher off-diagonal represent so-called *skip connections*. Off-diagonal blocks in Toeplitz-form resemble *convolutional* layers – a structure that has proved to be invaluable in image processing tasks, and so on. Besides the structure of the weight matrix, we can also choose what numeric type of entries it should contain. For example, instead of real valued weights, we can use integer valued weights, pick from an arbitrary set of discrete weights, e.g. binary ($w \in \{-1,1\}$) or ternary ($w \in \{-1,0,1\}$), or even use a compressed encoding of the weights. To summarize, there are countless interesting classes of weight matrices, and each of them has specific implications for the corresponding class of neural networks and offers specific opportunities for hardware acceleration.

In order for a neural network to do anything useful at all, the free parameters, i.e. weights and bias terms, must be set to specific task-dependent values. Earlier engineering approaches like the famous *Neocognitron* [13] anticipated many of features of deep neural networks, but relied heavily on domain-knowledge and inspiration from the network structures observed in nature[2]. The real breakthrough happened decades later, when improvements in computer technology suddenly made it feasible to directly optimize (or *train*) the weight matrices (and bias terms) of highly structured networks (e.g. deep convolutional feed-forward networks [1]) to minimize errors (or *loss*) on extremely large data-sets.

Since global optimization of such large non-linear systems is near impossible, the work-horse for the optimization of deep neural networks are simple, greedy, gradient-based algorithms, that differentiate the loss function on a training data-set with respect to the network's parameters, and use this information to iteratively improve the parameters [10]. While this is significantly more difficult for spiking neural networks (the discrete-time nature of their event-based communication makes the calculation of gradients difficult), some remedies exist [14, 15] that allow us to use similar tools even for training spiking neural networks. As a result, gradient-based methods have *de facto* become such a central part of deep learning, that *differentiable programming* has even been suggested as a more accurate label for the entire field [16].

[1] We only discuss neural network models for machine-learning, not bio-physically accurate models of nervous systems.

[2] The study of these biological and artificial *connectomes* under the name *connectionism* was the intellectual precursor of modern deep learning.

B. The Neuromorphic Zoo

As this brief overview/recap of neural networks hopefully shows, there are many knobs to turn in the construction of neural networks, and the possible hardware implementations are similarly plentiful. In the following, let's have a brief look at several different approaches to neuromorphic hardware design. We stay on a rather high conceptual level here, and discuss five clusters of approaches, grouped by the degree to which the network model is directly reflected in the hardware. A more complete overview and a more in-depth discussion of the various underlying hardware design concepts can be found e.g. in [17, 18]. We begin with conventional computing devices, and end with truly neuromorphic, fully analog designs that replicate each individual synapse in hardware.

1) Generic co-processors and graphics cards

Since a major fraction of the simulation and training time for neural networks is spent on MAC operations, the key innovation in most accelerator designs is an efficient hardware implementation of matrix multiplication for some class of weight matrices. Arguably the most flexible and generic form of hardware accelerators for neural networks are therefore conventional many-processor designs like graphics processing units (GPUs, [19]) or other "number crunching" co-processors like tensor processing/streaming units (TPUs, [20, 21]), which have been thoroughly optimized for large and fast matrix-multiplications. They are generally not considered neuromorphic hardware, but the high demand of such devices for deep learning applications, among others, has driven the development of a new generation of GPUs and TPUs optimized entirely for generic parallel compute tasks, and software libraries [22, 23] have accordingly begun to delegate more and more parallel operations away from the CPU to such co-processors. However, their versatility comes at a high price: the ability to execute arbitrary programs requires an extensive control-logic, powerful arithmetic logic units and a cache-, memory- and bus-system optimized for arbitrary memory access and fast data transfer. This overhead is unnecessary for many neural network architectures and can lead to power, performance or latency penalties. In addition, since the main speed-up offered by such co-processors is through the acceleration of matrix multiplication, they yield hardly any benefit at all for certain types of networks like SNNs, extremely sparsely connected networks or networks with non-linear synaptic effects.

2) Custom many-processor designs

Similar in spirit, albeit more closely focused on neural network applications, are specialized many-core designs (e.g. [24, 25] among many others), that distribute the task of simulating or training a large neural network across many independent processor cores. They typically support a (reduced) instruction set tailored and optimized towards neural network applications. Rather than by arbitrary access to shared memory, these designs typically implement an efficient routing or message passing system for the exchange of information between the nodes. Despite the focus on neural network applications, the neurons are here emulated algorithmically in software, and data flows through a shared bus-system, rather than dedicated synapses. Since these devices do not directly implement any of the components of a neural network in hardware, they are not genuine neuromorphic hardware in the narrowest sense[3], but they are generally discussed alongside neuromorphic hardware due to their near-identical application areas and user interfaces.

3) Digital deep learning accelerators

There is another class of digital accelerators (e.g. [26, 27, 28]), that is designed and optimized on a low level entirely for the implementation of (some) deep neural networks. Here, the operation of individual neurons is approximated by a dedicated digital logic circuit, that realizes the specific MAC operations required for the network class of interest (respecting the structure as well as the bit-precision the neurons' input weights and activation functions). These devices therefore don't allow arbitrary code execution, but instead require provisioning with the precise topology and coefficients of the network. Once configured, they act as a black box that efficiently executes the provisioned network, mapping digital input signals onto the network's digital outputs.

Rather than on their own, such deep learning accelerators can be used as small cores embedded within a larger many-core system similar to the ones discussed above. Such a modular design can be more effective, flexible and easier to scale, in particular when a specific network architecture to be accelerated allows for specific optimizations. For example, convolutional neural networks re-use the same structure of synaptic weights repeatedly for different neurons (also called *weight-sharing*), which can be implemented very efficiently by re-using the same hardware substrate in a time-multiplexed design that updates the network one neuron at a time. Generally in feed-forward networks, the neurons within one layer are conditionally independent given their input and can thus be processed in parallel. Here, an efficient hardware solution could update one entire layer at a time. For sparse weight matrices, an optimized handling of zeros can further improve performance, while networks with low-precision (e.g. ternary) weights can be implemented by much more compact circuits.

4) Analog deep learning accelerators

As we saw before, the mathematical models of neural networks are typically given in terms of real numbers. So rather than approximating them via discrete digital circuitry, another natural approach is to instead represent the real-valued quantities of the model by real-valued physical quantities like analog voltages, currents or charges. Such a use of analog circuit design goes back to the earliest attempts of neuromorphic hardware design in the 1950s [29], but fell out of favor during the digital revolution in electronics. While the susceptibility to noise is still a major challenge for most applications of analog circuitry, some neural networks have, quite surprisingly, proven to be remarkably robust to the effects of noise [30] – in fact, some forms of noise may even *help to improve* the robustness of the system [31]! Other pit-falls of analog circuit design, e.g. the difficulty of precisely controlling (non-)linearities in the system,

[3] In fact, they are not limited to neural network applications at all, and can be used for other tasks with similar demands.

are much less critical for neural networks than other applications, because the networks have sufficient degrees of freedom to counteract such defects (provided, of course, that the defects are known). Despite these challenges and limitations, which ultimately caused the transition to digital circuit design, there are of course also major benefits for the implementation of neural networks in the analog domain. First, our continuous model of a neuron is remarkably similar to that of a logic gate[4] – one might even view it as a continuous, weighted extension of logic gates – with one crucial difference: neurons can be differentiated with respect to their parameters, which, as we have seen above, is critical for deep learning [10]. Deep learning therefore provides a framework to optimize analog circuits in a way that cannot be directly applied to digital circuits. While it is of course possible to approximate the behavior of continuous neuron models by digital circuitry (see above), this can result in a high component count of transistor and logic gates, each of which has itself a complexity rivaling that of an analog implementation of the neuron model [17]. In the analog approach, multiplication and addition is instead realized by direct application of Ohm's and Kirchhoff's laws, i.e. by choosing appropriate values of resistive elements to represent individual synaptic weigths and accumulating the resulting currents. This also allows for ultra-low power applications, possibly at the expense of an increased noise-floor, and alleviates the need to wait for signals to settle, which makes low-latency asynchronous designs possible.

Due to strong barriers to entry, e.g. high manufacturing costs and long development cycles, only a comparatively small number of fully analog deep learning accelerators have actually seen the light of day. However, a vast amount of literature has been written about this already (see e.g. [17, 18] and references therein) and if the number of recent start-ups and research projects in that field is any indication, there is a substantial and growing commercial and academic interest, as well.

5) Spiking neural network accelerators

Last but not least, neuromorphic hardware for spiking neural networks, currently an outlier in the machine learning world, is set to become another major branch of hardware accelerated embedded AI. Contrary to conventional neural networks, the purely event-driven operation of spiking neural networks defies the simple mathematical frameworks of continuous function approximation and periodic sampling. An efficient implementation of such networks is therefore difficult for both clocked digital logic as well as in conventional imperative programming paradigms. Combined with the increased complexity of training algorithms for SNNs [14, 15, 32], this may explain the relative lack of attention these networks have received within the machine learning community – despite advocacy by some leading theoreticians in the field [11]. However, the very same properties of spiking neurons that appear as major obstacles for efficient software implementations (e.g. integration and low-pass filtering of signals over time and rising-edge triggered generation of pulses) are commonplace [12] in signal processing and can be implemented by simple analog circuits [33]. As a result, there is little overhead in complexity when switching from an analog to a spike-based network design. To the contrary, since each neuron's spiking output is a binary signal that can be converted into an analog signal merely by low-pass filtering (one of the axioms of the neural engineering framework [34]), spike-based neuromorphic hardware can combine the best of both worlds: the highly energy-efficient computation of analog circuitry and the binary transmission of signals via spikes, which decreases susceptibility to noise and simplifies routing and buffering.

Just as for digital hardware accelerators, the communication between individual neurons of a spiking network can therefore be implemented either through dedicated electrical lines or through a (digital) package routing system, the most popular of which is *address event representation* [35] encoding, where each spike is conveyed as a package containing the "address" of the neuron from which it originated. While such a routing system greatly improves the scalability of the system by time-multiplexing the usage of the same communication channels, it requires sophisticated scheduling and low latencies that can become prohibitive as the number of interconnected neurons increases. A hybrid approach that uses many cores with full *internal* connectivity through dedicated lines, connected *to each other* via a common bus system, is therefore a popular compromise (e.g. in [36, 37]).

IV. Why is neuromorphic hardware relevant today?

With this brief overview in mind, one might wonder, why neuromorphic hardware has so suddenly become a hot topic among AI researchers and ASIC developers. None of these ideas seem novel enough to justify this rise in popularity – in fact, similar ideas have been continuously suggested since the very beginning of artificial intelligence research and computer science in the 1950s [29]. Even the term "neuromorphic hardware" was popularized already in the 1990s by Carver Mead [38], who has been a pioneer in this research area since its early days. So why should we invest in neuromorphic hardware today, and why didn't this happen before?

The most obvious argument is purely opportunistic: never before have neural networks had sufficient size to be practically useful for complex, data-driven applications. Now, with the breakthroughs in image classification competitions during the recent years, neural networks have finally proven their worth for commercial applications, and have received massive exposure to the public and industry ever since. This growing popularity has correspondingly lead to an increased demand of efficient hardware on which to run neural networks.

The applications in fields like image processing [1], gaming [39], text analysis [40], audio processing [41, 4] and data-science, e.g. in medical image analysis [6], have diversified and become more complex, with weight coefficients numbering anywhere from hundreds of thousands up to a staggering billion

[4] The earliest theories of neural networks by McCulloch & Pitts [67] already established this connection.

for extreme cases [42][5]. The range of applications is likely only going to increase, as a growing number of mobile devices from smartphones [43] all the way to autonomous vehicles do already (or will soon) use neural networks for demanding image recognitions tasks, and require the corresponding computing power.

So far, we have been riding the wave of ever-improving CPUs and GPUs, and technological progress alone could sustain the growing demand, but as we begin to witness the end of Moore's [9] law, we need fundamentally new ideas. Current state-of-the-art 7nm CMOS technology approaches physical limits and it seems unlikely that we can continue scaling down size, power-consumption or latency much further. At the same time, the total training time, power-consumption and initial cost of systems capable of simulating larger state-of-the-art neural networks has sky-rocketed to an unsustainable level [44], while the limited power budget of mobile devices has been a limiting factor for many potentially interesting applications.

Of course, neuromorphic hardware, too, has benefitted from the technological advances in electronics manufacturing during the last few decades. Besides a new market, the availability of new technologies is thus another reason for a renewed interest in neuromorphic hardware development. For example, new transistor design principles like fin field-effect transistors (FinFETs, [45]), fully depleted silicon on insulator (FD-SOI, [46]) and floating multi-gate MOSFET transistors [47, 48] as well as special neuro-transistors (vMOS, [49]) have enabled extremely low-power applications and novel neuromorphic hardware designs. Since neural networks require a sizeable amount of memory for storing the network topology and synaptic weights, neuromorphic hardware also stands to gain a lot from new trends in memory technology. With small feature sizes of 28nm and below and advances in dynamic RAM (DRAM) and static RAM (SRAM), it has become possible to store reasonably sized networks directly *in silico*, and process data right where it is stored. While this new paradigm of *in-memory computing* [50, 51] is by no means limited to neuromorphic hardware, the highly distributed structure of neural networks can leverage this advantage particularly well, and thus overcome the memory-bottleneck that conventional von-Neumann architectures suffer from.

Since the network coefficients (typically) do not change at all during inference, emerging non-volatile memory technologies (eNVM, [52]) are particularly interesting for neuromorphic hardware. One major development is the emergence of several forms of *memristive* devices, which can act as the programmable resistive components required for re-configurable analog hardware accelerators [53]. Competing technologies like charge trap flash memory (CTF) , ferro-electric field-effect transistors (FeFETs), resistive RAM (ReRAM), conductive bridge RAM (CBRAM) and phase change memory (PCM) [52, 54] all exploit different physical phenomena to allow non-volatile storage on chip, many of them supporting the storage of analog values at a multi-bit resolution [51], which is critical for analog hardware accelerators and reduces the die-space required for memory.

Lastly, we have learned a lot about neural networks in the meantime: we have demonstrated their capability, know now that there are indeed use-cases for large networks, and we have found better ways and tools [22, 23] to train even large networks. As we continue to learn more in-depth about which network topologies are effective for which specific tasks, and why, we also develop a better understanding of what sort of networks are worth accelerating in hardware – and which are not.

V. WHERE IS THE DEVELOPMENT OF NEUROMORPHIC HARDWARE HEADED IN THE NEAR FUTURE?

Of course, predicting the future is hard, particularly for a field that has been through all the season (including the dreaded, recurring "AI winter") several times already. But with new exciting technologies on the horizon, such as 3D [55] and wafer-scale [56] integration, nano-wire transistors [57], carbon nano-tubes [58] on-chip, silicon photonics [59], spintronics [60], ever-smaller micro- and nano-electro-mechanical systems (MEMS, NEMS [61]), integrated sensor-processor systems and more, it is hard to not feel optimistic about the future of neuromorphic hardware.

The technological progress is likely to bring neuromorphic hardware into new application areas, where it can reduce energy consumption, latency or the cost of existing solutions. For example, co-processors for AI are already being included in modern smartphones [43] to reduce CPU load during AI applications and therefore prolong battery life. On the other end of the spectrum, the increasing demand for high-performance computing clusters and cloud-services that provide "deep-learning-as-a-service" [62] shows a market for server-side energy efficient, dedicated neuromorphic hardware co-processors.

Neuromorphic hardware can also enable applications that are in principle possible right now, but not economically viable yet, such as natural language [63] or gesture based [64] user-interfaces for controlling a wide range of electrical devices, e.g. in the context of the internet-of-things or home-automation and appliances. Industry could use neuromorphic hardware to make even low-level processes in manufacturing more adaptive or responsive, or improve the interaction between humans and machines.

Finally, the adoption of neuromorphic hardware might even enable machine learning solutions that are flat-out impossible right now, such as many sophisticated real-time applications like the processing and fusion of complex, high-dimensional sensor data and the intelligent real-time control of sophisticated robots or production plants. Power savings might enable novel mobile applications like large-scale distributed sensor networks [65], or autonomous systems that are smart enough to act on their own and resilient enough to survive in difficult environments. By optimizing die-space, neuromorphic hardware could also find its way into miniaturized sensors, e.g. ingestible medical sensors

[5] It should be recognized that despite the general trend of increasing network sizes, there are also contrary efforts to reduce the number of parameters, e.g. [68].

[66] and much more. Of course, the most important applications might turn out to be entirely different from the ones listed here, but we are confident that industry and academia will find countless ways to capitalize on neuromorphic hardware in the future!

REFERENCES

[1] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advanced in Neural Information Processing Systems 25*, 2012.

[2] O. M. Parkhi, A. Vedaldi and A. Zisserman, "Deep face recognition," in *Procedings of the British Machine Vision Conference 2015*, Swansea, 2015.

[3] J. P. Dominguez-Morales et al., „Deep Spiking Neural Network model for time-variant signals classification: a real-time speech recognition approach," in *2018 International Joint Conference on Neural Networks (IJCNN)*, Rio de Janeiro, 2018.

[4] D. Li, H. Geoffrey und K. Brian, „New types of deep neural network learning for speech recognition and related applications: An overview," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013.

[5] Y. Wu et al., „Google's neural machine translation system: bridging the gap between human and machine translation," *arXiv:1609.08144 [cs]*, 10 2016.

[6] G. Litjens et al., "A survey on deep learning in medical image analysis," *Medical Image Analysis*, vol. 42, pp. 60-88, 12 2017.

[7] D. Xiao, Z. Junw, L. Ting und D. Junw, „Deep learning for event-driven stock prediction," in *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[8] M. Salathé, "Digital epidemiology: what is it, and where is it going?," *Life Sciences, Society and Policy*, vol. 14, p. 1, 12 2018.

[9] M. M. Waldrop, "The chips are down for Moore's law," *Nature*, vol. 530, pp. 144-147, 2 2016.

[10] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436-444, 5 2015.

[11] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, vol. 10, pp. 1659-1671, 12 1997.

[12] B. M. Oliver, J. R. Pierce und C. E. Shannon, „The philosophy of pcm," *Proceedings of the IRE*, Bd. 36, pp. 1324-1331, 11 1948.

[13] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, pp. 193-202, 4 1980.

[14] A. Sengupta, Y. Ye, R. Wang, C. Liu and K. Roy, "Going deeper in spiking neural networks: vgg and residual architectures," *Frontiers in Neuroscience*, vol. 13, 2019.

[15] E. O. Neftci, H. Mostafa und F. Zenke, „Surrogate gradient learning in spiking neural networks: bringing the power of gradient-based optimization to spiking neural networks," *IEEE Signal Processing Magazine*, Bd. 36, pp. 51-63, 11 2019.

[16] Y. LeCun, *Facebook Post.* https://www.facebook.com/yann.lecun/posts/10155003011462143

[17] G. Indiveri and T. K. Horiuchi, "Frontiers in neuromorphic engineering," *Frontiers in Neuroscience*, vol. 5, 2011.

[18] C. D. Schuman et al., „A survey of neuromorphic computing and neural networks in hardware," *arXiv:1705.06963 [cs]*, 5 2017.

[19] S. Chetlur et al., „Cudnn: efficient primitives for deep learning," *arXiv:1410.0759 [cs]*, 12 2014.

[20] *Groq's tensor streaming architecture*, 2019, https://groq.com/groqs-tensor-streaming-architecture/

[21] N. P. Jouppi, C. Young, N. Patil and D. Patterson, "A domain-specific architecture for deep neural networks," *Communications of the ACM*, vol. 61, pp. 50-59, 8 2018.

[22] The Theano Team et al., „Theano: A Python framework for fast computation of mathematical expressions," *arXiv:1605.02688 [cs]*, 5 2016.

[23] M. Abadi et al., „Tensorflow: large-scale machine learning on heterogeneous distributed systems," *arXiv:1603.04467 [cs]*, 3 2016.

[24] S. B. Furber, F. Galluppi, S. Temple und L. A. Plana, „The spinnaker project," *Proceedings of the IEEE*, Bd. 102, pp. 652-665, 5 2014.

[25] *Graphcore: Accelerating machine learning for a world of intelligent machines, https://www.graphcore.ai/*

[26] Y.-H. Chen, T. Krishna, J. S. Emer und V. Sze, „Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," *IEEE Journal of Solid-State Circuits*, Bd. 52, pp. 127-138, 1 2017.

[27] M. H. Ionica und D. Gregg, „The Movidius Myriad Architectures Potential for Scientific Computing," *IEEE Micro*, Bd. 35, pp. 6-14, 1 2015.

[28] „NVDLA Primer," 2018, http://nvdla.org/primer.html

[29] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain.," *Psychological Review*, vol. 65, pp. 386-408, 1958.

[30] D. Rolnick, A. Veit, S. Belongie und N. Shavit, „Deep learning is robust to massive label noise," *arXiv:1705.10694 [cs]*, 2 2018.

[31] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: representing model uncertainty in deep learning," in *International Conference on Machine Learning*, 2016.

[32] J. H. Lee, T. Delbruck und M. Pfeiffer, „Training deep spiking neural networks using backpropagation," *Frontiers in Neuroscience*, Bd. 10, 11 2016.

[33] Y. P. Tsividis, „Integrated continuous-time filter design," in *Proceedings of IEEE Custom Integrated Circuits Conference - CICC '93*, 1993.

[34] C. Eliasmith und C. H. Anderson, Neural engineering: computation, representation, and dynamics in neurobiological systems, Cambridge, Mass: MIT Press, 2003.

[35] M. Mahowald, An analog vlsi system for stereoscopic vision, Boston, MA: Springer US, 1994.

[36] M. Davies et al., „Loihi: a neuromorphic manycore processor with on-chip learning," *IEEE Micro*, Bd. 38, pp. 82-99, 1 2018.

[37] F. Akopyan et al., „Truenorth: design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Bd. 34, pp. 1537-1557, 10 2015.

[38] C. Mead, „Neuromorphic electronic systems," *Proceedings of the IEEE*, Bd. 78, pp. 1629-1636, 10 1990.

[39] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang und W. Zaremba, „Openai gym," *arXiv:1606.01540 [cs]*, 6 2016.

[40] *Better language models and their implications*, 2019, https://openai.com/blog/better-language-models/

[41] A. v. d. Oord et al., „Wavenet: a generative model for raw audio," *arXiv:1609.03499 [cs]*, 9 2016.

[42] *Megatronlm: training billion+ parameter language models using gpu model parallelism*, 2019, https://nv-adlr.github.io/MegatronLM

[43] *Ai benchmark: all about deep learning on smartphones in 2019.* https://www.groundai.com/project/ai-benchmark-all-about-deep-learning-on-smartphones-in-2019/1

[44] J. Toole, *Deep learning has a size problem*, 2019, https://heartbeat.fritz.ai/deep-learning-has-a-size-problem-ea601304cd8

[45] B. Yu et al., „FinFET scaling to 10 nm gate length," in *Digest. International Electron Devices Meeting,*, 2002.

[46] J.-P. Colinge, „Fully-depleted SOI CMOS for analog applications," *IEEE Transactions on Electron Devices*, Bd. 45, pp. 1010-1016, 5 1998.

[47] F. Khateb, "Bulk-driven floating-gate and bulk-driven quasi-floating-gate techniques for low-voltage low-power analog circuits design," *AEU - International Journal of Electronics and Communications*, vol. 68, pp. 64-72, 1 2014.

[48] J. P. Colinge, "Multi-gate soi mosfets," *Microelectronic Engineering*, vol. 84, pp. 2071-2076, 9 2007.

[49] Z. Wang et al., "Capacitive neural network with neuro-transistors," *Nature Communications*, vol. 9, pp. 1-10, 8 2018.

[50] D. Ielmini and H.-S. P. Wong, "In-memory computing with resistive switching devices," *Nature Electronics*, vol. 1, pp. 333-343, 6 2018.

[51] M. Le Gallo et al., "Mixed-precision in-memory computing," *Nature Electronics*, vol. 1, pp. 246-253, 4 2018.

[52] A. Chen, "A review of emerging non-volatile memory (Nvm) technologies and applications," *Solid-State Electronics*, vol. 125, pp. 25-38, 11 2016.

[53] O. Krestinskaya, A. P. James und L. O. Chua, „Neuromemristive circuits for edge computing: a review," *IEEE Transactions on Neural Networks and Learning Systems*, Bd. 31, pp. 4-23, 1 2020.

[54] C.-Y. Lu, „Future Prospects of NAND Flash Memory Technology—The Evolution from Floating Gate to Charge Trapping to 3D Stacking," *Journal of Nanoscience and Nanotechnology*, Bd. 12, pp. 7604-7618, 10 2012.

[55] J. U. Knickerbocker et al., „3D silicon integration," in *2008 58th Electronic Components and Technology Conference*, 2008.

[56] J. Schemmel, J. Fieres und K. Meier, „Wafer-scale integration of analog neural networks," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 2008.

[57] W. Taube Navaraj et al., "Nanowire fet based neural element for robotic tactile sensing skin," *Frontiers in Neuroscience*, vol. 11, 2017.

[58] J. Tang et al., "Flexible CMOS integrated circuits based on carbon nanotubes with sub-10 ns stage delays," *Nature Electronics*, vol. 1, pp. 191-196, 3 2018.

[59] R. Soref, „The past, present, and future of silicon photonics," *IEEE Journal of Selected Topics in Quantum Electronics*, Bd. 12, pp. 1678-1687, 11 2006.

[60] I. Žutić, J. Fabian und S. Das Sarma, „Spintronics: Fundamentals and applications," *Reviews of Modern Physics*, Bd. 76, pp. 323-410, 4 2004.

[61] S. E. Lyshevski, Mems and nems : systems, devices, and structures, CRC Press, 2018.

[62] M. S. V. Janakiram, *The rise of artificial intelligence as a service in the public cloud*.2018, https://www.forbes.com/sites/janakirammsv/2018/02/22/the-rise-of-artificial-intelligence-as-a-service-in-the-public-cloud/

[63] J. Hirschberg and C. D. Manning, "Advances in natural language processing," *Science*, vol. 349, pp. 261-266, 7 2015.

[64] N. Neverova, C. Wolf, G. W. Taylor and F. Nebout, "Multi-scale deep learning for gesture detection and localization," in *Computer Vision - ECCV 2014 Workshops*, Cham, 2015.

[65] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam und E. Cayirci, „A survey on sensor networks," *IEEE Communications Magazine*, Bd. 40, pp. 102-114, 8 2002.

[66] K. Kalantar-zadeh, N. Ha, J. Z. Ou und K. J. Berean, „Ingestible sensors," *ACS Sensors*, Bd. 2, pp. 468-483, 4 2017.

[67] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, pp. 115-133, 12 1943.

[68] F. N. Iandola et al., „SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," *arXiv:1602.07360 [cs]*, 11 2016.

135
UNIVERSITÄT OSNABRÜCK

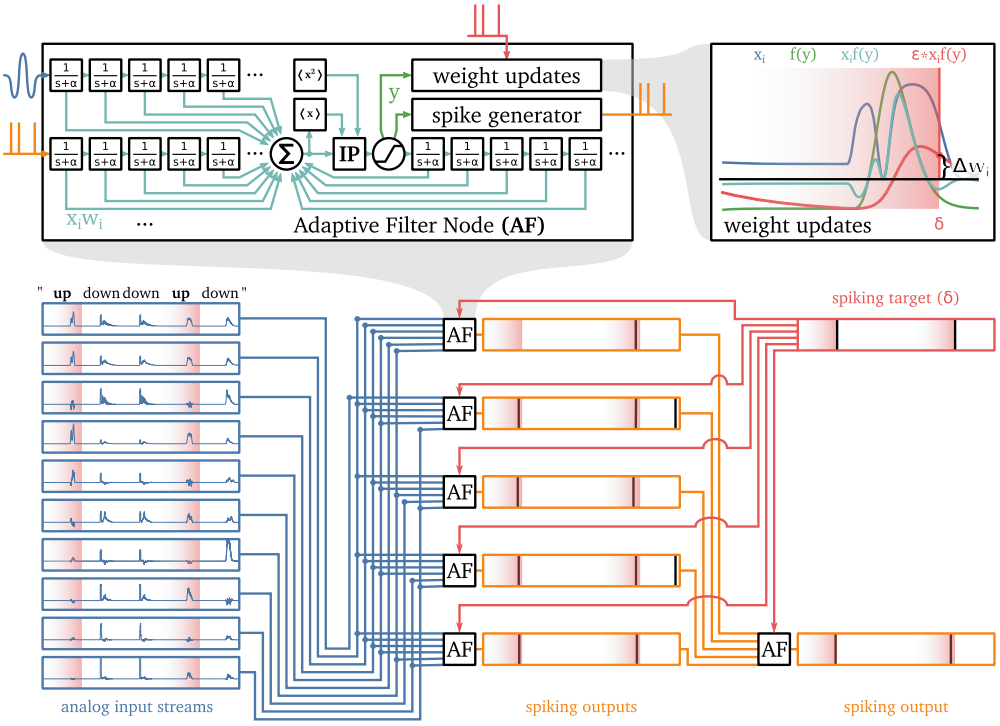
Institute of Cognitive Science

Pascal Nieters pnieters@uos.de @PascalNieters
Johannes Leugering jleugeri@uos.de @JoLeugering
Gordon Pipa gpipa@uos.de @PipaGordon

Neuromorphic Adaptive Filters for event detection, trained with a gradient free online learning rule

Machine learning problems are typically framed in a regression, classification or prediction setting, where a set of distinct data points is to be identified with corresponding labels. Artificial neural networks excel at such problemse, because their universal function approximation capability and differentiability can be leveraged for powerful gradiente-based optimization algorithms. Neuromorphic hardware however, interacting with its environment in real time faces challenges that defy this framework. One such ıple is the **detection of specific events in real time**, e the mapping from a **continuous stream** of noisy signals onto a **discrete set of events** is to be learned. temporal dimension of this task entails a **credit nment problem** for learning, since the detector must ıate a history of input signals and needs to be afforded flexible **processing delays**, which makes defining a entiable loss function for the event detection task ult. This is aggravated in a setting where the **target ıls themselves are delayed**. The constraints of ɔmorphic hardware design further restrict the available ing algorithms to "any-time" computations ɛmentable just by (traces of) locally available mation, which precludes many of the established ent-based optimization procedures.

propose a neuromorphic event detector, the **omorphic Adaptive Filter** (**NAF**) and ensembles of, that utilizes **Gamma Filter** banks [4,5] to learn a neterized multidimensional signal filter through a **rvised gradient-free online learning rule**.



te & Weight Traces:

$$\alpha \cdot (I_j - x_0^j)$$
$$\alpha \cdot (x_{i-1}^j - x_i^j)$$
$$\eta \cdot (x_i^j \cdot \left(\frac{y-1}{2}\right)^2 - \bar{w}_i^j)$$

Intrinsic Plasticity:

$$\dot{\mu} = \beta \cdot (\nu - \mu)$$
$$\dot{\sigma^2} = \beta \cdot ((\nu - \mu)^2 - \sigma^2)$$
$$\dot{\theta} = -\gamma \cdot \theta$$

ived Variables:

$$\sum_j \sum_i x_i^j w_i^j$$
$$y = \mathrm{sat}(\frac{\nu - \mu}{\sqrt{2\sigma^2}})$$

Signal condition:

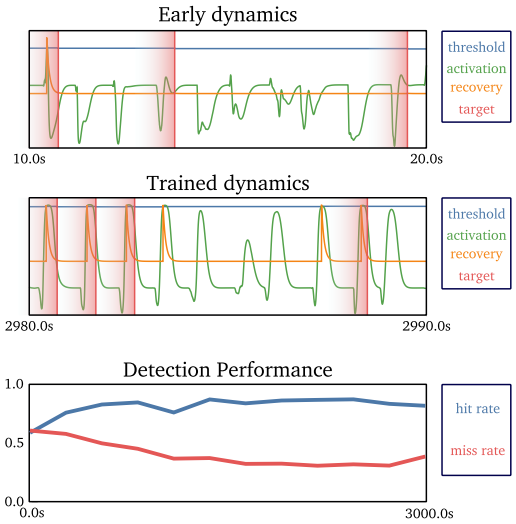
$$y \geq \theta + r$$

late on signal:

$$\theta \leftarrow \theta + \tau^+$$
$$r \leftarrow r + 1$$
$$w_i^j \leftarrow w_i^j - \kappa \cdot \bar{w}_i^j$$
$$w \leftarrow \frac{w}{||w||}$$

Update on target:

$$\theta \leftarrow \theta - \tau^-$$
$$r \leftarrow r + \tau^-$$
$$w_i^j \leftarrow w_i^j + \kappa \cdot \bar{w}_i^j$$
$$w \leftarrow \frac{w}{||w||}$$



⇧Speech command detections with ENAF.
A continuous audio stream is represented by its spectral power in 10 Mel-adjusted frequency bands. 5 NAFs each receive as input a subset of 6 random chosen frequency bands. Their spiking outputs are used as input signals for another NAF, which then provides the ensemble output. Each occurence of the desired word ("UP") in the audio stream is shortly followed by a spiking target signal to each ensemble member.

⇦NAF performance during training.
Detection rates increase while false positive rates decrease during training as the ensemble members adjust. Early on, the filter responses within an individual NAF show little specificity for the target word ("UP"), but towards the end of training, the NAF has become selective and the threshold is set to provide a reasonable trade-off between false-positive and missed detections.

Key properties of the model:

- It can be formalized as a **system of ordinary differential equations** with discontinuous updates at distinct time points.
- Event signals are generated as **output** akin to **spiking** neuron models, and **target** signals are similarly provided as a sequence of **events**.
- The learning rule, similar in spirit to the Tempotron-rule[1], in concert with weight normalization adjusts the **shape of the learned filter**.
- The learning mechanism can be thought of as enforcing an "**energy budget**", where parameter updates are performed only at two kinds of distinct events: the **emission** of a detection signal incurrs a penalty, leading to a **reduction** in the weights of contributing features, whereas at each **target** signal, the weights assigned to those features that recently contributed to high membrane potentials are **amplified**.
- An adaptive threshold paired with intrinsic plasticity mechanisms **in**- or **decreased** the detector's **sensitivity** in response to **target** or **emitted** signals, respectively, thus adjusting the trade-off between false positives and false negatives.
- An individual NAF can be viewed as a variation of a **generalized functional linear model** [6] and thus represents a reasonable week learner for machine learning applications.
- **Performance** and **robustness** can be enhanced by routing the output of multiple NAFs with diverse inputs through another NAF capable of compensating the different incurred processing delays, thus forming an **ensemble** (ENAF).
- The training and inference procedure can be realized purely by **instantaneous local computation** and **exponential memory traces.** (E)NAFs are thus good candidate systems for **neuromorphic hardware implementation**.

References:

[1] Gütig, R., & Sompolinsky, H. (2006). **The tempotron: a neuron that learns spike timing–based decisions.** Nature Neuroscience, 9(3), 420–428.
[2] Lazar, A. A. (2006). **A simple model of spike processing.** Neurocomputing, 69(10–12), 1081–1085.
[3] Lazar, A. A., & Toth, L. T. (2003). **Time encoding and perfect recovery of bandlimited signals.** In IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.
[4] Principe, J. C., de Vries, B., & de Oliveira, P. G. (1993). **The gamma-filter-a new class of adaptive IIR filters with restricted feedback.** IEEE Transactions on Signal Processing, 41(2), 649–656.
[5] Vries, B. de, & Principe, J. C. (1992). **The gamma model—A new neural model for temporal processing.** Neural Networks, 5(4), 565–576.
[6] Müller, H.-G., & Stadtmüller, U. (2005). **Generalized functional linear models.** The Annals of Statistics, 33(2), 774–805.

Acknowledgements:

We would like to thank our students *Tobias Ludwig*, *Devrim Celic* and *Felix Meyer zu Driehausen* for their input and contributions.

Neuromorphic computation in multi-delay coupled models

P. Nieters
J. Leugering
G. Pipa

Neuromorphic computing provides a promising platform for processing high-dimensional noisy signals on dedicated hardware. Using design elements inspired by neurobiological findings and advances in machine learning methodology, delay-coupled systems have recently been developed in the field of neuromorphic computing. Delayed feedback connections enable such systems to generate a complex representation of injected input in the internal state of single nodes*, which in our context refer to hardware components with nonlinear behavior and without any memory. In contrast to classical combinatorial circuits or feed-forward networks, this state is not distributed in space but in time. Hardware implementations with low hardware component counts are therefore particularly easy to design for delay-coupled systems. In this paper, we present an argument for using delay-coupled reservoirs using multiple feedback terms with different delays. We present a theoretical analysis of the resulting system, discuss surprising effects pertaining to the precise choice of delays, and provide a guideline for the optimal design of such systems.*

Introduction

Driven by recent advances in neuro-inspired machine learning, the growing field of neuromorphic computing develops new kinds of hardware dedicated to fast and energy efficient implementations of artificial neural networks (ANNs). IBM Research is developing the neuromorphic TrueNorth chip, simulating more than one million spiking neurons at just under 70 mW of power [1]. In the context of the Human Brain Project, two neuromorphic hardware architectures are investigated as part of its computing platform: an analog implementation operating in faster than real time [2] and SpiNNaker, a clocked digital design combining multiple mobile CPU cores through a fast spike routing system [3, 4].

Much like biological nervous systems [5, 6], which have inspired neural network and neuromorphic hardware research, these systems are faced with unique challenges pertaining to the reliable processing of noisy signals through complex, delayed interactions of their hardware components. For systems in discrete time, delays smaller than one clock period are below the temporal resolution of the system and do not need to be treated explicitly, greatly simplifying the design of synchronous clock systems. In this domain, many ANN architectures have been proposed and shown to be successful in modern machine learning applications [7–10]. However, this raises the question of how very fast systems or clock-free systems such as biological nervous systems can rely on delayed signal interactions. For these systems, it is a theoretically challenging but necessary task to model delays because the system is sufficiently fast for them to become relevant.

Real-time spiking neural networks have been proposed as the third generation of neuronal networks [11] and implicitly include delay effects due to synaptic interactions. One such network is the Liquid State Machine (LSM) [12] that, together with Echo State Networks [13], established reservoir computing (explained further in the next section) [14, 15] as a field of recurrent network research. Recent work has explored different approaches to implement reservoir computers in hardware including architectures based on memristors [16, 17],

Digital Object Identifier: 10.1147/JRD.2017.2664698

0018-8646/17 © 2017 IBM

field-programmable gate arrays (FPGAs) [18], and atomic switch networks (ASNs) [19].

Particularly relevant to the effects of delayed interactions are recent advances in reservoir computing research that explore simplistic real-time systems using delayed feedback instead of recurrent connectivity to generate highly complex, non-linearly history-dependent system states [20–22]. Here, we focus on the so-called delay-coupled reservoir (DCR) [21, 22] and study the advantages of using multiple delayed feedback signals.

In this paper, we present a theoretical rationale for the study of DCRs with more than one delay. To this end, we first introduce the most important concepts of DCRs. Next, we demonstrate and explain several qualitatively new effects resulting from a second delayed feedback signal. The system is evaluated on a non-linear history-dependent task, and improved task performance is verified with respect to a DCR with a single delay. We leave the reader with a critical discussion of the discovered effects, a guideline for an optimal design of a two-delay DCR, suggestions for further research, and some concluding remarks.

Delay-coupled reservoir computing

Here, we first introduce the unfamiliar reader to the main concepts of DCR computing. Reservoir computing, in general, subsumes approaches where a recurrent network undergoes little to no adaptation or training over time
[12, 13]. The activity of the recurrent network, the reservoir, is used as input to a linear model that learns to map reservoir activity onto a desired target value. As the reservoir itself is a purely input-driven dynamic system that can be viewed as a means for recurrent feature expansion, multiple different target functions can be approximated simultaneously by independent linear models. Although randomly initialized recurrent connections in the network are not optimized for any specific task, such systems can already exhibit universal computational power [23]. Combining the desirable computing qualities with the simplicity of the underlying model, the framework has proven itself as a fertile ground for research into recurrent computation in dynamic systems. Notable advances include self-organizing computation in spiking neural networks [24–26], and transferring learning from the readout layer to reservoir connectivity [27] as well as the possibility to control reservoir dynamics with conceptors [28].

An approach different from traditional reservoir computing replaces the recurrent neural network by the dynamics of a single node with delayed feedback [20–22]. Instead of the spatially distributed network activity, the state of the system is then defined by a window over its own history. This variety of a reservoir computer is referred to as a DCR [21] and implements non-linear, history-dependent computation. A schematic of a DCR is given in the upper part of **Figure 1**.

We denote the state of node with x, the delay with τ, and the input to the node at time point t with $J(t)$. We can then describe the single node dynamics with a delay differential equation (DDE) such as Equation (1):

$$\frac{dx(t)}{dt} = -\alpha x(t) + f(x(t-\tau) + \gamma\, J(t)), \tag{1}$$

where delayed activity $x(t-\tau)$ mixes with input signal $J(t)$ and enters the equation via the nonlinear function f. In accordance with previous work [21, 29], we chose $f(x) = \eta \times [x/(1+x^{\rho})]$, resulting in a system akin to the autonomous Mackey-Glass Oscillator [30]. While α, η, and ρ are parameters of the Mackey-Glass equation, γ regulates the degree to which input perturbs the dynamics in a DCR.

In each disjoint time interval of length τ, also referred to as one cycle of the system, we sample the state of the node N times. For one cycle $[t_0,\ t_0+\tau]$, we thus collect the vector $\boldsymbol{\nu}$ of sample states $\nu_k = x(t_k),\ t_k = t_0 + k\theta$, $k \in \{1 \ldots N\}$, where $\theta = \tau/N$ denotes the temporal distance between sampled states. We also refer to $\boldsymbol{\nu}$ as the vector of virtual nodes. In analogy with the spatially distributed nodes of an echo state reservoir [13], the temporally distributed virtually nodes v_k define the state of the discretized DCR during one cycle. The input signal is generated from an input function $u(t)$ multiplexed with a random mask: $J(t) = M(t)u(t)$. During each cycle, $u(t)$ is constant, and $M(t)$ is a τ-periodic, piecewise constant two-valued function composed of segments of length θ. Modulating the input this way prevents the dynamic behavior of the single node from converging and thus increases the variability in the system state during constant values of $u(t)$.

The linear readout is then trained to map the state $\boldsymbol{\nu}$ of the DCR for each cycle onto a target value. Therefore, the system as a whole learns a non-linear mapping of the history of an input sequence $u(t)$ onto a target sequence $y(t)$. In the top of Figure 1, this mapping is illustrated for both the one delay DCR and the two delay DCR introduced in the next section.

DCRs lend themselves particularly well to neuromorphic hardware implementations only requiring one non-linear node and a tapped delay line. Consequently, DCRs have recently been studied as a promising alternative to classical computational paradigms. They have been implemented in optoelectronic hardware [21, 22, 31] and in fully optical hardware [32, 33].

Methods

In this section, we extend the DCR described in Equation (1) with an additional delayed feedback and describe the

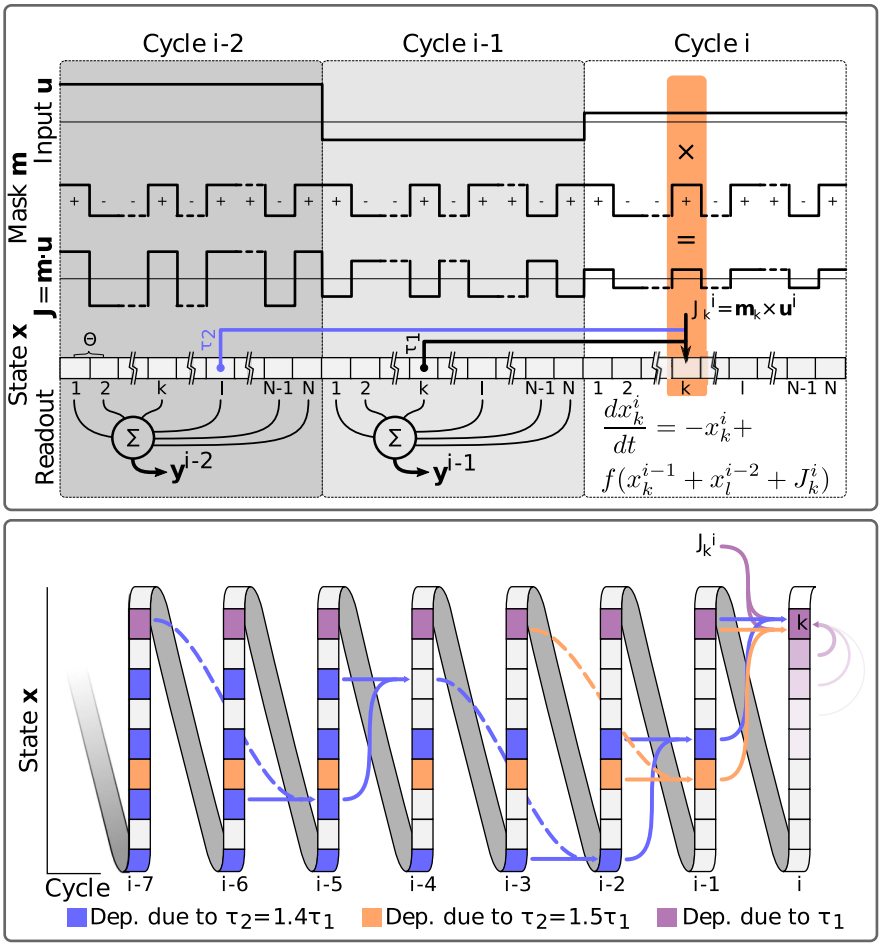


Figure 1

Reservoir computing. Top: a schematic illustration of delay-coupled reservoir computing. Piecewise constant input u is time multiplexed via the τ_1 periodic random two-valued mask $\boldsymbol{m}$ to yield the system's input J. The system's state x is given by an exponentially smoothed non-linear transformation of a linear combination of the input J and delayed feedback of previous system states (only τ_1 in the single delay DCR). On each disjoint time-interval of constant u, referred to as a cycle, a readout value $\hat{y}$ is generated by linear transformation of the vector of equidistantly sampled system states, called virtual nodes. In the two delay DCR, the system state update at a particular virtual node k within a cycle i explicitly depends ("dep.") on one or two virtual nodes delayed by τ_1 and τ_2. Bottom: The explicit dependencies introduced this way are recursively tracked across cycles in the bottom part of the figure for two different chosen values of τ_2.

theoretical differences and advantages the second delay introduces. We formalize the system as follows:

$$\frac{dx(t)}{dt} = -\alpha x(t) + f(x(t-\tau_1) + x(t-\tau_2) + \gamma\, J(t)). \quad (2)$$

Here, two delayed terms $x(t-\tau_1)$ and $x(t-\tau_2)$ are linearly mixed with the input $J(t)$, while the rest of Equation (1) remains unchanged. This extension is illustrated in the top of Figure 1.

Following [29, 34], we derive a discrete time update equation for the virtual nodes. We omit some of the mathematical detail for simplicity and refer the interested reader to [29, 34].

Consider a single cycle of length τ_1. Let $\tau_1 < \tau_2$ without loss of generality, and let $\phi(t)$ denote a solution of DDE Equation (2) for all $t < t_0$. The continuation of solution ϕ on the interval $[t_0,\ t_0 + \tau_1]$ then satisfies

$$\frac{dx(t)}{dt} = -\alpha x(t) + f(\phi(t-\tau_1) + \phi(t-\tau_2) + \gamma\, J(t)). \quad (3)$$

Given ϕ, this linear non-homogeneous first order initial value problem can be solved analytically by variation of constants. The solution is given by

$$x(t) = \phi(t_0)e^{\alpha(t_0-t)} + e^{\alpha(t_0-t)}\int_{t_0}^{t} f(\phi(s-\tau_1) + \phi(s-\tau_2) + \gamma\, J(s))e^{\alpha(s-t_0)}ds, \quad (4)$$

where s is the integration variable over time. For an efficient and conceptually simple approximation of (4), time is discretized with step-size θ. In the following, we assume that τ_1 and τ_2 are divisible by θ. As above, we consider the support point $t_k = t_0 + k\theta$ for $k = \{1 \dots N\}$. For simplicity, we write

$$f_{t_k} = f(\phi(t_k - \tau_1) + \phi(t_k - \tau_2) + \gamma J(t_k)) \ .$$

In discrete time, Equation (4) can be approximated using the trapezoidal rule:

$$x(t_k) \approx \phi(t_0)e^{\alpha(t_0-t_k)} + \frac{\theta}{2}\sum_{j=1}^{k} e^{\alpha(t_j-t_k)}(f_{t_j} + e^{-\alpha\theta}f_{t_{j-1}}). \quad (5)$$

Next, the approximate solution Equation (5) can be extended iteratively in cycles of length τ_1. To this end, we introduce the notation: $t_k^i := t_k + i\tau_1$, $f_k^i := f(t_k^i) = f(\phi(t_k^i - \tau_1) + \phi(t_k^i - \tau_2) + \gamma J(t_k^i)) \quad \forall k \in \{1 \dots N\}$ and $\phi_0^i := \phi(t_N^{i-1})$, $f_0^i = f_N^{i-1}$. Now, Equation (5) can be written as

$$v_k^i := x(t_k^i) \approx \phi_0^i e^{-\alpha k\theta} + \frac{\theta}{2}\sum_{j=1}^{k} e^{\alpha(j-k)\theta}(f_j^i + e^{-\alpha\theta}f_{\{j-1\}^i}). \quad (6)$$

We refer to v_k^i as the value of virtual node k during cycle i. Notice that in Equation (6), v_k^i only depends on ϕ evaluated at support points of previous cycles because τ_1 and τ_2 are integer multiples of θ. Equation (6) can thus be viewed as an update equation that derives the state of the virtual nodes in cycle i from the state of virtual nodes in previous cycles. This becomes apparent when written in matrix form:

$$\boldsymbol{v}^i = \mathbf{b}\left(\frac{\theta}{2}f_0^i + \phi_0^i\right) + \frac{\theta}{2}\mathbf{C}\mathbf{f}^i, \quad (7)$$

$$\text{where } \mathbf{b} = \begin{bmatrix} e^{-\alpha\theta} \\ e^{-\alpha 2\theta} \\ \vdots \\ e^{-\alpha N\theta} \end{bmatrix}$$

$$\text{and } \mathbf{C} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 2e^{-\alpha\theta} & 1 & & \vdots \\ \vdots & & \ddots & 0 \\ 2e^{-\alpha(N-1)\theta} & \cdots & 2e^{-\alpha\theta} & 1 \end{bmatrix} \text{ and } \mathbf{f}^i = \begin{bmatrix} f_1^i \\ f_2^i \\ \vdots \\ f_N^i \end{bmatrix}$$

Here, $\mathbf{b}$ captures the influence of the initial value for each cycle given by the last value of the previous cycle. The matrix $\mathbf{C}$ describes an exponential mixing of non-linearly transformed, previous virtual node values and input as given by $\mathbf{f}^i$. There are two different ways in which a virtual node may be dependent on different virtual nodes in previous cycles. First, when two delays are used, different virtual nodes v_i^k and v_j^m are linearly combined and then used as input to the nonlinear function f. We refer to this as mixing via *explicit dependency*. Second, the matrix $\mathbf{C}$ describes a linear mixing of these nonlinearly transformed values. It corresponds to a causal exponential filter parametrized by the rate of decay $\alpha\theta$ and is referred to as mixing via exponential smoothing.

To illustrate the effect of the exponential decay, consider two extreme examples. First, assume α approaches zero, corresponding to a system without decay term. The vector $\mathbf{b}$ then approaches unity, resulting in a constant offset of the state within each cycle. The matrix $\mathbf{C}$ becomes the numerical integral operator of the trapezoidal rule. This leads to nearby virtual nodes (i.e., virtual nodes with indices close to each other) exhibiting similar states due to continuity of the integral and thus less variability of the DCR state within each cycle.

Now, assume α approaches infinity. The vector $\mathbf{b}$ vanishes, and $\mathbf{C}$ converges to the identity matrix. The update equation (7) thus reduces to

$$\boldsymbol{v}^i = \frac{\theta}{2}\mathbf{f}^i. \quad (8)$$

In this case, the state can be discontinuous allowing for large variability within each cycle. However, mixing of virtual nodes can only be achieved if f_k^i explicitly depends on multiple virtual nodes. Here, this is realized via the second delay that introduces explicit dependencies on other nodes. Both types of mixing, exponential smoothing and explicit dependencies, can contribute to the computational capabilities of the DCR. As it constitutes the principle and only means of mixing in the single delay case, the former effect has previously been described [34]. Here, we present an investigation into the latter type of mixing with explicit dependencies.

Node dependency structure in a two delay DCR

In this section, to isolate the effect of mixing via two delay terms, we assume $\alpha \to \infty$ and analyze Equation (8) in additional detail. To this end, we describe the set of virtual nodes v_l^j in previous cycles $j < i$ on which a given virtual node v_k^i in cycle i depends. As a consequence of the first delay τ_1, v_k^i explicitly depends on v_k^{i-1}, since τ_1 equals the length of the cycle.

In the simplest case, previously studied in [35], τ_2 is an integer multiple of τ_1; here, $\tau_2 = 2\tau_1$, and v_k^i is additionally explicitly dependent on v_k^{i-2}. Since virtual node v_k^{i-1} itself already depends on v_k^{i-2}, this introduces no new

dependencies on other virtual nodes. In the absence of mixing via exponential smoothing, the virtual nodes are decoupled and only depend on their own history. We expect no qualitative improvement in the performance of the DCR, since the second delay does not allow for mixing of more virtual nodes than the one delay case. However, a minor performance increase attributable to an increased history dependence of each individual node is possible.

More generally, consider a delay τ_2 that is not necessarily an integer multiple of τ_1. The virtual node v_k^i then explicitly depends on v_k^{i-1} and v_l^j for some $j < i$ and $l \neq k$, which in turn depends on v_l^{j-1} and v_p^m for some $m < j$ and $p \neq l$. Continuing this scheme iteratively yields a dependency set $D_k = \{k + qd \bmod N, \text{ for } q \in \mathbb{N}\}$ of indices of virtual nodes, the values of which v_k^i depends on in previous cycles. Here, the distance d between virtual nodes indexed in D_k is given by the greatest common divisor $d = \mathrm{GCD}(\tau_1, \tau_2)$, and D_k thus corresponds to the set of residuals of $k \bmod d$. Consequently, any given virtual node with index k is affected by the history of $|D_k| = N/d$ virtual nodes. For delays $\tau_1 = N\theta$ and $\tau_2 = M\theta$, this implies that $|D_k| = N/[\mathrm{GCD}(N,M)]$ is at its minimal value 1 if M is an integer multiple of N and at its maximum value N if M and N are coprime. The number of virtual nodes that are mixed together within the history of each virtual node can thus be controlled by the choice of the second delay relative to the first. Depending on the task, we thus expect the choice of τ_2 and the resulting size of the dependency set to affect the performance of the DCR. See the bottom half of Figure 1 for an illustration of how the set D_k can be traced back in time across cycles.

The mask M, defined above as a τ_1-periodic function, is evaluated only at the positions of the virtual nodes. Therefore, it can be identified with the vector $\mathbf{m}$ with elements $m_k := M(t_k^i)$ for arbitrary cycles i, where each mask value can be associated with the corresponding virtual node v_k. Due to $\mathbf{m}$ being two-valued, it separates the virtual nodes into two classes defined by the mask value associated with them. When mixing virtual nodes via the second delay term $\tau_2 = \tau_1 + \varepsilon\theta$, $\varepsilon \in \mathbb{N}$, the autocorrelation of $\mathbf{m}$ at shift ε indicates the degree to which nodes of both classes are mixed. A low autocorrelation implies an equal mixing of virtual nodes belonging to both classes within each cycle, whereas a high autocorrelation implies that virtual nodes will be predominantly mixed with the same class. The same methodology can be applied to analyze the interaction of the mask and mixing via exponential smoothing. In this case, the autocorrelation at shifts close to zero must be considered. We propose that a lower autocorrelation of the mask at relevant shifts increases the complexity of the mixing and may thus be beneficial for the DCR. When we use the term "beneficial," we refer to allowing the DCR state to realize a more complex encoding of the input, benefiting the computational power of the DCR, or leading to systems with particularly interesting properties from a research perspective. We leave an analysis of other possible choices of mask values—be it single-valued (corresponding to no masking at all), or two-valued, multi-valued, or real-valued—to future work.

Notes on implementation

Here, we use a Mackey-Glass DCR, where the non-linearity $f(\phi(t-\tau_1) + \phi(t-\tau_2) + \gamma J(t))$ is given by $f(x) = \eta \frac{x}{1+x^\rho}$. For all experiments presented here, the parameters are $\eta = 0.4$, $\gamma = 0.05$ and $\rho = 1$. The size of the DCR used is fixed throughout the experiments with only τ_2 varying. We choose $\tau_1 = 480$ and $\theta = 0.6$, resulting in $N = 800$ virtual nodes. The results presented here are robust against changes in these parameters that have been chosen simply to enable a good comparison between different cases.

In each task, linear least squares regression is used to train a linear model to estimate the task-dependent target values from virtual node activity.

Results

The benchmark used in this paper to evaluate memory and non-linear computing capabilities of the system is to model a nonlinear autoregressive moving average (NARMA) time-series in response to a random, uniformly distributed input $u(t) \sim \mathcal{U}_{[0,\,0.5]}$. We use a NARMA-10 variant, where the target response is given by

$$y(t) = 0.3y(t-1) + 0.05y(t-1)\sum_{d=1}^{10} y(t-d) + 1.5u(t-1)u(t-10) + 0.1. \tag{9}$$

The DCR must model non-linear dependencies on the history of $y(t)$ and $u(t)$. In order to increase the requirement on memory in the system (i.e., the ability of the system to retain information about previous input), we also evaluate lagged NARMA-10 variants, where the target output is shifted in time: $y(t) = y(t-1)$. The input $u(t)$ remains not shifted. We evaluate the performance with the normalized root-mean-square error (NRMSE):

$$nrmse(y, \hat{y}) = \sqrt{\frac{\sum_n (y - \hat{y})^2}{n\ var(y)}}. \tag{10}$$

Effects of the second delay

In a first setup, we systematically vary the second delay and average the NRMSE of the model predictions on the NARMA-10 task for each value of the second delay across 50 trials. We choose the second delay with $\tau_2 = \tau_1 + \varepsilon\theta$ with $\varepsilon \in [0 \ldots 2N]$. The average NRMSE evaluated on validation data is shown in **Figure 2**. We identify three effects. The most prominent is an average

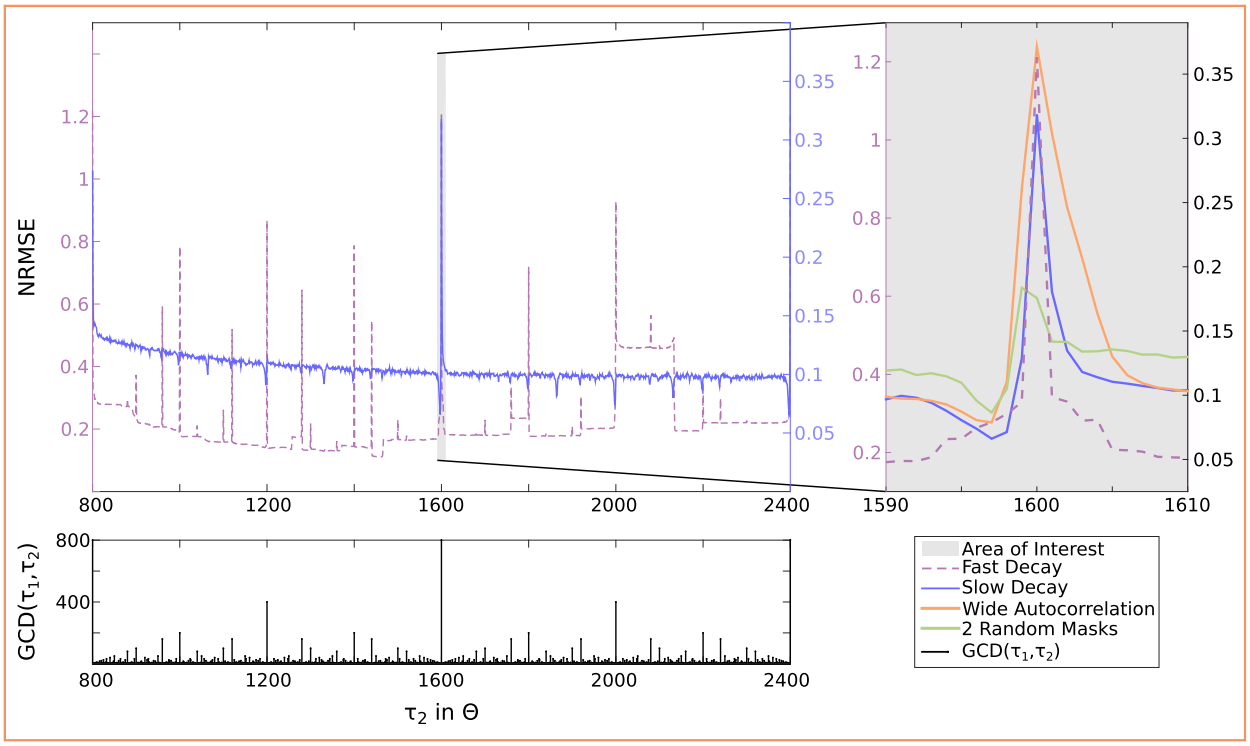


Figure 2

Dependency of the two-delay DCR performance on the choice of the second delay. Depicted in blue is the NRMSE between a target NARMA-10 time-series and the correspondingly trained linear readout of the DCR when varying the second delay τ_2 with a slow decay rate $\alpha = 1$. In purple, we repeat the experiment with a very fast decay rate $\alpha = 1000$ to isolate mixing due to explicit dependencies. On a separate scale below, we plot the GCD between the two delays. Notice that, especially for the very fast decay rate, the GCD is large exactly where performance on the prediction task is poor. The inset around $\tau_2 = 2\tau_1$ shows two additional control experiments, where the masking procedure has been altered. In green, we show the NRMSE achieved with two sequential random masks that do not correlate. A mask with a particularly wide autocorrelation profile is used to compute the green error curve. Here, the influence of autocorrelation on the shape of the error peak is demonstrated, as well as the lack of asymmetry for a system with very fast decay.

increase in performance (i.e., a decrease in the NRMSE) as τ_2 increases. This can be attributed to an increase in the effective memory of the reservoir due to injection of further delayed activity. Next, we observe a large increase in prediction error at a narrow peak around $\tau_2 = \tau_1 + N\theta = 2\tau_1$. Third, the lowest error is consistently achieved at $\tau_2 = \tau_1 + (N-3)\theta$. To a lesser degree, the last two effects systematically appear also for some rational quotients τ_2/τ_1 and become more pronounced with increasing rate of decay. These results are in accordance with the predictions derived above, in particular since the optimal delay $\tau_2 = 1597$ is the largest prime below $2\tau_1 = 1600$ and thus coprime to τ_1. In Figure 2 at bottom, we also provide a plot of the greatest common divisor (GCD) for the tested delays to allow for a visual comparison of the qualitative (location of peaks) and quantitative (relative height of peaks) relationship between the GCD curve and the error curve.

This qualitative structure of the error curve continues for $\in [N \ldots 2N \ldots]$. For the NARMA-10 task presented here, the first effect of an increase in effective memory vanishes during the interval $\varepsilon \in [N, 2N]$ and minimal values of the error curve never achieve error rates as low as $\tau_2 = 1597$. As explained in further analysis of our results, we expect this specific optimal choice of τ_2 to be attributable to a specific memory increase fitting the particular task.

Analysis at critical choices of second delay

As described above, two of the three effects that the choice of the second delay can have are most prominent around values of τ_2 where the GCD of τ_1 and τ_2 is large. To examine these critical choices of the second delay, we present three control experiments and illustrate performance around $\varepsilon \in [N-20 \ldots N+20]$ in the right inset of Figure 2.

First, the shape of the error peak is explained by the causal exponential smoothing in the update equation. It

leads to non-linear mixing of virtual nodes with a delay slightly larger than τ_1. As described earlier, choosing a second delay slightly larger than $2\tau_1$ has the same effect, and we do not introduce new mixing into the system. Choosing a second delay τ_2 slightly smaller than τ_1, however, introduces additional mixing of later nodes in previous cycles. It complements the exponential filter and results in an effect akin to non-causal filtering around each node. Notice how in Figure 2 the asymmetry vanishes when a fast decay rate is chosen.

A mask with a wide autocorrelation profile leads to a wide error peak around $\varepsilon = N$ whereas a low autocorrelation profile achieved by alternating between two different masks for even and odd cycles leads to a shallower peak. Comparing a DCR with very fast exponential decay and slow exponential decay, we observe the predicted sharp peak in the error for $\tau_2 = 2\tau_1$ (see Figure 2, inset). This supports the hypothesis that the interplay between the mask's autocorrelation and exponential smoothing influences the complexity of mixing in the system, and thus performance.

Second, we discuss the valley in the error curve occurring to the left of the error peak for a system with slow exponential decay (see blue line in the inset of Figure 2). From our discussion of the GCD, the position of this valley is intuitive. The GCD of $\tau_1 = N\theta$ and $\tau_2 = \tau_1 + (N - \beta)\theta$ must be smaller than β, resulting in consistently small GCDs for small β. The interaction with the exponential filter then leads to mixing of neighboring nodes for which the GCD is small. This mixing of small neighborhoods leads to neighboring nodes exhibiting similar states. Additionally, due to the large dependency set resulting from this mixing, changes in one node propagate through all other nodes over long periods of time. The fact that the two delay periods are almost but not quite integer multiples of each other therefore leads to an effect analogous to beat phenomena, where slower oscillations emerge from superposition of two similar oscillations. In preliminary experiments, we find stronger, slower components in the Fourier power spectrum of virtual node states corresponding to smoother transitions between values of virtual nodes. The performance benefit of this effect is task dependent, as it trades a longer history dependence for the volatility of the system state. For example, there is no benefit in choosing $\varepsilon = N - 3$ for different *n*-bit parity tasks (results not shown here). The intuition here is that information about the history beyond the window, which the recurrent Equation (9) depends on, can make prediction of the NARMA-10 time-series easier. The *n*-bit parity task however is only dependent on the exact window where the parity is computed and thus stands to benefit more from complex transformations of the recent input history rather than on longer memory. Further analysis of how different tasks can benefit from the described effect is left to future work.

Increase in performance for memory dependent non-linear tasks

In **Figure 3**, we show the increased performance of two different two-delay DCRs compared to a single-delay DCR compatible with earlier work [20] on lagged NARMA-10 prediction tasks. We choose the second delay as $\tau_2 = \tau_1 + (N - 3)\theta$ and $\tau_2 = \tau_1 + N\theta$ corresponding to what we predict to be an optimal and worst-case choice respectively. In the optimal case, the second delay leads to a large and consistent performance increase over both a single-delay DCR and a worst case two-delay DCR. The reported average NRMSE for the optimal choice is 0.065 with a variance of 0.0015 on the no lag prediction task, increasing performance by 63.5% over the single-delay DCR. The poor performance of the worst-case two-delay DCR, despite the second delay being very close to the optimal case, highlights the importance of choosing additional delays correctly.

Introducing lag in the prediction task raises the task difficulty, as the increase in average NRMSE for the single-delay DCR illustrates. Comparatively, the optimal two-delay DCR shows very slow deterioration for increasing lag, revealing the significant memory increase achieved.

Discussion

We have presented a mathematical analysis of discretized DCRs and dependency structures introduced by adding a second delayed feedback signal. We have demonstrated how this analysis presents a tool to choose a second delay, effectively resulting in greatly improved performance on standard benchmarks. One limitation of the analysis based on greatest common dividers between two delays is its restriction to time-discrete systems. An additional side effect of the discrete nature of the current work is the extreme sensitivity of the GCD to small changes in the delays. Consider for example a system where $\tau_1 = p\theta$ and p is prime. The GCD is now insensitive to the choice of the second delay, as all natural numbers except multiples are coprime to p. Simply changing the systems to $\tau_1 = (p + 1)\theta$ will dramatically change the sensitivity towards the second delay. This holds particularly true for system with very fast exponential decay. In future work, we aim to extend the understanding developed here to the realm of time continuous systems to help guide the development of a larger variety of delay-coupled computing systems.

We also developed a better understanding of the effect masks have on DCRs. We show how the autocorrelation structure of the mask determines whether nodes of the same or different classes are mixed and demonstrate that this can have a large influence on performance (effect on NRMSE) in various control tasks. Further research is needed to fully understand the interaction of masking and mixing.

While we have presented large performance increases on NARMA prediction tasks, systematic study of a

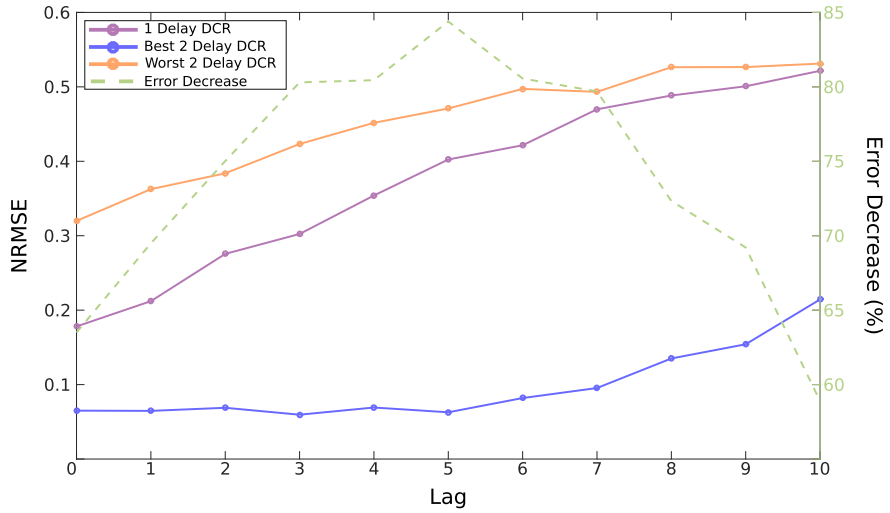


Figure 3

Single and two delay DCR performance for varying memory requirements of the task. The NRMSE evaluated on a lagged NARMA-10 task is plotted against different lags for an optimal two delay DCR with $\tau_2 = \tau_1 + (N-3)\theta$ (blue), a worst-case two delay DCR with $\tau_2 = 2\tau_1$ (orange) and a single delay DCR (purple). The relative decrease in error of the optimal two-delay DCR over the single delay DCR is presented in green. The significant increase in memory of the system in the optimal case compared to the single delay DCR becomes apparent.

task-dependent trade-off between history and non-linear computation is needed. In this context, remaining model parameters, such as the decay rate in combination with multiple delays, should also be investigated more closely. This will further the understanding of the computational power of complex delay-coupled systems and allow for easier use on real-world tasks.

In hardware implementations, physical properties of the system constrain the rate of decay and in turn the mixing via exponential smoothing. Here, introducing additional feedback connections offers the opportunity to influence the degree of mixing via explicit dependencies. In turn, this methodology allows control of the inertia vs. the volatility of the system. Thus, the effects presented here can help guide the design of DCRs realized in dedicated hardware.

Conclusion

DCR computing represents a computational paradigm that lends itself particularly well to implementation in hardware and can offer a different perspective on delays in physical and biological computing systems. For example, it demonstrates that in neuromorphic hardware, and potentially the brain, delayed feedback signals can allow simple non-linear elements to exhibit complex history-dependent behavior. Here, we have explained how different effects due to the inertia of the system, input masking, and the precise choice of a second delay can be isolated and explained separately. In particular, the choice of the second delay has tremendous consequences on the computational capabilities of the DCR. We have developed a mathematical framework within which an informed choice of the second delay is possible, allowing the single node system to perform on par with large spatially distributed reservoir computers in complex time-series prediction tasks. We conclude that multi-delay-coupled systems present an attractive paradigm for the development of new neuromorphic hardware for computation in non-von-Neumann architectures.

References

1. P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B.L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. D. Flickner, W. P. Risk, R. Manohar, and D. S. Modha, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 668, p. 668–572, 2014.
2. J. Schemmel, J. Fieres, and K. Meier, "Wafer-scale integration of analog neural networks," *in Proc. IEEE Int. Joint Conf. Neural Netw.*, 2008, pp. 431–438
3. S. Furber and S. Temple, "Neural systems engineering," *J. Roy. Soc. Interface*, vol. 4, pp. 193–206, 2006.
4. A. Rast, F. Galluppi, S. Davies, L. Plana, C. Patterson, T. Sharp, D. Lester, and S. Furber, "Concurrent heterogeneous neural model simulation on real-time neuromimetic hardware," *Neural Netw.*, vol. 24, no. 9, pp. 961–978, 2011.
5. R. Vincente, L. L. Gollo, C. R. Mirasso, I. Fischer, and G. Pipa, "Dynamical relaying can yield zero time lag neuronal synchrony despite long conduction delays," *in Proc. Nat. Academy Sci.*, vol. 105, no. 44, pp. 17157–17162, 2008.
6. T. Fukuda, T. Kosaka, W. Singer, and R. A. W. Galuske, "Gap junctions among dendrites of cortical GABAergic neurons establish a dense and widespread intercolumnar network," *J. Neurosci.*, vol. 26, no. 13, pp. 3434–3443, 2006.

7. Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time-series," in *The Handbook of Brain Theory and Neural Networks*, Cambridge, MA, USA: MIT Press, 1995.
9. G. E. Hinton, S. Osindero, and Y.W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, pp. 1527–1554, 2006.
9. S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
10. Y. LeCun, Y. Bengio, and G. E. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.
11. W. Maas, "Networks of spiking neurons: The third generation of neural network models," *Neural Netw.*, vol. 10, no. 9, pp. 1659–1671, 1997.
12. W. Maas, T. Nachtschlaeger, and H. Markram, "A new framework for neural computation based on perturbations," *Neural Comput.*, vol. 14, no. 11, pp. 2531–2560, 2002.
13. H. Jaeger, "The "echo state" approach to analyzing and training recurrent neural networks," German Nat. Res. Center Inf. Technol., Sankt Augustin, Germany, GMD Tech. Rep. 148, vol. 147, 2001
14. M. Lukosevicius and H. Jaeger, "Reservoir Computing approaches to recurrent neural network training," *Comput. Sci. Rev.*, vol. 3, no. 3, pp. 127–149, 2009
15. L. Buesing, B. Schrauwen, and R. Legenstein, "Connectivity, dynamics, and memory in reservoir computing with binary and analog neurons," *Neural Comput.*, vol. 22, no. 5, pp. 1272–1311, 2010.
16. D. Kudithipudi, Q. Saleh, C. Merkel, J. Thesing, and B. Wysocki, "Design and analysis of a neuromemristive reservoir computing architecture for biosignal processing," *Front. Neurosci.*, vol. 9, 2016, Art. no. 502.
17. M. S. Kulkarni, "Memristor-based reservoir computing," in *Proc. IEEE/ACM Int. Symp. Nanoscale Architect.*, 2012, pp. 226–232.
18. M. L. Alomar, V. Canals, N. Perez-Mora, V. Martinez-Moll, and J. L. Rossello, "FPGA-based stochastic echo state networks for time-series forecasting," *Comput. Intell. Neurosci.*, vol. 2016, 2016, Art. no. 15.
19. H. O. Sillin, R. Aguilera, H. Shieh, A. V. Avizienis, M. Aono, A. Z. Stieg, and J. K. Grimzewski, "A theoretical and experimental study of neuromorphic atomic switch networks for reservoir computing," *Nanotechnology*, vol. 24, no. 38, 2013, Art. no. 384004.
20. N. D. Haynes, M. C. Soriano, D. P. Rosin, I. Fischer, and D. J. Gauthier, "Reservoir computing with a single time-delay autonomous Boolean node," *Physical Rev. E*, vol. 91, no. 2, p. 020801, 2016.
21. L. Appeltant, M. C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. R. Mirasso, and I. Fischer, "Information Processing using a single dynamical node as a complex system," *Nature Commun.*, vol. 2, 2011, Art. no. 468.
22. S. Ortin, M. C. Soriano, L. Pesquera, D. Brunner, D. San-Martin, I. Fischer, C. R. Mirasso, and J. M. Gutierrez, "A unified framework for reservoir computing and extreme learning machines based on a single time-delayed neuron," *Sci. Rep.*, vol. 5, 2015, Art. no. 14945.
23. W. Maas and H. Markram, "On the computational power of recurrent circuits of spiking neurons," *J. Comput. Syst. Sci.*, vol. 69, no. 4, pp. 593–616, 2004.
24. A. Lazar, G. Pipa, and J. Triesch, "SORN: A self-organizing recurrent neural network," *Front. Comput. Neurosci.*, vol. 3, 2009, Art. no. 23.
25. M. Lukosevicius, "On self-organizing reservoirs and their hierarchies," School Eng. Sci., Jacobs Univ. Bremen, Bremen, Germany,Tech. Rep. No. 25, 2010. [Online]. Available: http://techreports.user.jacobs-university.de/files/25_2396_Lukosevicius10.pdf
26. P. Zheng and J. Triesch, "Robust development of synfire chains from multiple plasticity mechanisms," *Front. Comput. Neurosci.*, vol. 8, 2014, Art. no. 66.
27. D. Sussillo, and L. F. Abbott, "Transferring learning from external to internal weights in echo-state networks with sparse connectivity," *PLoS One*, vol. 7, no. 5, 2012, Art. no. e37372.
28. H. Jaeger, "Controlling recurrent neural networks by conceptors," arXiv: 1403.3369v1 [cs.NE], Mar. 13, 2014. [Online]. Available: https://arxiv.org/abs/1403.3369
29. J. Schumacher, H. Toutounji, and G. Pipa, "An analytical approach to single node delay-coupled reservoir computing," in *Proc. 23rd Int. Conf. Artif. Neural Netw.*, 2013, pp. 26–33.
30. M. C. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, no. 4300, pp. 287–289, 1977
31. Y. Paquot, F. Duport, A. Smerieri, J. Dambre, B. Schrauwen, M. Haelterman, and S. Massar, "Optoelectronic reservoir computing," *Sci. Rep.*, vol. 2, 2012, Art. no. 287.
32. D. Brunner, M. C. Soriano, C. R. Mirasso, and I. Fischer, "Parallel photonic information processing at gigabyte per second data rates using transient states," *Nature Commun.*, vol. 4, 2013, Art. no. 1364.
33. K. Vandoorne, P. Mechet, T. Van Vaerenbergh, M. Fiers, G. Morthier, D. Verstraeten, B. Schrauwen, J. Dambre, and P. Bienstman, "Experimental demonstration of reservoir computing on a silicon photonics chip," *Nature Commun.*, vol. 5, 2014, Art. no. 3541.
34. J. Schumacher, H. Toutounji, and G. Pipa, "An introduction to delay-coupled reservoir computing," in *Artificial Neural Networks*. Berlin, Germany: Springer, 2015, pp. 63–90.
35. L. Appeltant, "Reservoir Computing based on delay-dynamical systems," Ph.D. dissertation, Vrije Universiteit Brussel/ Universitat de les Illes Baleros, 2012.

Received May 15, 2016; accepted for publication June 11, 2016. Date of current version May 5, 2017.

Pascal Nieters *Institute of Cognitive Science, Neuroinformatics, University Osnabrueck, Osnabrueck* 49074*, Germany (pnieters@uos.de).* Mr. Nieters is currently studying for his Ph.D. degree at the Neuroinformatics lab at the Institute for Cognitive Science at University Osnabrueck, Germany. He holds an M.Sc. degree in cognitive science from University Osnabrueck and was heavily involved in a student-led cognitive computing study project with IBM that uses data analysis methods to model the spread of infectious diseases. His research interest lies in simple mathematical models of neural computation in domains that range from single neurons to recurrent networks and complex macro-networks.

Johannes Leugering *Institute of Cognitive Science, Neuroinformatics, University Osnabrueck, Osnabrueck 49074, Germany (jleugeri@uos.de).* Mr. Leugering is currently a Ph.D. student supervised by Prof. Pipa and a research assistant in the Neuroinformatics lab at the Institute for Cognitive Science at University Osnabrueck. He graduated with a degree in cognitive science in 2015 and is currently pursuing a second degree in mathematics. His research interests lie in the mathematical modeling of complex stochastic systems and the analysis of their means of information processing. A secondary research interest is the development of efficient mobile data analysis for a biomedical data acquisition device.

Gordon Pipa *Institute of Cognitive Science, Neuroinformatics, University Osnabrueck, Osnabrueck 49074, Germany (gpipa@uos.de).* Prof. Pipa is currently chair of the Neuroinformatics Lab at the Institute of Cognitive Science at Osnabrück University, Germany. He started this position after research positions at the Max Planck Institute for Brain Research in Frankfurt, and the Department of Brain and Cognitive Sciences at MIT. He studied physics with a focus on complex systems and statistical physics, holds a Ph.D. degree in computer science and Habilitation in biology. Additionally, he holds several patents in the domain of neuro-inspired image processing.

ARTICLE Communicated by Jochen Triesch

A Unifying Framework of Synaptic and Intrinsic Plasticity in Neural Populations

Johannes Leugering
jleugeri@uos.de
Gordon Pipa
gpipa@uos.de
Neuroinformatics Group, Institute of Cognitive Science, Osnabrück University, D-49069 Osnabrück, Germany

A neuronal population is a computational unit that receives a multivariate, time-varying input signal and creates a related multivariate output. These neural signals are modeled as stochastic processes that transmit information in real time, subject to stochastic noise. In a stationary environment, where the input signals can be characterized by constant statistical properties, the systematic relationship between its input and output processes determines the computation carried out by a population. When these statistical characteristics unexpectedly change, the population needs to adapt to its new environment if it is to maintain stable operation. Based on the general concept of homeostatic plasticity, we propose a simple compositional model of adaptive networks that achieve invariance with regard to undesired changes in the statistical properties of their input signals and maintain outputs with well-defined joint statistics. To achieve such invariance, the network model combines two functionally distinct types of plasticity. An abstract stochastic process neuron model implements a generalized form of intrinsic plasticity that adapts marginal statistics, relying only on mechanisms locally confined within each neuron and operating continuously in time, while a simple form of Hebbian synaptic plasticity operates on synaptic connections, thus shaping the interrelation between neurons as captured by a copula function. The combined effect of both mechanisms allows a neuron population to discover invariant representations of its inputs that remain stable under a wide range of transformations (e.g., shifting, scaling and (affine linear) mixing). The probabilistic model of homeostatic adaptation on a population level as presented here allows us to isolate and study the individual and the interaction dynamics of both mechanisms of plasticity and could guide the future search for computationally beneficial types of adaptation.

Neural Computation **30, 945–986** (2018)
doi:10.1162/NECO_a_01057

1 Introduction

When we talk about adaptation, we take it to mean what W. Ross Ashby had in mind when writing his seminal book *Design for a Brain*, where he argued that in a volatile environment, "'adaptive' behavior is equivalent to the behavior of a stable system" (Ashby, 1954, p. 64). Active dynamical mechanisms that stabilize the activity of neural populations in spite of sudden changes in sensory inputs, lesions, or rewiring of synaptic connections have been studied extensively under the general term *homeostatic plasticity*. Some of them are confined within individual neurons, generally referred to as *intrinsic plasticity*; others operate within synapses and are thus grouped under the umbrella term of *synaptic plasticity*.

Here we aim to unify such forms of plasticity in a single mathematically simple framework of continuous-time stochastic processes that enables us to analyze their distinct functional roles and interactions and allows us to extend the notion of homeostatic plasticity from the level of individual neurons to that of populations.

A wide variety of specific candidate mechanisms of synaptic and intrinsic plasticity has been studied extensively, from both biological as well as information theoretical perspectives: Bienenstock, Cooper, and Munro (1982) proposed a form of synaptic plasticity that determines the growth or decay of synaptic connections under the constraint of maintaining a fixed mean firing rate. Turrigiano and Nelson (2004) conjectured a role of homeostatic plasticity in stabilizing the transmission of information in feedforward networks by fine-tuning the balance between excitatory and inhibitory connections. Both approaches assume a self-regulating form of synaptic adaptation that renders a neuron population invariant to additive shifts in its inputs, ensuring that resulting mean firing rates remain well within physiological bounds. In the abstract, dynamical systems models of biological spiking neurons, spike rate adaptation effects were incorporated by slowly changing adaptation variables (Izhikevich, 2003) and spike-induced responses (Jolivet, Lewis, & Gerstner, 2004) to closely match experimental measurements. In these models, a neuron's adaptation to its spiking output, rather than its input, is the driving force of homeostatic plasticity.

Scale invariance was proposed via a form of plasticity referred to as synaptic scaling (Turrigiano, 2008) or gain control (Burrone & Murthy, 2003), where synaptic connection strength or neural excitability is regulated, such that the variance of membrane potentials remains fixed. Multiple timescales of such adaptation were observed, potentially serving different purposes, that jointly improve information transmission (Fairhall, Lewen, Bialek, & de Ruyter Van Steveninck, 2000). Synaptic depression (Abbott, Varela, Sen, & Nelson, 1997) and diffusion of neurotransmitters (Sweeney, Kotaleski, & Hennig, 2015) were suggested as two fast-acting candidate mechanisms that could achieve scale invariance in order to

increase the dynamic range of a neuron's output and thus improve its ability to transmit information.

The concept of a neuron as a bottleneck of information transmission (Bell & Sejnowski, 1995; Stemmler & Koch, 1999) expanded this notion and offered an information-theoretical explanation for the utility of homeostatic adaptation. In this framework, the maintainance of stable characteristics of a neuron's output, in spite of changes in the characteristics of its input, allows a neuron to discover and encode information about its inputs in a stable representation, thus making a population more robust to environmental changes, noise, or spike timing variability (Buesing & Maass, 2010). Intrinsic plasticity mechanisms were proposed to tune the nonlinear response function of neurons to optimize properties of their outputs, such as the maximization of information transmission (Toyoizumi, Pfister, Aihara, & Gerstner, 2005) or the minimization of divergence from a desired stationary distribution (Savin, Joshi, & Triesch, 2010; Triesch, 2007). Their interaction with synaptic plasticity and synaptic scaling was analyzed (Toyoizumi, Kaneko, Stryker, & Miller, 2014) and shown to yield emerging properties, such as the ability to implement blind source separation on prewhitened inputs in simple model neurons (Buesing & Maass, 2010; Savin et al., 2010; Triesch, 2007), a feat observable in vitro as well (Isomura, Kotani, & Jimbo, 2015). Similar results were obtained by Hyvärinen and Oja (1998), who used synaptic scaling in combination with simple nonlinear Hebbian learning rules to discover independent components.

These results provide crucial insight into the capabilities and limitations of neural plasticity and serve as the basis of this article. We contribute to this field of theoretical research by unifying the information-theoretical concept of intrinsic plasticity, enforcing a stable, fixed distribution of activations in the face of changing input statistics and Hebbian synaptic plasticity within an abstract but simple, probabilistic, and compositional model of adaptive neuron populations that avoids several limitations of the approaches discussed above. The model's activation function is directly parameterized by the membrane potential statistics that neurons should adapt to. Thus, the often complex update rules proposed in models of intrinsic plasticity are replaced by causal (nonlinear) filters of the membrane potential, such that adaptation dynamics can be analyzed and convergence ensured. Since our neuron model is fully determined by the stochastic properties of the neuron's membrane potential and activation processes, it could be easily adjusted to experimental data observed in vivo.

The continuous-time nature of our model facilitates studying the interaction between neural dynamics, synaptic dynamics, and plasticity and makes it easier to reconcile with its biological counterpart. By restricting plasticity to intrinsic and Hebbian synaptic plasticity while excluding global mechanisms such as a detailed balancing of excitation and inhibition or synaptic scaling, our model makes use only of information locally available to neurons and synapses, respectively. In light of Ashby (1954), we

view many of the suggested benefits of plasticity as instances of the more general principle of homeostatic self-regulation, such that, for example, the purpose of blind source separation becomes, first and foremost, to find an informative, transformation-invariant representation of a population's input. We thus aim to elevate the notion of homeostatic adaptation from the level of individual neurons to the level of populations through the interplay of intrinsic and synaptic plasticity.

To illustrate the capabilities of the model, we reproduce results by Savin et al. (2010) and theoretically analyze the complementary role that intrinsic plasticity plays in stabilizing Hebbian learning, thus allowing individual neurons to discover informative components of their input signals. We demonstrate the generality of these results in a network trained on image patches, where lateral decorrelation drives neurons to learn a transformation-invariant representation of their multivariate inputs by implementing a form of principal or independent component analysis.

2 An Adaptative Network Model

The adaptive network model comprises adaptive neurons and adaptive synapses. The intrinsic plasticity mechanism implemented within each neuron uses locally available information about statistical properties of the neuron's membrane potential to adapt its behavior, such that its output remains stationary with predetermined statistics. Hebbian synaptic plasticity uses the product of (a function of) activations of a synapse's presynaptic source and postsynaptic target to update its connection strength. Correlated activity between source and target thus drives synaptic growth. By combining the adaptation of the marginal statistics via intrinsic plasticity with the adaptation of the copula via synaptic plasticity, the adaptive network realizes plasticity of its multivariate joint outputs and can become invariant to a large range of changes in its input statistics.

A population of neurons can thus be seen as a computational unit that receives a multivariate stochastic process as its input and linearly transforms and projects it on a set of adaptive neurons via adaptive synaptic connections. Each neuron then nonlinearly transforms its marginal input, and the joint activations of these neurons are taken to be the multivariate output of the population that subsequently becomes the input to the same or another population. (See Figure 1 for an illustration of the adaptive network model.)

We adhere to a simplistic yet powerful class of linear-nonlinear models (Ostojic & Brunel, 2011) that separates the dynamics of a neuron into two components: a linear, spatiotemporal filtering of inputs and a nonlinear transformation thereof, which yields the instantaneous firing intensity of the neuron. Although beyond the scope of this work, the model can be further extended using a spike train point process that samples spikes according to the neuron's time-varying firing intensity. Multiple input signals are linearly combined within the neuron's dendritic tree through weighted

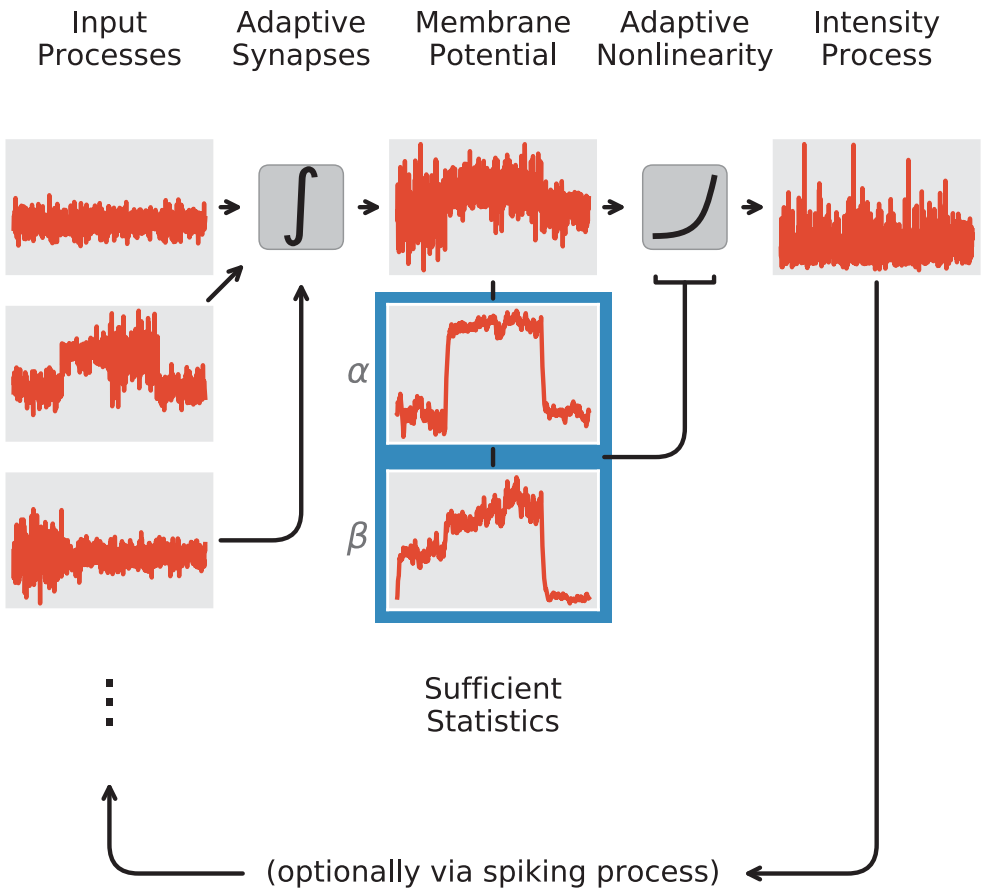


Figure 1: Schematic overview of the adaptive network model. Stochastic processes representing inputs into the population are combined via weighted, adaptive synaptic connections and integrated into membrane potential processes. Sufficient statistics of these processes are estimated in real time and used to adapt the neurons' nonlinear activation functions such that they result in stationary activation processes with predefined distributions. These activations, or sampled spike trains with accordingly time-varying intensity (not discussed here), are in turn used as inputs for other neurons.

adaptive synaptic connections and integrated into a neuron's time-varying membrane potential X_t, a real-valued stochastic process. The statistical properties of this membrane potential process are assumed to change rarely or slowly, such that the process can be locally well approximated by a (piece-wise) stationary process. The neuron's nonlinear activation function $\nu_\phi(x) \in C^2(\mathbb{R}, \mathbb{R}_+)$ is parameterized by the vector ϕ and maps its membrane potential X_t to an intensity or instantaneous firing rate $Y_t = \nu_\phi(X_t)$, also referred to as the neuron's activation or output. It follows that the activation is a stationary stochastic process as well, and for both the membrane potential and the activation, stationary distributions P_X and P_Y can be derived such that $X_t \sim P_X$ and $Y_t \sim P_Y$ (see lemma 5 in the supplementary text A.1). For the special case of an exponential function ν, this corresponds to a

continuous-time generalized linear model as proposed by Truccolo, Eden, Fellows, Donoghue, and Brown (2005).

Conversely, given both a stationary distribution of membrane potentials P_X and a desired distribution of firing rates P_Y, a nonlinearity ν can be derived that satisfies $Y_t = \nu(X_t) \sim P_Y$ for $X_t \sim P_X$ (see lemma 6). This nonlinear activation function can be parameterized by some sufficient statistics $\bar{\eta}$ of the membrane potential distribution P_X, the estimation of which is the principal task of intrinsic plasticity. By continuously estimating $\bar{\eta}$, such an adaptation mechanism can maintain the desired output distribution P_Y in spite of gradual changes in the statistics $\bar{\eta}$ of its membrane potential and thus generates a (nearly) stationary output process Y_t. In this work, spike generation is not modeled; instead, the continuous activations are used directly in a rate-coding paradigm to derive theoretical results.

2.1 Model Components. The network model described is composed of four components to be chosen independently. First, a class of stochastic processes can be chosen to model the dynamics of individual membrane potentials and thus determine their marginal membrane potential distributions. Second, the desired marginal distribution of the neurons' activation can be defined to match theoretical considerations (e.g., the exponential distribution—Triesch, 2007) or biological data (e.g., the log-normal distribution—Hromádka, DeWeese, & Zador, 2008). Third, codependency between the signals projected onto the neurons can be introduced or modified by an appropriate connectivity structure of synaptic connections. Finally, the precise mechanisms of intrinsic and synaptic plasticity can be chosen to implement invariance with respect to certain changes in the population's input statistics. Each of these modeling choices is briefly discussed next.

2.1.1 Membrane Potential Processes. With little loss of generality, we assume that the membrane potential of an idividual neuron can be modeled as a process operating on two timescales. The time-varying potential X_t is described by a stationary stochastic diffusion process of the general form

$$\mathrm{d}X_t = a(X_t)\mathrm{d}t + b(X_t)I_t\mathrm{d}t, \tag{2.1}$$

where a models the autonomous deterministic behavior of the membrane potential and b modulates the impact of the time-varying input I_t. We assume for convenience that the stationary distribution P_X of X_t is a member of an exponential family of distributions, parameterized by a vector of (minimally) sufficient statistics $\bar{\eta}$. This constraint is not particularly restrictive, since a large variety of probability distributions belong to an exponential family, for which a diffusion process with according stationary distribution can be constructed (Bibby, Skovgaard, & Sørensen, 2005). On

an orders-of-magnitude slower timescale, the statistics $\bar{\eta}$ of the membrane potential are themselves subject to changes that are not part of the model. The chosen diffusion process can be adjusted to match dynamic properties such as the membrane potential's impulse response, thus giving the modeler flexibility to choose in accordance with biological observations. For illustration purposes, we model the membrane potential as a simple leaky integrator of the form

$$\mathrm{d}X_t = \theta(\mu - X_t)\mathrm{d}t + \sigma\sqrt{2\theta}I_t\mathrm{d}t, \tag{2.2}$$

where θ is the leak rate time constant of the membrane potential, μ is the resting potential, and σ controls the sensitivity of the membrane potential to external input. When the neuron is driven by white noise (modeled as a derivative of Brownian motion) $I_t\mathrm{d}t = \mathrm{d}\mathcal{B}_t$, the resulting membrane potential resembles an Ornstein-Uhlenbeck process with gaussian stationary distribution $P_X = \mathcal{N}(\mu, \sigma^2)$ (see lemma 4). This choice of stochastic process is commonly used as a candidate model of membrane potentials (Ricciardi & Lánský, 2006). While this choice of membrane potential process is mathematically convenient, questions have been raised about the applicability of such a model to biological data (Shinomoto, Sakai, & Funahashi, 1999), and a more sophisticated choice could be made here if required. The stationary membrane potential distribution family of choice should be general enough that any relevant changes in the distribution (e.g., due to effects of learning) are captured in its parameters. In the examples presented here, input processes exhibit either stationary gaussian, Laplacian, or beta distributions.

Note that the stochastic input term I_t in the process makes no distinction between an unknown "signal" and "noise" present in the input—just the combination of both is modeled. This makes it possible to match the statistical properties to biological observations without knowledge of what is signal and what is noise in light of the neuron's computational role.

2.1.2 Activation Distributions. The stationary distribution P_Y of firing rates affects the neuron's capacity to transmit information about its membrane potential X_t and has a considerable impact on the neuron's computational role. For a fixed mean firing rate, for example, the exponential distribution maximizes the entropy of the neuron's intensity (Triesch, 2007), whereas a narrowly peaked bimodal distribution could alternate between periods of high and low firing rates, leading to more precisely timed bursts of spikes. Distributions with heavier tail probabilities turn the neuron into a coincidence detector, while others may match biological observations for certain neuron types best. When linearly decorrelated, marginally uniform activations become maximally informative (see also section 2.1.4). For a more detailed discussion of the effects, that the choice of an activation distribution can have on the neuron model, we refer readers to the

supplementary text A.6. In the following, a continuous distribution P_Y is assumed for mathematical convenience, but this is not strictly necessary.

When modeling connected neural populations, the distribution P_Y should be chosen in accordance with the stationary distribution P_X of the membrane potential process, such that a linear combination of multiple neurons' outputs, filtered by synaptic responses and integrated, results in the assumed stationary distribution of the membrane potential. However, when a gaussian process is chosen for the membrane potential, this may be neglected for a large number of synaptic inputs due to the central limit theorem and the combined smoothing effect of the synaptic and membrane potential spike response. For the examples presented here, either log-gaussian or log-Laplacian stationary activation distributions are used.

2.1.3 Intrinsic Plasticity. Consider a model neuron as described above with a known membrane potential process X_t that has the stationary exponential family distribution P_X parameterized by sufficient statistics $\bar{\eta} = \mathbb{E}[\eta(X_t)]$, where η is some continuous function of the random variable X_t. By filtering the process $\phi_t = \eta(X_t)$ with a causal exponential filter, we construct an exponentially weighted running estimate $\bar{\phi}_t \approx \bar{\eta}$ of the membrane potential's sufficient statistics. The dynamic properties of this process $\bar{\phi}_t$ are derived in the supplementary text A.4. As outlined in section 2, the neuron's activation function $\nu_{\bar{\eta}} \overset{\text{def}}{=} F_Y^{-1} \circ F_X$ is chosen to map the membrane potential distribution P_X with cumulative distribution function F_X onto the desired output distribution P_Y with cumulative distribution function F_Y. By using $\bar{\phi}_t$ as an empirical estimate of the expected value $\mathbb{E}[\phi_t] = \mathbb{E}[\eta(X)] = \bar{\eta}$, the activation function $\nu_{\bar{\phi}_t}$ thus always approximates the mapping $\nu_{\bar{\eta}}$ from the current membrane potential distribution parameterized by $\bar{\eta}$ to the desired firing intensity distribution, making the neuron invariant to any (slow) changes in the parameters $\bar{\eta}$ of the membrane potential distribution. It should be stressed that this functional notion of intrinsic plasticity models the combined effect of all forms of homeostatic mechanisms within the neuron, such as spike-rate adaptation, effects due to depletion, diffusion, and aggregation of neurotransmitters, gene expression, and more. If desired, multiple adaptation variables, changing with different timescales, can be included as long as they serve as sufficient statistics of the membrane potential distribution.

2.1.4 Connectivity and Copulas. Neurons within a population may be related through shared inputs or lateral connections. To model the relationship between the signals processed by each neuron, we assume that a population of neurons receives inputs from a multivariate stochastic process, which is subsequently linearly transformed by synaptic connections, projected onto the neurons, and integrated into their membrane potentials. With an appropriate choice of weights, a sufficiently large number of

synaptic connections can induce an arbitrary covariance structure. Consequently, any change in the covariance structure of the input process could be reversed by an appropriate choice of synaptic connections (see the supplementary text B.3 for more information).

Since each neuron only (invertibly) transforms its marginal membrane potential distribution to its marginal output distribution, only neurons driven by (in)dependent inputs exhibit (in)dependent outputs. To be able to define the dependency structure independent of the chosen marginal membrane potential and output distributions, we factorize the stationary joint membrane potential distribution P_X into the marginal distributions P_{X_i} and a so-called *copula* C with density c (Embrechts, Lindskog, & McNeil, 2001). For n neurons, this allows us to express the joint distributions of a population's membrane potentials X and activations Y as follows, where f denotes a density and F a cumulative distribution function of the corresponding random variable in the subscript (see lemma 8 in the supplementary text B.1):

$$F_X(x) = F_X(x_1, \ldots, x_n) = C(F_{X_1}(x_1), \ldots, F_{X_n}(x_n)), \tag{2.3}$$

$$F_Y(x) = F_Y(x_1, \ldots, x_n) = C(F_{Y_1}(y_1), \ldots, F_{Y_n}(y_n)), \tag{2.4}$$

$$f_X(x) = c(F_{X_1}(x_1), \ldots, F_{X_n}(x_n)) \prod_{i=1}^{n} f_{X_i}(x_i), \tag{2.5}$$

$$f_Y(y) = c(F_{Y_1}(y_1), \ldots, F_{Y_n}(y_n)) \prod_{i=1}^{n} f_{Y_i}(y_i). \tag{2.6}$$

Under the effect of intrinsic plasticity, all activations Y_i are identically distributed, and the joint distribution of activations is fully determined by the desired marginal stationary activation distribution F_{Y_i} and the copula C. Note that C, which captures the relation between the neurons' joint outputs, also reflects the structure already found between their membrane potentials, which is in turn determined by the structure between the input signals driving the neurons. The copula thus captures the interrelation between the neurons independent of their activation functions or any invertible transformation thereof (see the supplementary text B.1). The copula itself is a probability distribution, the entropy of which measures the mutual information between the random variables modeled by it (Ma and Sun, 2011). Mutual independence is thus achieved when the entropy of their copula is maximized, resulting in a jointly uniform distribution, also referred to as the independence copula.

Specifically for stationary membrane potential processes with gaussian or Laplacian distributions, as used in the following, the copula is fully determined by the covariance structure of the inputs and the synaptic connection weights. A direct consequence of this is that an appropriate

choice of synaptic connection weight can be used to render the neurons fully statistically independent (see the supplementary text B.2). In the special case of marginally uniform firing rate distributions, the copula is identical to the population's joint firing rate distribution, and mere uncorrelatedness of the neurons corresponds to jointly uniform, and thus statistically independent, outputs. The choice of the marginal membrane potential distribution thus defines what aspect of the copula synaptic weights can influence, whereas the marginal output distribution defines what can be revealed about it through (linear) correlation coefficients or Hebbian learning. This conceptual separation of the marginal distributions, on which the adaptive neurons operate, and the copula, which captures the full interrelation of neurons due to their related inputs and synaptic connections, allows us to better analyze the effects of intrinsic and synaptic plasticity and their interaction.

2.1.5 Hebbian Plasticity. In the adaptive network model, a generalized rate-coding model of Hebbian plasticity (cf. the activity product rule proposed by Brown, Kairiss, & Keenan, 1990) updates synapses connecting a presynaptic source with a postsynaptic target according to a product of their respective activations. Generally, assuming a synaptic interaction delay $\tau^{(i,j)}$, the weight $W^{(i,j)}$ of a synapse that connects a source j with activation $Y_t^{(j)}$ to a target i with activation $Y_t^{(i)}$ is updated according to a multiplicative rule of the form

$$\mathrm{d}W_t^{(i,j)} = \delta(f_{\mathrm{pre}}(Y^{(j)}_{t-\tau^{(i,j)}}) \cdot f_{\mathrm{post}}(Y_t^{(i)}) - W_t^{(i,j)})\mathrm{d}t, \tag{2.7}$$

where δ is a synaptic learning rate and f_{pre} and f_{post} model the potentially nonlinear dependency of the weight updates on the pre- and postsynaptic activations. The specific choice of f_{pre} and f_{post} allows adapting the synaptic plasticity rule to the chosen marginal distribution of neural activations or tuning it to learn different nonlinear correlations. For example, by choosing f_{pre} and f_{post} to be the cumulative distribution functions of pre- and postsynaptic activity, respectively, Hebbian learning can be set to approximate Spearman's rank correlation (Genest & Favre, 2007), whereas setting both to the identity function makes the learning rule approximate simple covariance between pre- and postsynaptic activation. Various choices for f_{post} are discussed by Brito and Gerstner (2016), leading to the discovery of sparse codes, whereas the emergence of principal or independent components can be proven for the specific choices $f_{\mathrm{post}}(x) = x^2$ or $f_{\mathrm{post}}(x) = x^3$, respectively, in linear model neurons with constrained weights (Hyvärinen & Oja, 1998; Oja, 1982). It should be noted that these rules consider activations rather than membrane potentials, which is crucial here, since only the activation is subject to the effects of intrinsic adaptation.

For the simulation experiments presented here, both functions are set to simply subtract the mean activation $\bar{y} = \mathbb{E}[Y_t]$ enforced by intrinsic plasticity, such that uncorrelated firing at the mean rate leads to no synaptic growth. Transmission delays are neglected, resulting in the simple Hebbian learning rule

$$\mathrm{d}W_t^{(i,j)} = \delta(s(Y_t^{(j)} - \bar{y}) \cdot (Y_t^{(i)} - \bar{y}) - W_t^{(i,j)})\mathrm{d}t. \tag{2.8}$$

For strictly inhibitory synapses, weights are constrained to $W < 0$ and $s < 0$ is chosen, resulting in a form of anti-Hebbian learning (Földiák, 1990).

2.2 The Full Network Model. The dynamics of a neuron i within the adaptive network can be summarized in the most general form by equations 2.9 to 2.12, whereas a synaptic weight from neuron j to i adapts according to equation 2.13:

$$I_t^{(j)} \leftarrow \text{ external input or } Y_t^{(j)}, \tag{2.9}$$

$$\mathrm{d}X_t^{(i)} = a(X_t^{(i)})\mathrm{d}t + b(X_t^{(i)}) \sum_j W_t^{(i,j)} I_{t-\tau^{(i,j)}}^{(j)} \mathrm{d}t, \tag{2.10}$$

$$\mathrm{d}\bar{\phi}_t^{(i)} = \gamma(\eta(X_t^{(i)}) - \bar{\phi}_t^{(i)})\mathrm{d}t, \tag{2.11}$$

$$Y_t^{(i)} = \nu_{\bar{\phi}_t^{(i)}}(X_t^{(i)}) = (F_Y^{-1} \circ F_{X,\bar{\phi}_t^{(i)}})(X_t^{(i)}), \tag{2.12}$$

$$\mathrm{d}W_t^{(i,j)} = \delta(f_{\mathrm{pre}}(Y_{t-\tau^{(i,j)}}^{(j)}) \cdot f_{\mathrm{post}}(Y_t^{(i)}) - W_t^{(i,j)})\mathrm{d}t. \tag{2.13}$$

Here, superscripts denote neurons, I_t is an input signal (either external or the output of another neuron), X_t is the membrane potential, $\bar{\phi}_t$ is the empirical estimate of the membrane potential's sufficient statistics $\bar{\eta} = \mathbb{E}[\eta(X_t)]$, Y_t is the neuron's activation, W_t is the matrix of synaptic connection weights, and γ and δ are the learning rates of intrinsic and synaptic plasticity, respectively.

In the following, specific instances of this model are used, where the membrane potential is modeled by leaky integration; inputs are themselves assumed to be stationary, Markovian, mean-reverting stochastic processes with either gaussian, Laplacian, or beta distribution, the marginal output distribution is chosen to be log-gaussian, log-Laplacian, or uniform; synaptic delays are neglected; and simple Hebbian learning is employed. In the cases shown here, equations 2.10 and 2.13 take the form:

$$\mathrm{d}X_t{}^{(i)} = (-\theta X_t^{(i)} + \sqrt{2\theta} \sum_j W_t^{(i,j)} I_t^{(j)})\mathrm{d}t, \tag{2.14}$$

$$\mathrm{d}W_t^{(i,j)} = \delta(s(Y_t^{(j)} - \bar{y}) \cdot (Y_t^{(i)} - \bar{y}) - W_t^{(i,j)})\mathrm{d}t, \tag{2.15}$$

Table 1: Parameterization of Different Neuron Models.

$P_X \to P_Y$	Gaussian → log-Gaussian	Laplacian → log-Laplacian	Beta → Uniform	Laplacian → log-Gaussian
$\eta(x)$:	$\begin{pmatrix} x \\ x^2 \end{pmatrix}$	$\begin{pmatrix} x \\ \lvert x-\alpha_t \rvert \end{pmatrix}$	$\begin{pmatrix} x \\ x^2 \end{pmatrix}$	$\begin{pmatrix} x \\ \lvert x-\alpha_t \rvert \end{pmatrix}$
$\nu_{\bar{\phi}_t}(x)$:	$\exp\left(\frac{x-\alpha_t}{\beta_t}\right)$	$\exp\left(\frac{x-\alpha_t}{\beta_t}\right)$	$I_x(\alpha_t, \beta_t)$	$F_Y^{-1}(F_X(x; \alpha_t, \beta_t))$
where α_t:	$(\bar{\phi}_t)_0$	$(\bar{\phi}_t)_0$	$\frac{(\bar{\phi}_t)_0^2(1-(\bar{\phi}_t)_0)}{\sqrt{(\bar{\phi}_t)_1-(\bar{\phi}_t)_0^2}} - (\bar{\phi}_t)_0$	$(\bar{\phi}_t)_0$
and β_t:	$\sqrt{(\bar{\phi}_t)_1 - (\bar{\phi}_t)_0^2}$	$(\bar{\phi}_t)_1$	$\alpha((\bar{\phi}_t)_0^{-1} - 1)$	$(\bar{\phi}_t)_1$

where $\bar{y}$ is the expected value of the activation distribution, θ is the membrane potential time constant, and s is a scaling factor. The function η defining the sufficient statistics $\bar{\eta}$ and the activation functions ν resulting from the different choices of membrane potential and activation distributions used in this paper are listed in Table 1.

In the simulations presented here, the inputs to the model neurons are either outputs from other neurons, pixel intensities of image patches, or artificially generated colored noise. Noise stimuli with an exponential autocorrelation structure and stationary gaussian distribution are generated by Ornstein-Uhlenbeck processes in the form of equation 2.16, whereas a Laplacian stationary distribution is produced by stochastic processes as given by equation 2.17:

$$\mathrm{d}I_t = \gamma(\mu_t - I_t)\mathrm{d}t + \sigma_t\sqrt{2\gamma}\,\mathrm{d}\mathcal{B}_t \tag{2.16}$$

$$\mathrm{d}I_t{}^{(j)} = \gamma(\mu_t - I_t)\mathrm{d}t + \sigma_t\sqrt{\gamma(1 + \frac{\sqrt{2}|I_t|}{\sigma_t})}\mathrm{d}\mathcal{B}_t. \tag{2.17}$$

The functions μ_t and σ_t above are the (possibly time-varying) mean and standard deviation of the noise stimuli.

In some simulations, to ensure that multiple neurons within a population represent different aspects of their inputs, lateral inhibition is employed, such that one neuron inhibits all others, the next neuron inhibits all but the first, and so on. These inhibitory synaptic weights are trained via simple anti-Hebbian learning (Földiák, 1990) according to rule 2.15 with a negative scaling constant $s < 0$ and constrained to remain negative. This connectivity structure introduces a strict ordering of decorrelated neurons without imposing recurrent connectivity on the population.

3 Results

To illustrate the properties of the model described above, we present three simulation examples.

3.1 Intrinsic Plasticity in Isolation. First, consider a gaussian model neuron with a leaky integrating membrane potential as given in equation 2.14, driven by a single gaussian process as given by equation 2.16. During simulation, the mean μ_t and standard deviation σ_t of the gaussian input signal are changed at three points in time, resulting in four time intervals in each of which the membrane potential exhibits a different stationary gaussian distribution. We estimate the sufficient statistics of the respective membrane potential distributions through the two adaptation variables α_t and β_t (see Table 1, gaussian $\rightarrow$ log-gaussian), both of which change on a timescale much slower than the autocorrelation in the membrane potential process.

During simulation, the sufficient statistics, as well as the Kullback-Leibler divergence between the neuron's estimated and real membrane potential distribution, are tracked and aggregated over 100 trials. This equivalently measures the divergence between the desired, log-gaussian output distribution and the one actually achieved by the neuron (see supplementary text A.5 for a derivation).

The results are summarized in Figure 2. Evidently the model neurons reliably adapt to the various input statistics on a timescale determined by the dynamics of the adaptation variables, as well as the dynamics of the membrane potential. For an analytical derivation of the adaptation dynamics, we again refer you to the supplementary text A.4. The result illustrates that using the estimated membrane potential statistics in the parameterized activation function allows the neuron to compensate for sudden changes in the input distribution and maintain the neuron's desired output distribution.

3.2 Interaction between Intrinsic and Synaptic Plasticity. While the ability of individual neurons to adapt to changes in their environment as illustrated in the previous experiment is arguably a very useful feature by itself, our primary interest lies in the interaction of such plastic neurons with plastic synapses. We present two variations of an experiment inspired by Hyvärinen and Oja (1998) and Savin et al. (2010) to illustrate how the interaction of both plasticity mechanisms can drive individual neurons to become selective to a transformation-invariant representation of their multivariate input by discovering a principal or independent component. We are able to reproduce these results using only intrinsic plasticity and Hebbian synaptic plasticity as discussed above and need not rely on a third, distinct mechanism of synaptic scaling as suggested there. For both conditions, we provide a detailed fixed-point analysis of the synapse dynamics in the presence and absence of intrinsic plasticity and show theoretically how independent component analysis emerges (only) from the complementary contributions of intrinsic and synaptic plasticity. To this end, we use plasticity operating on two distinct timescales: a fast-acting form of intrinsic firing rate adaptation that renders the neurons invariant to changes in the scale of

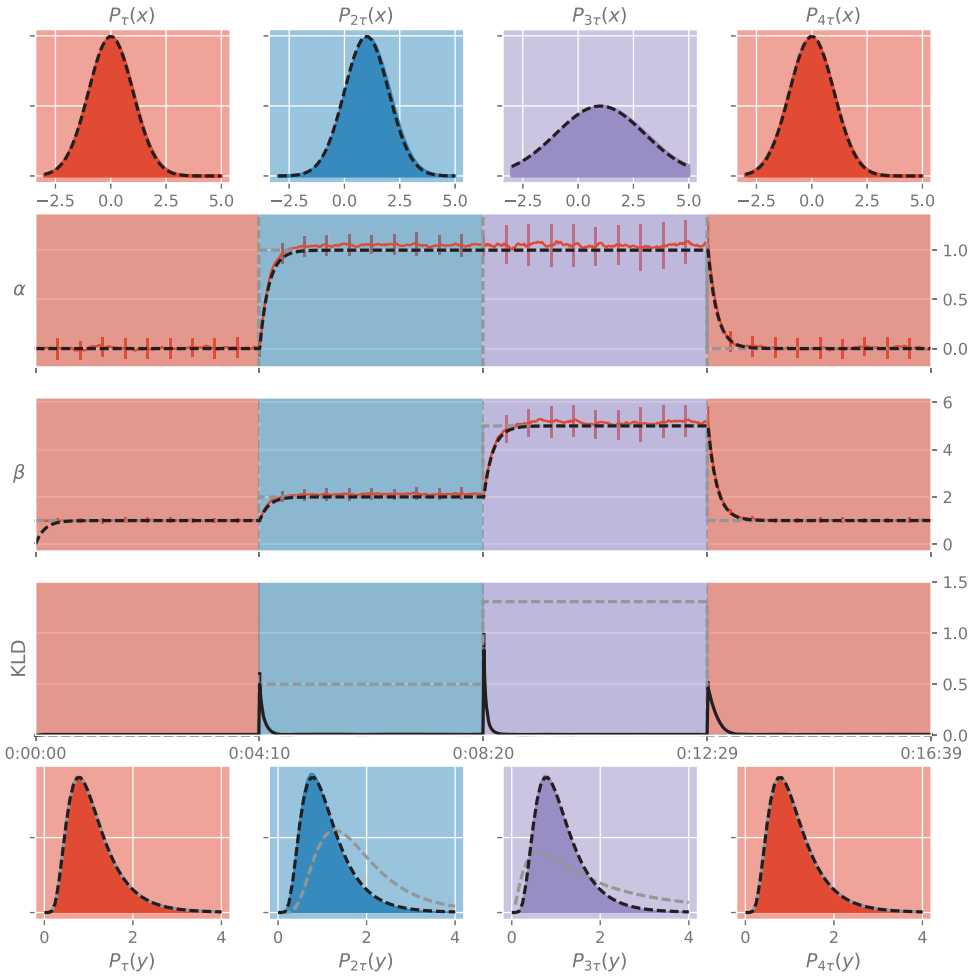


Figure 2: Adaptation of a model neuron to changing input distributions. Top: Stationary distributions of the gaussian membrane potential process (dashed lines) during four stages of the simulation (color-coded backgrounds) and the respective distributions estimated by the neuron's adaptation parameters at the end of each interval (filled histograms). Rows 2 and 3: Time course of the adaptation variables α and β, estimating the first and uncentered second moment of the membrane potential process, respectively. Across 100 trials, mean (solid red lines) and standard deviation (error bars) of the activation parameters are shown. The true moments (dashed gray lines), as well as the analytical values attainable via causal exponential filtering of sufficient statistics (dashed black lines), are provided for reference (see supplementary text A.4 for a derivation). Row 4: Trace of the Kullback-Leibler divergence between underlying and estimated membrane potential or, equivalently, desired and realized output distribution (solid black line). See supplementary text A.5 for an analytical derivation. For reference, the divergence for a hypothetical nonadaptive neuron with identical initial parameter values is given (dashed gray line). Bottom: Desired log-gaussian (dashed black line) and achieved (filled histograms) distributions of activation at the end of each time interval. For reference, the distributions of activation resulting from the same input distributions are given for a nonadaptive neuron with identical initial parameter values (dashed gray line).

their membrane potentials and much slower-acting Hebbian plasticity that adapts the weight vectors projecting into the neuron.

In both simulations, consider two independent source signals modeled by stationary stochastic processes. The source signals are mixed into two different linear combinations by multiplying the two-dimensional signal vector s_t with a mixing matrix M, here chosen to be the rotation matrix with an angle of 0.3π. The transformed signal vector $I_t = M \cdot s_t$ is then projected into two adaptive neurons through a matrix of adaptive synaptic weights W_t. The process driving the two adaptive neurons is thus given by $W_t \cdot I_t = W_t \cdot M \cdot s_t$. Recovering the two original source signals such that $W_t \cdot I_t = s_t$ requires finding the matrix $W_t \to M^{-1}$ without knowing M. This problem is accordingly referred to as blind source separation (Keziou, Fenniri, Messou, & Moreau, 2013). Since the sources here represent statistically independent signals, this can be realized through independent component analysis (Hyvärinen & Oja, 2000, short: ICA). In both simulations, the covariance matrix of both neurons' membrane potentials $\Sigma_t = W_t M (W_t M)^T$ is determined by the constant mixing matrix M and the time-varying weight matrix W_t. In particular, the variance $\sigma_t^{(i)^2} = (\Sigma_t)_{i,i} = w_t^{(i)} M (w_t^{(i)} M)^T$ of each neuron i's membrane potential is given as a function of the corresponding row $w_t^{(i)}$ of W_t, that is, the vector of weights projecting into neuron i. At each point in time, the effect of the fast-acting intrinsic plasticity, which renders the neurons invariant to changes in their marginal membrane potential variances, can thus be modeled as a normalization of each membrane potential: $Y_t^{(i)} \approx \nu(\frac{X_t^{(i)}}{\sigma_t^{(i)}})$. This effect of intrinsic plasticity is equivalent to synaptic scaling (Turrigiano, 2008), albeit implemented within the neuron without explicit knowledge of the weight vector, rather than changing the weight vector itself. For each weight vector $w_t^{(i)}$, this makes it possible to theoretically derive the vector field of expected weight changes $\mathbb{E}[\frac{\mathrm{d}w_t^{(i)}}{\mathrm{d}t}]$ from the Hebbian weight update equation 2.7, identify fixed points, and prove convergence. A corresponding vector field in the absence of intrinsic plasticity can be derived in strict analogy, only replacing the normalization of the membrane potential by a scaling with a constant independent of the weight vector: $Y_t^{(i)} \approx \nu(c \cdot X_t^{(i)})$. An analytical derivation of attractor landscapes can be found in the supplementary text C.2.

We show that the basic Hebbian plasticity rule, which updates synaptic weights according to equation 2.15 based on the pre- and postsynaptic activities I_t and Y_t, respectively, can achieve blind source separation if and only if paired with intrinsic plasticity. By recovering the original independent sources, the neurons become invariant to the mixing effect of an (orthogonal) transformation M, illustrating a network-level form of invariance due to plasticity. Thus, multiple neurons can develop a nonredundant, information theoretically efficient representation of their multidimensional inputs (Isomura et al., 2015).

3.2.1 PCA in Individual Neurons. For the first experiment, assume the sources to be gaussian with different variances. The model neurons are chosen to map gaussian membrane potentials to log-gaussian activations. Over time, we trace the evolution of the synaptic weights under the influence of synaptic and intrinsic plasticity and observe that both weight vectors rapidly align themselves with the largest eigenvector of the covariance matrix of the joint membrane potential distribution. As a consequence, both neurons become tuned to the first principal component of the two-dimensional input, which corresponds to the one recovered original source with largest variance. The magnitude of the resulting weight vectors is equal for both neurons. The top-left panel of Figure 3 summarizes these results, and the two panels on the top right confirm that due to plasticity, the empirical distributions of the membrane potential and activation approach the desired distributions at the end of the simulation.

To understand these results, the respective roles of intrinsic and synaptic plasticity must be analyzed. Hebbian synaptic plasticity as employed here drives the synaptic connections to maximize postsynaptic activations. Due to the nonlinear relationship, here exponential, between membrane potential and activation, more dispersed stimuli are more effective at driving the neuron to fire at high rates, since large, positive deviations of the membrane potential are amplified, whereas similar negative deviations are attenuated. (For a precise derivation of how changes in higher-order moments of the membrane potential distribution affect the neuron's mean firing rate due to its nonlinear activation function, see the supplementary text C.1.) Hebbian plasticity thus rotates the weight vectors toward those directions in input space where dispersion is maximized. Here, for the example of gaussian signals, where all moments beyond the second vanish and due to the activation function's strong sensitivity to the second moment, they correspond to the directions of the principal components, ordered by the associated eigenvalues. In the absence of intrinsic plasticity, synaptic scaling, or other stabilizing mechanisms such as the BCM rule (Bienenstock et al., 1982), such a rule invariably leads to instability. Here, however, fast-acting intrinsic plasticity stabilizes Hebbian plasticity by keeping the post-synaptic neuron's output distribution constant, independent of the magnitude of the current weight vector, thus constraining synaptic growth. The choice of the desired activation distribution of the neuron thus indirectly determines the stable length of the weight vector resulting from Hebbian plasticity, as well as the speed and reliability of convergence to the principal components.

The bottom left panel of Figure 3 shows the attractor landscape of the Hebbian learning procedure, which exhibits two stable solutions, each corresponding to a weight vector aligned with the first principal component and an unstable fixed point at 0, pushing weight vectors to non trivial solutions. As indicated by the domains of attraction, a single neuron with Hebbian synapses and intrinsic plasticity converges to either of two

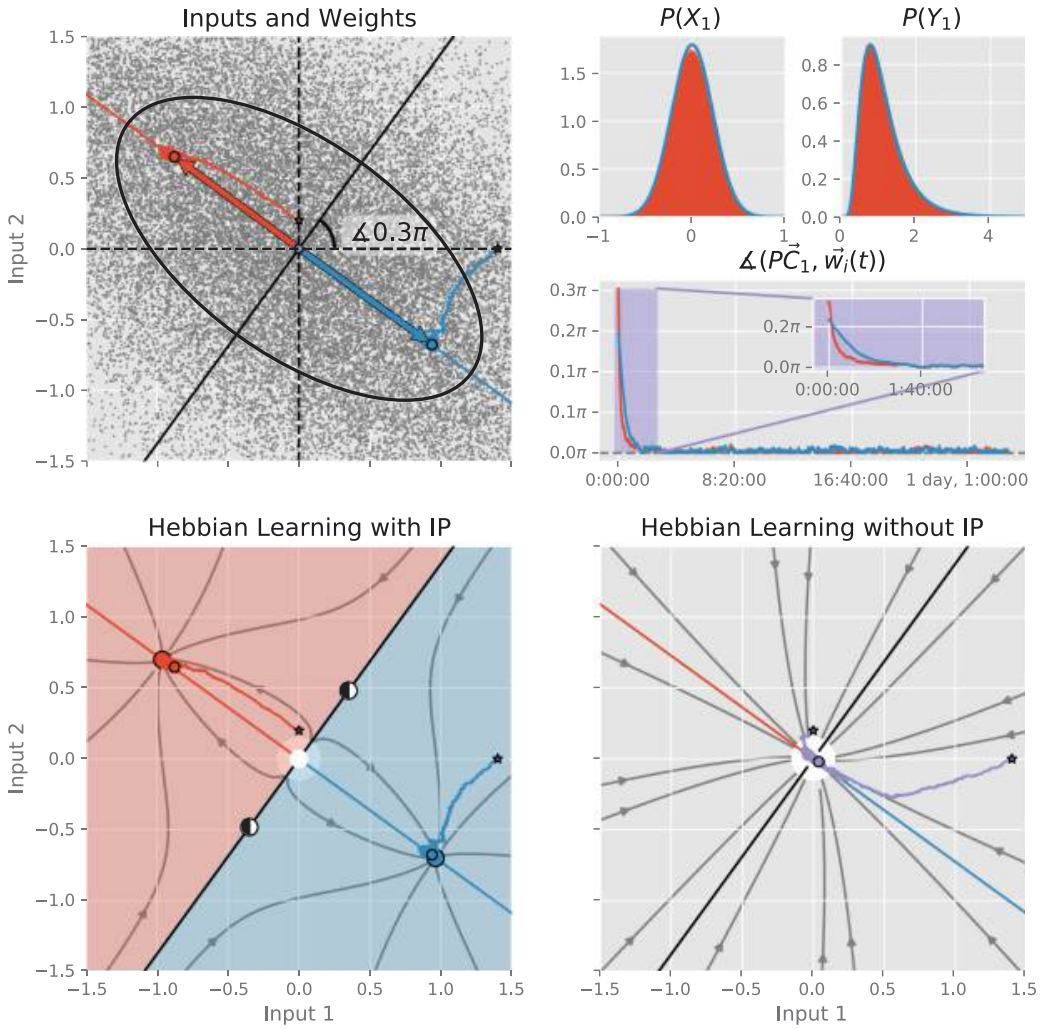


Figure 3: Principal component analysis realized by two plastic neurons with Hebbian synapses. The top-left panel shows, color-coded for each neuron, the weight vectors evolving over time, from their respective starting values indicated by a star to final values indicated by a circle. Dashed lines mark the standard basis, and solid straight lines represent eigen directions of the input covariance matrix. The jointly gaussian input distribution is represented by random samples (gray dots) and an iso-probability ellipse. The two panels on the top right show empirical and analytically expected or desired probability densities of the first neuron's membrane potential X_t and its activation Y_t. The right-hand panel presents the absolute inner angle in radians between each weight vector (color-coded) and the first principal component direction as a function of time. The bottom-left panel presents the adaptation of the weight vectors in the phase space of the Hebbian learning rule (see equation 2.15) under the influence of IP for the given input distribution. The weight vectors converge to either of the two stable fixed points (filled circles), depending on the domain of attraction (coded by background color), and diverge from the trivial solution at (0, 0). On the separatrix, two saddle points emerge (half-filled circles). The bottom-right panel, for comparison, shows the phase space and example weight trajectories resulting from synaptic plasticity alone in the absence of intrinsic plasticity.

representations of the first principal component of the input, recovering the source signal with larger variance. The temporal dynamics and stability of the adaptation procedure are illustrated in the second row panel on the right side of Figure 3, where the angles between each weight vector and the direction of the first principal component (PC_1) are plotted as a function of time. After quickly rotating toward PC_1, the weight vectors remain stable at an angle close to 0, demonstrating the long-term stability of the solution. To illustrate the crucial role of intrinsic plasticity for stabilizing Hebbian learning, the bottom right panel of the same figure shows the phase space and weight vector trajectories from identical starting positions in the absence of intrinsic plasticity. For the parameters chosen here, both initial conditions now lie within the domain of attraction of a stable fixed point at the trivial solution $(0, 0)$.

3.2.2 ICA in Individual Neurons. The perspective gained from the previous example can be transferred to a more interesting experiment presented by Savin et al. (2010). Consider now that the two sources are identical, independent nongaussian (here Laplacian) processes, which are again mixed by the same matrix M and projected onto adaptive neurons through a matrix of adaptive synaptic weights W_t. Applying the same reasoning as in the previous example, Hebbian learning paired with intrinsic plasticity leads the neuron to discover those projections of the input signal that result in the most dispersed membrane potentials. However, due to the fact that now their covariance matrix itself is orthogonal, any normed linear combination of the inputs has the same variance, and thus no particular unique principal components can be defined. As a consequence, the original source signals cannot be recovered using any method such as PCA that takes only the second moment, that is, the covariance matrix, into account. However, since the input signals are now no longer gaussian, higher-order moments beyond the variance, such as kurtosis, can be taken into account to define unique directions of maximum dispersion. In the example presented here, the source signals' marginal Laplace distributions are leptokurtic, showing a higher kurtosis than any normalized linear combination of the two. The independent components (ICs), maximizing kurtosis, thus correspond to the demixed original source signals.

Since the nonlinear activation function is also sensitive to higher moments beyond the variance (see the supplementary text C.1), the neural adaptation procedure in this case selects the directions that maximize kurtosis and thus discovers the most prominent independent component, which corresponds to one demixed source signal. (See Figure 4 for a summary of the results.) To demonstrate that both neurons, due to different initial weight vectors, truly discover independent components, the top-right panel of Figure 4 shows the copula function of both neurons at the beginning and end of the simulation, converging to a uniform distribution that indicates statistical independence (rather than mere uncorrelatedness).

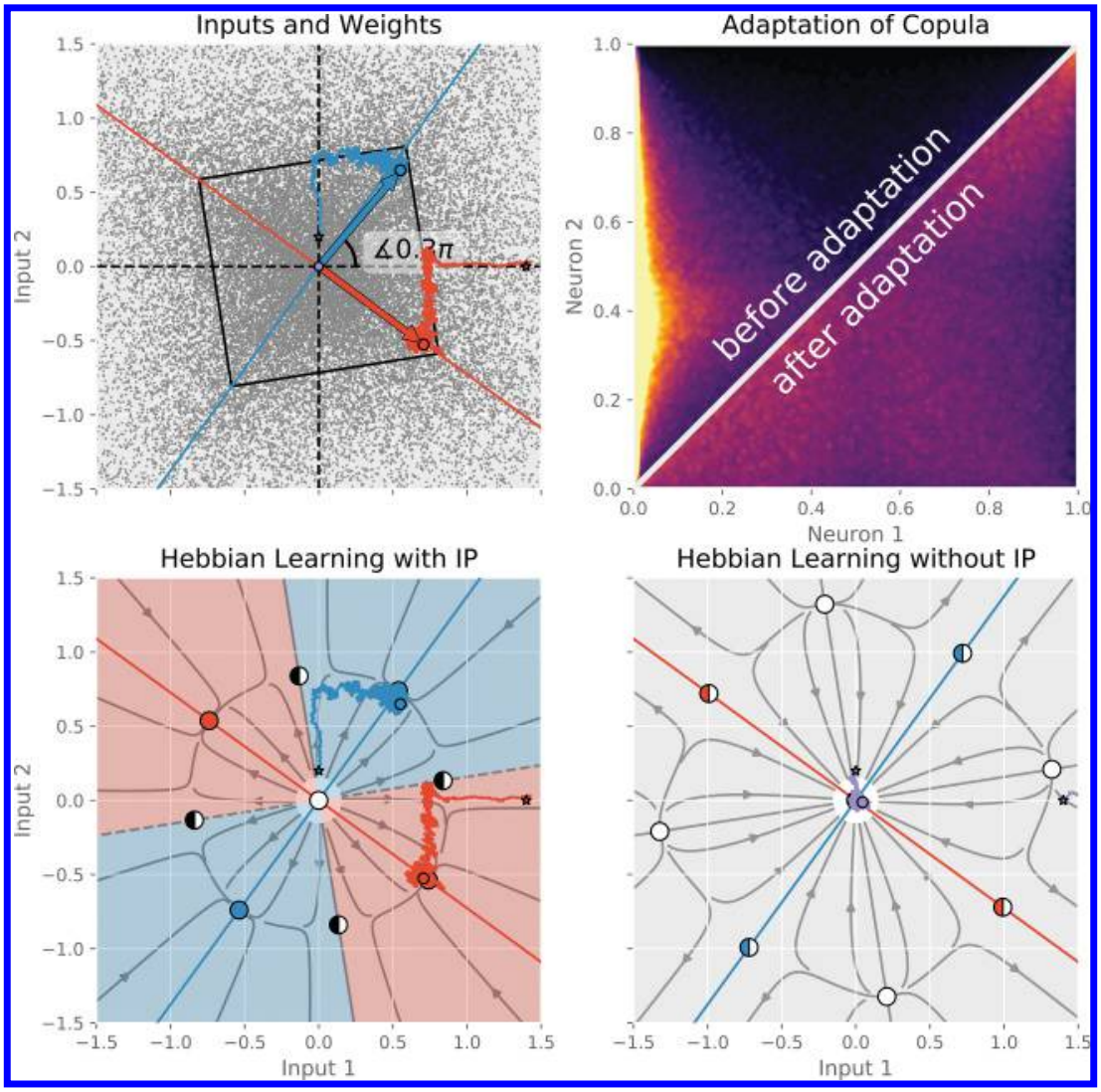


Figure 4: Independent component analysis realized by two plastic neurons with Hebbian synapses. The top-left panel shows, color-coded for both independent components of the input, the weight vectors evolving over time from their respective starting values indicated by a star to final values indicated by a circle. Dashed lines mark the standard basis, and solid straight lines mark the basis rotated by 0.3π. The joint input distribution is represented by random samples (gray dots) and an iso-probability diamond. The two plots on the top right show the copula between both neurons at the beginning and end of learning. The bottom-left panel presents the adaptation of the weight vectors in the phase space of the Hebbian learning rule (see equation 2.15) under the influence of IP for the given input distribution. The weight vectors converge to any of the four stable fixed points (filled circles), depending on the domain of attraction (coded by background color), and diverge from the trivial solution at (0, 0). On the separatrices, four saddle points emerge (half-filled circles). The bottom-right panel, for comparison, shows the phase space and example weight trajectories resulting from synaptic plasticity alone in the absence of intrinsic plasticity.

3.2.3 PCA and ICA in Populations. In the two experiments above, we consider the extreme cases where either no unique independent components can be defined, because the kurtosis of the gaussian inputs is 0, or no unique principal components can be defined, as the covariance matrix of the Laplacian inputs is the identity. In each case, a neuron then recovers the appropriately defined dominant component, a principal component or an independent component, respectively. Since the nonlinear activation functions chosen here depend on both second- and third-order moments (and more), this raises the question of whether, in the presence of both principal and independent components, a model neuron would select the former, the latter, or neither, instead converging to a compromise between the two. For an exponential activation function, the optimal solution maximizes a weighted combination of all moments of the neuron's membrane potential, with lower moments being the most influential (see supplementary text C.1). While dominant principal components are thus strongly favored by the adaptation procedure, independent components or yet other directions could coexist as stable solutions.

In order for a population of adaptive neurons to implement either PCA, ICA, or any other transformation-invariant representation of the population's input signals, it is crucial that individual neurons reliably become selective to different components. In the presence of strongly dominant components, as shown in the first example, different initial conditions alone are insufficient to ensure that different principal or independent components are discovered.[1] To mitigate this problem, lateral strictly inhibitory synapses with scale parameter $s = -5$ are placed between the adaptive neurons and updated via anti-Hebbian learning as outlined in section 2.2.

For two Laplacian source signals as in the previous example, we vary the mixing coefficients to realize three different scenarios in which the directions of maximum dispersion change from favoring principal components to favoring independent components. In all three cases, we train adaptive neurons mapping the Laplacian inputs to log-gaussian outputs and trace the weight vectors under the effect of Hebbian and intrinsic plasticity. (See Figure 5 for a summary of the results.) In accordance with expectations, the lateral decorrelation forces neurons to discover different components of the inputs in a predictable manner. While one adaptation variable emulates synaptic scaling by controlling the membrane potential's variance, the other compensates for the imbalance between feedforward activation and the combined effect of feedforward and lateral inhibition by controlling the mean. The identities of the extracted components by each neuron

[1] If it were the case that an individual neuron, depending on its initial weight vector, would select a nondominant component, this would be considered a flaw of the algorithm, as in that case, it could not be ensured that the most relevant components of the input are discovered.

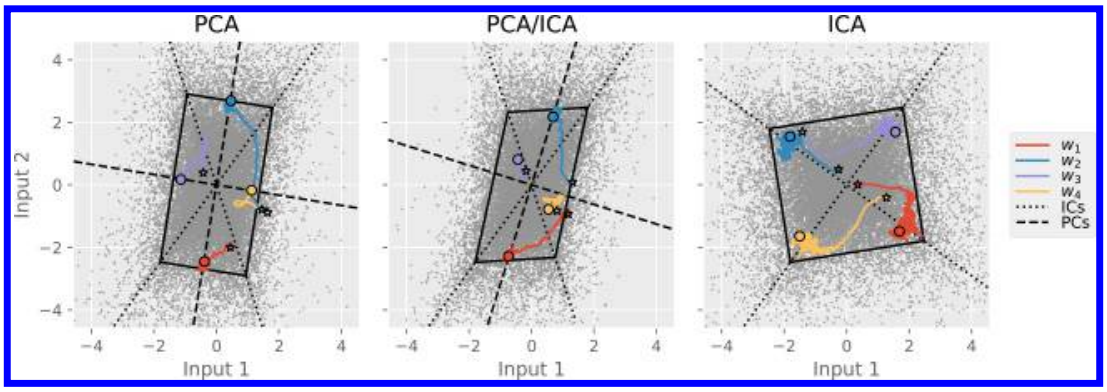


Figure 5: PCA and ICA in a population of four laterally decorrelated adaptive neurons with Hebbian synapses. Gray dots represent samples of the bivariate stochastic input process, solid black lines show iso-probability contours, dashed black lines show principal component directions, and dotted black lines represent independent component directions. Colored lines show trajectories of weight vectors learned by the neurons, with a star marking the initial and a circle marking the final value. On the left, all four neurons converge to both principal components with alternating signs. In the middle, two neurons converge to the dominant principal component, whereas the other two converge to the dominant independent component, all with alternating signs. On the right, all four neurons converge to both independent components with opposing signs.

may change under nonorthogonal transformations, but the resulting representation of the input learned by the population is invariant with respect to further orthogonal transformations of the input.

3.3 Learning Representations of Image Patches. Finally, to demonstrate the generality of the results discussed thus far on high-dimensional data, we present a network of the same structure as above, evaluated on two different data sets of image patches of size 28×28 pixels. For each pixel, an adaptive "sensory neuron" is trained to map its input, the corresponding pixel's intensity (plus a small noise term) scaled to the range from 0 to 1, which we assumed to be distributed according to a beta distribution, onto a uniformly distributed activation in the range from -0.5 to 0.5. These marginally uniform activations are subsequently used as stimuli and projected through Hebbian synapses onto five adaptive neurons that are laterally decorrelated via anti-Hebbian synapses in the same manner as in the previous example with a relative weight scale of $s = -3$. The neurons are chosen to map gaussian inputs to log-gaussian activations and thus exhibit the same exponential activation function as discussed previously. Each image patch is presented for a simulated time of 50 ms, and the total simulation time is 150,000 s at discretization steps of 1 ms each. Time constants of the neurons' membrane potentials are set to 5 ms, and adaptation rates for

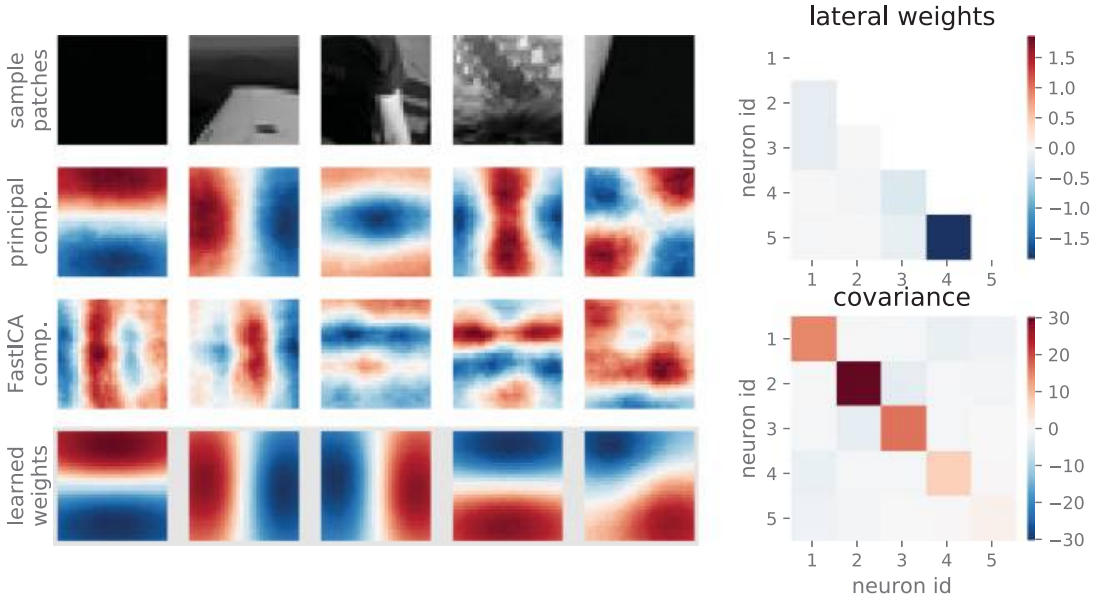


Figure 6: On the left, from top to bottom, we show five random sample patches, the first five principal components of the stimuli, the first five independent components extracted by FastICA, and the weights learned by the five neurons of the population. On the top right, the final weight matrix of lateral inhibitory synapses is shown. The bottom right contains the resulting correlation matrix between the five neurons' activations. Evidently the population discovers mostly uncorrelated principal components. (Image patches courtesy of Winder & Impressive Machines LLC, N.d.)

intrinsic, anti-Hebbian, and Hebbian plasticity are set to 10 s, 100 s, and 200 s, respectively.

First, a freely available collection of 500,000 natural image patches (Winder & Impressive Machines LLC, N.d.) is used, where the model neurons are shown to become selective to principal components, rather than independent components as extracted using an implementation of the FastICA algorithm (Hyvärinen & Oja, 2000; Pedregosa et al., 2011). For a summary of the results, see Figure 6. Due to the unstructured nature and translation-invariant statistics of the natural image patches, this result is in line with our expectations. Since the inputs to the adaptive neurons violate the assumption of gaussianity, intrinsic plasticity fails to achieve identical variance in the activation of all neurons but still succeeds in stabilizing Hebbian plasticity. The discrepancy between the assumed gaussian membrane potential distribution and the sparser observed distributions resulting from the neurons' discovery of nongaussian components could be alleviated by using a more general membrane potential distribution family such as the generalized normal distribution, which contains both gaussian and Laplacian distributions as special cases. Here, however, we restrict ourselves to the much simpler assumption of gaussian membrane potentials, for which closed-form expressions of the sufficient statistics exist.

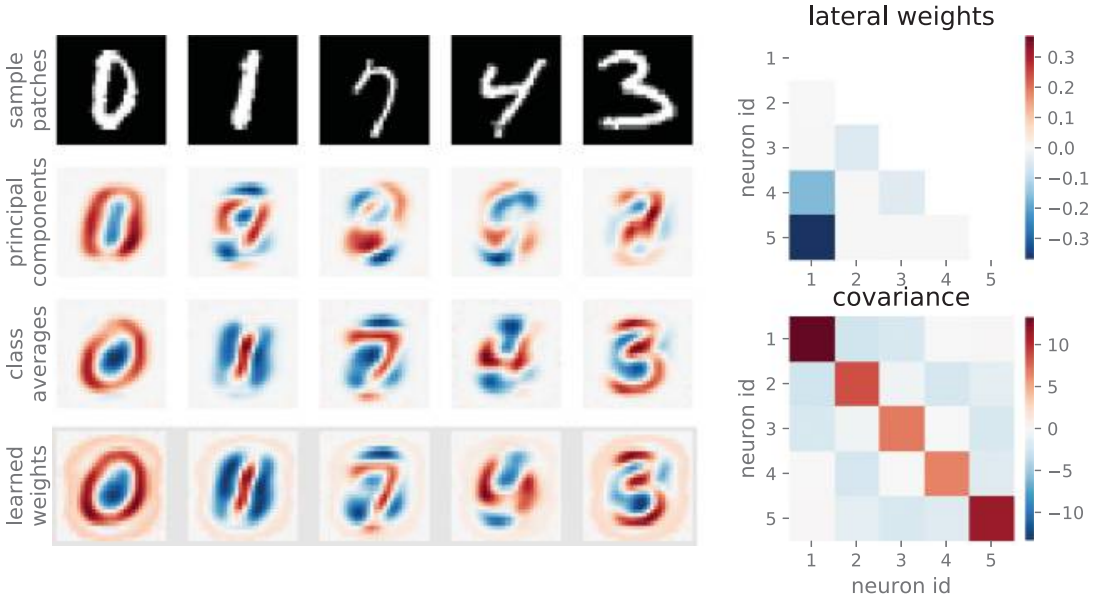


Figure 7: On the left, from top to bottom, we show five sample images, five principal components of the stimuli, the average stimuli for each of five classes, and the weights learned by the five neurons of the population. On the top right, the final weight matrix of lateral inhibitory synapses is shown. The bottom right contains the resulting correlation matrix between the five neurons' activations. Evidently the population discovers mostly uncorrelated representations of the classes. (MNIST patches courtesy of LeCun, Cortes, & Burges, N.d.)

Second, an identical network with the same parameters as in the previous case is used with 60,000 image patches of the same size from the MNIST database of handwritten digits (LeCun, Cortes, & Burges, N.d.). (For a summary of the results, see Figure 7). Due to the structuredness of the sample images, components representing the class averages emerge as the learned weights rather than principal components. In an unsupervised fashion, the population thus learns a sparse representation of its inputs. This result is in line with a prediction made by Triesch (2007), who proposes that in a high-dimensional learning problem with clustered data, the relative contribution of each cluster on a neuron's activation can be approximated by the expected activation in a corresponding percentile of the activation distribution.[2] In this case, a neuron optimally tuned to respond strongly to a single cluster out of the 10 available can be heuristically expected to exhibit the 10% of its highest firing rates in response to inputs drawn from its favored cluster, resulting in an expected activation of $y_{\text{fav}} = \mathbb{E}_{P_Y}[Y | F_Y^{-1}(0.9) < Y]$ in response to that class. Assuming that intrinsic plasticity successfully enforces the desired activation distribution P_Y with mean $\bar{y} = \mathbb{E}_{P_Y}[Y]$, the ratio $\frac{y_{\text{fav}}}{\bar{y}}$ could be viewed as a measure of selectivity toward a dominant class,

[2] Assuming that cluster centers are linearly independent and equally frequent.

which can be predetermined by an appropriate choice of activation distribution. As the neuron's mean activation drives Hebbian learning of its input synapses, this selectivity in the neuron's response to specific input classes is proportionally reflected in its synaptic weights. For a log-gaussian distribution with parameters $\mu = 0$ and $\sigma = 1$ as used here, a well-tuned neuron could thus be expected to show an overall mean activation of $\bar{y} \approx 1.65$ with a mean activation of $y_{\text{fav}} \approx 6.42$ in response to its favored class input. For comparison, a neuron with enforced exponential activation distribution of equal mean could be expected to respond with a mean activation of $y_{\text{fav}} \approx 5.45$ to its favored cluster and would thus be slightly less selective for a unique class. For the first neuron in the numerical simulations above, the empirical results at $\bar{y} \approx 1.86$ and $y_{\text{fav}} \approx 9.82$ even exceed the heuristic predictions, implying that the neuron is indeed highly selective to a single cluster mean, in this case corresponding to the digit 0. The discrepancy between theoretical prediction and numerical results may be attributed to several compounding effects, such as temporal dynamics of the stochastic processes used here, high intraclass variability, or the correlatedness of the cluster centers of the MNIST digits, which make a direct quantitative comparison between numerical results and theoretical predictions difficult.

In both cases discussed, convergence to the final solution is quick, with components converging one by one in the sequence determined by the lateral inhibition structure, and the discovered representation remains stable over the remainder of the simulation. As the learned components reflect the structure present in the stimuli, the population's representation of its input is thus invariant to (orthogonal) transformations in the 784-dimensional input space, as any such transformation of the input space is counteracted by an according adjustment of synaptic weights and intrinsic excitability.

4 Discussion

Our numerical results confirm the theoretical conclusion that a population of adaptive neurons, implementing only local mechanisms of intrinsic and synaptic plasticity, can realize homeostatic self-regulation that renders it invariant with respect to affine-linear/orthogonal transformations of its multivariate input. This is here achieved by the recovery of independent or principal components, but could be similarly realized by finding any other arbitrary but uniquely defined representation of the population's input. For clustered inputs, the activation distribution could be chosen in accordance with a heuristic outlined by Triesch (2007), such as to achieve a certain specificity in the neurons' response to specific clusters and thus enforce a sparse code.

The generality of the model allows the implementation and combination of various other forms of invariance, even where gradient-based methods might become prohibitively complex. For example, the stochastic membrane potential process with its parametric stationary distribution can be

chosen freely, so that its sufficient statistics resemble the properties we wish the neuron to become invariant to, such as higher-order moments.

Nonlinear forms of Hebbian plasticity as discussed by Brito and Gerstner (2016) could be used to learn sparse input representations other than independent or principal components. Additionally, a concave nonlinear dependency on the postsynaptic activation could act to further stabilize Hebbian learning by slowing synaptic growth at high firing rates, thus allowing for much slower intrinsic adaptation to compensate.

By including interaction delays in the synapse model, synaptic plasticity could be used to influence not just the instantaneous correlation structure between neurons but the full auto- and cross-correlation structure in time. By choosing different time constants for different adaptation variables, multiple timescales of adaptation can be modeled, a phenomenon observed in vivo (Fairhall et al., 2000; Turrigiano, 2008). Here, only a combination of slow Hebbian plasticity with fast intrinsic plasticity, as could be realized by rapidly acting homeostatic mechanisms such as spike rate adaptation or the accumulation or depletion of neurotransmitters, is discussed. However, biological evidence suggests that in particular, slow intrinsic adaptation (Turrigiano, 2008) and fast-changing weights (Zucker & Regehr, 2002) should be studied. Instead of the artificially constrained topology of lateral connectivity, which is used here to demonstrate the recovery of (only) dominant components, random sparse connectivity could be used in a sufficiently large network to find a similar, overcomplete, and invariant representation of the population's input.

Despite the generality of our results, we make several assumptions that should be asserted. First, intrinsic plasticity here operates on membrane potentials only. While rate-based or spike-triggered adaptation could be included, we abstain from doing this due to the complexity of the resulting model. Second, the neuron model is rate based; spiking effects could be included only via point or renewal processes with time-varying rate functions. This constrains the scope of spike timing effects and thus spike timing–dependent plasticity mechanisms that can be incorporated into the model. Third, intrinsic adaptation effects are not modeled as part of the membrane potential itself but rather as separate adaptation variables, similar to, for example, the adaptive exponential model (Gerstner & Brette, 2009), and can thus be observed only by their effect on the neurons' output behavior. Fourth, for the derivation of adaptation dynamics, the sufficient statistics that the neurons become invariant to are assumed to be twice differentiable for mathematical convenience, despite this not being a necessary condition. Fifth, the restriction of membrane potentials to (locally) stationary diffusion processes, albeit still very general, may be overly restrictive for some applications where synaptic inputs exhibit more structure, such as neurons driven by a low number of incoming spikes. In such cases, a different class of processes should be used. Finally, the model is stated here in the limiting case, where changes to the parameters of the neurons' inputs occur so slowly or rarely that the resulting membrane potentials can

be assumed to be locally or piece-wise stationary. When adaptation variables and input statistics evolve on similar timescales, this assumption is violated; the separation between input processes and their (slowly changing) parameters breaks down, and some of the arguments presented here can no longer be directly applied. In a machine learning context, this problem is commonly referred to as concept drift (Tsymbal, 2004), and adaptive networks might be well capable of coping with it. In this regime of very fast adaptation, intrinsic plasticity might no longer stabilize neural activations but could instead endow neurons with a local estimate of membrane potential statistics, potentially increasing their dynamic range (Fairhall et al., 2000; Fairhall, Lewen, Bialek, & de Ruyter Van Steveninck, 2001). This remains to be studied in future work.

In ongoing work, we explore whether a simple choice of intrinsic plasticity, resulting in scale invariance, can be used to stabilize activity in sensory neuron populations, deep feedforward and recurrently connected networks, thus improving information transmission as suggested by Turrigiano and Nelson (2004). This effect could potentially complement or replace specifically crafted inherently stable (deep) feedforward networks (see Klambauer, Unterthiner, Mayr, & Hochreiter, 2017, for example) by addressing the problem of vanishing gradients, a pressing issue in the field of deep learning.

Conversely, despite the hypothesized stabilizing effect of intrinsic plasticity in feedforward networks, pathological recurrent connectivity could potentially lead intrinsic plasticity to actively destabilize a network. Such pathological network behavior needs to be studied to discover limitations and possible undesired sideeffects of plasticity and could potentially provide some theoretical insight into neurological conditions such as epilepsy.

Finally, in order to assert the biological compatibility of the abstract mathematical model presented here, the biophysical mechanism underlying plasticity needs to be tied to adaptation variables, and a reasonable class of membrane potential processes and instantaneous firing rate distributions needs to be determined from biological evidence to guide modeling choices.

Appendix A: Mapping Stochastic Processes

A.1 Preliminaries.

Lemma 1 (Itô's lemma, theorem 4.2.1. of Øksendal, 2003). *For a function $u \in C^{1,2}([0,\infty] \times \mathbb{R}^m, \mathbb{R}^k)$ and an m-dimensional stochastic Itô process X_t, the transformation $Y_t = u(t, X_t)$ is a k-dimensional Itô process that can be expressed as*

$$du_k(t, X_t) = \frac{\delta u_k}{\delta t}(t, X_t)dt + \sum_i \frac{\delta u_k}{\delta x^i}(t, X_t)dX_t^i + \frac{1}{2}\sum_{i,j} \frac{\delta^2 u_k}{\delta x^i \delta x^j}(t, X_t)dX_t^i dX_t^j.$$

For linear transformations u, we see that stochastic differentials preserve linearity. A special case of this lemma is:

Corollary 1. *For a one-dimensional process X_t and a time-invariant function u, Itô's lemma yields*

$$du(X_t) = u'(X_t)dX_t + \frac{1}{2}u''(X_t)(dX_t)^2 \tag{A.1}$$

$$= \left(u'(X_t)a(t, X_t) + \frac{1}{2}u''(X_t)b(t, X_t)^2\right)dt + u'(X_t)b(t, X_t)d\mathcal{B}_t, \tag{A.2}$$

which is an Itô process as well.

Lemma 2 (stationary distribution, theorem 4.68 of Capasso & Bakstein, 2015, and section 3.2 of Stepanov, 2013). *Let X_t be a stationary diffusion process of the form*

$$dX_t = a(X_t)dt + b(X_t)d\mathcal{B}_t$$

with sufficiently regular functions a and $b > 0$.[3] Then the density of the stationary distribution is (up to a scaling factor) given by

$$f_X^{\infty}(x) \propto \frac{1}{b(x)^2}\, exp\left(\int_0^x \frac{2a(y)}{b(y)^2}dy\right).$$

Using corollary 1 and lemma 2, the properties of a specific process, the Ornstein-Uhlenbeck process, can be derived.

Lemma 3 (Ornstein-Uhlenbeck process). *Let X_t be a one-dimensional Itô process of the form*

$$dX_t = \theta(\mu_t - X_t)dt + \sigma_t\sqrt{2\theta}d\mathcal{B}_t,$$

where $\theta > 0, \sigma > 0$. We call X_t an Ornstein-Uhlenbeck process (Stepanov, 2013, section 2.6). For an initial condition X_0, its solution is given by

$$X_t = exp\,(-\theta t)X_0 + \int_0^t \theta\, exp\,(\theta(s-t))\mu_s ds$$

$$+ \sqrt{2\theta}\int_0^t exp\,(\theta(s-t))\sigma_s d\mathcal{B}_s.$$

[3] The conditions are quite general yet very technical and can be found in Capasso and Bakstein (2015, theorem 4.56).

For constant μ_t, σ_t, it is a stationary process with stationary gaussian distribution with mean μ_t and autocovariance $\mathbb{E}[X_{t_1} \cdot X_{t_2}] = \sigma_t^2 \, exp\,(-\theta|t_1 - t_2|)$.

Proof. Let X_t be as defined. We apply Itô's lemma to define $\tilde{X}_t = u(t, X_t)$ with $u(t, x) = \exp(\theta t)x$ and simplify:

$$\begin{aligned} \mathrm{d}\tilde{X}_t &= \theta \exp(\theta t) X_t \mathrm{d}t + \exp(\theta t)\mathrm{d}X_t \\ &= \theta \exp(\theta t)\mu_t \mathrm{d}t + \exp(\theta t)\sigma_t \sqrt{2\theta}\mathrm{d}\mathcal{B}_t. \end{aligned}$$

Rewriting the stochastic differential equation above as stochastic integrals gives

$$\begin{aligned} \tilde{X}_t &= X_0 + \int_0^t \theta \exp(\theta s)\mu_s \mathrm{d}s + \int_0^t \exp(\theta s)\sigma_s \sqrt{2\theta}\mathrm{d}\mathcal{B}_s \\ \Rightarrow X_t &= \exp(-\theta t)X_0 + \int_0^t \theta \exp(\theta(s-t))\mu_s \mathrm{d}s \\ &\quad + \sqrt{2\theta}\int_0^t \exp(\theta(s-t))\sigma_s \mathrm{d}\mathcal{B}_s. \end{aligned}$$

For a proof of stationarity and the full autocorrelation function, see, for example, Stepanov (2013, section 2.9).

Using lemma 2, it follows that the stationary distribution of X_t has the gaussian density

$$f_X^\infty(x) \propto \frac{1}{b(x)^2} \exp\left(\int_0^x \frac{2a(y)}{b(y)^2}\mathrm{d}y\right) \tag{A.3}$$

$$= \frac{1}{2\theta\sigma_t^2} \exp\left(-\frac{(x-\mu_t)^2}{2\sigma^2}\right), \tag{A.4}$$

$$\propto \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu_t)^2}{2\sigma^2}\right) \tag{A.5}$$

$$= \mathcal{N}(\mu_t, \sigma_t^2). \tag{A.6}$$

□

When the above process X_t is driven not by white noise but an integrable time-varying function I_t (i.e., the solution of another stochastic process), a solution of X_t can be equivalently derived where integration is not performed with respect to the differential $\mathrm{d}\mathcal{B}_s$ but instead with respect to $I_s\mathrm{d}s$.

A.2 Deriving a Nonlinear Activation Function.

Lemma 4. *Let X_t denote a stationary Itô diffusion process and let f_X^∞ denote the density of the corresponding stable distribution P_X^∞. Choose a monotonically*

increasing function $v \in C^2(\mathbb{R})$. Then $Y_t = v(X_t)$ is again a stationary Itô process with a stable distribution P_Y^∞ that has the density

$$f_Y^\infty(y) \propto (v^{-1})'(y) f_X^\infty(v^{-1}(y)),$$

where we use $(v^{-1})'(y) = \frac{dv^{-1}(y)}{dy}$ to denote the derivative of v^{-1} with respect to y.

Proof. Given v as above and a diffusion process of the form

$$\mathrm{d}X_t = a(X_t)\mathrm{d}t + b(X_t)\mathrm{d}\mathcal{B}_t,$$

Itô's lemma implies for $Y_t = v(X_t)$ that

$$\mathrm{d}Y_t = \left(v'(X_t)a(X_t) + \frac{1}{2}v''(X_t)b(X_t)^2\right)\mathrm{d}t + v'(X_t)b(t, X_t)\mathrm{d}\mathcal{B}_t \tag{A.7}$$

$$= \tilde{a}(Y_t)\mathrm{d}t + \tilde{b}(Y_t)\mathrm{d}\mathcal{B}_t, \quad \text{where} \tag{A.8}$$

$$\tilde{a}(y) = v'(v^{-1}(y))a(v^{-1}(y)) + \frac{1}{2}v''(v^{-1}(y))b(v^{-1}(y))^2 \tag{A.9}$$

$$= \frac{a(v^{-1}(y))}{(v^{-1})'(y)} - \frac{(v^{-1})''(y)}{2((v^{-1})'(y))^3}b(v^{-1}(y))^2, \tag{A.10}$$

$$\tilde{b}(y) = v'(v^{-1}(y))b(v^{-1}(y)) = \frac{b(v^{-1}(y))}{(v^{-1})'(y)}. \tag{A.11}$$

Using lemma 2 on the stationary processes Y_t and X_t allows us to derive the relationship between the two stationary densities f_X^∞ and f_Y^∞:

$$f_Y^\infty(y) \propto \frac{1}{\tilde{b}(y)^2}\exp\left(\int_0^y \frac{2\tilde{a}(x)}{\tilde{b}(x)^2}\mathrm{d}x\right) \tag{A.12}$$

$$= \frac{1}{\tilde{b}(y)^2}\exp\left(\int_0^y \frac{2a(v^{-1}(x))}{b(v^{-1}(x))^2}(v^{-1})'(x)\mathrm{d}x - \int_0^y \frac{(v^{-1})''(x)}{(v^{-1})'(x)}\mathrm{d}x\right) \tag{A.13}$$

$$= \frac{1}{\tilde{b}(y)^2}\exp\left(\int_0^{v^{-1}(y)} \frac{2a(x)}{b(x)^2}\mathrm{d}x + \int_{v^{-1}(0)}^0 \frac{2a(x)}{b(x)^2}\mathrm{d}x - [\log((v^{-1})'(x))]_0^y\right), \tag{A.14}$$

$$\propto (v^{-1})'(y)\frac{1}{b(v^{-1}(y))^2}\exp\left(\int_0^{v^{-1}(y)} \frac{2a(x)}{b(x)^2}\mathrm{d}x\right), \tag{A.15}$$

$$\propto (\nu^{-1})'(y) f_X^\infty(\nu^{-1}(y)). \tag{A.16}$$

□

Corollary 2. *Let F_X^∞ denote the cumulative distribution function (CDF) of P_X^∞, and let F_Y denote the CDF of an arbitrary distribution P_Y. Define $\nu \overset{def}{=} F_Y^{-1} \circ F_X^\infty$. If P_Y is continuous with density f_Y, then the distribution P_Y^∞ from lemma 4 is equal to P_Y with density $f_Y^\infty = f_Y$.*

Proof. For ν as defined, it follows that

$$\nu^{-1} = (F_X^\infty)^{-1} \circ F_Y, \tag{A.17}$$

$$(\nu^{-1})' = (((F_X^\infty)^{-1})' \circ F_Y) \cdot f_Y \tag{A.18}$$

$$= \frac{f_Y}{f_X^\infty \circ (F_X^\infty)^{-1} \circ F_Y)} \tag{A.19}$$

$$= \frac{f_Y}{f_X^\infty \circ \nu^{-1}}. \tag{A.20}$$

Then by lemma 4,

$$f_Y^\infty(y) \propto (\nu^{-1})'(y) f_X^\infty(\nu^{-1}(y)) = \frac{f_Y(y)}{(f_X^\infty \circ \nu^{-1})(y)} (f_X^\infty \circ \nu^{-1})(y) = f_Y.$$

This proves that $\nu = F_Y^{-1} \circ F_X^\infty$ indeed maps a process with stationary distribution P_X^∞ onto a process with the arbitrarily chosen stationary distribution P_Y. □

A.3 Autocorrelation of Filtered OU Process. Assuming that the input into the leaky integrating membrane potential X_t of the simple neuron model with $\mu_t = 0$ and $\sigma_t = \sigma_X$ is not Brownian motion but instead another (OU) process I_t (see lemma 3) with autocorrelation function

$$\mathbb{E}[I_{t_1} \cdot I_{t_2}] = \sigma_I^2 \exp(-\gamma |t_1 - t_2|),$$

then the resulting autocorrelation function $R(t_1, t_2)$ for X_t with $X_0 = 0$ is determined by the stochastic input I_t and can be derived as follows (we assume $t_1 = t_0, t_2 = t_0 + \Delta_t, \Delta_t \geq 0$):

$$R(t_1, t_2) = \mathbb{E}[X_{t_1} \cdot X_{t_2}] \tag{A.21}$$

$$= \mathbb{E}\left[\left(\sqrt{2\theta} \int_0^{t_1} \exp(\theta(s - t_1)) \sigma_X I_s \mathrm{d}s \right) \right.$$

$$\left. \cdot \left(\sqrt{2\theta} \int_0^{t_2} \exp(\theta(s - t_2)) \sigma_X I_s \mathrm{d}s \right) \right] \tag{A.22}$$

$$= \frac{2\theta\sigma_X^2}{\exp(\theta(t_1+t_2))}\int_0^{t_1}\int_0^{t_2}\exp(\theta(s_1+s_2))\mathbb{E}[I_{s_1}I_{s_2}]\mathrm{d}s_2\mathrm{d}s_1 \quad \text{(A.23)}$$

$$= \frac{2\theta\sigma_X^2\sigma_I^2}{\exp(\theta(t_1+t_2))}\int_0^{t_1}\int_0^{t_2}\exp(\theta(s_1+s_2)-\gamma|s_1-s_2|)\mathrm{d}s_2\mathrm{d}s_1 \quad \text{(A.24)}$$

$$= \frac{2\theta\sigma_X^2\sigma_I^2}{\exp(\theta(t_1+t_2))}\int_0^{t_1}\Bigg[\exp(s_1(\theta-\gamma))\int_0^{s_1}\exp(s_2(\theta+\gamma))\mathrm{d}s_2 + \exp(s_1(\theta+\gamma))\int_{s_1}^{t_2}\exp(s_2(\theta-\gamma))\mathrm{d}s_2\Bigg]\mathrm{d}s_1 \quad \text{(A.25)}$$

$$= \frac{2\theta\sigma_X^2\sigma_I^2}{\exp(\theta(t_1+t_2))}\int_0^{t_1}\Bigg[\frac{\exp(2s_1\theta)-\exp(s_1(\theta-\gamma))}{\theta+\gamma} + \frac{\exp(t_2(\theta-\gamma))\exp(s_1(\theta+\gamma))-\exp(2s_1\theta)}{\theta-\gamma}\Bigg]\mathrm{d}s_1 \quad \text{(A.26)}$$

$$= \frac{2\theta\sigma_X^2\sigma_I^2}{(\theta^2-\gamma^2)}\Bigg(\left(\frac{\gamma}{\theta}+1\right)\exp(-t_1\theta-t_2\theta)-\frac{\gamma}{\theta}\exp(-\triangle_t\theta) - \exp(-\gamma t_1-\theta t_2)+\exp(-\triangle_t\gamma)-\exp(-\gamma t_2-\theta t_1)\Bigg). \quad \text{(A.27)}$$

By considering the limit where both t_1 and t_2 are far from 0 and $t_2 - t_1 = \triangle_t$ and the influence of the initial condition vanishes, this can be simplified to a stationary autocovariance function:

$$R(\triangle_t) = \lim_{t_0\to\infty}\mathbb{E}[X_{t_1}\cdot X_{t_2}] \quad \text{(A.28)}$$

$$= \frac{2\sigma_X^2\sigma_I^2}{\theta^2-\gamma^2}(\theta\exp(-\triangle_t\gamma)-\gamma\exp(-\triangle_t\theta)). \quad \text{(A.29)}$$

A direct result from equation A.29 is that the variance of the membrane potential then is given by $R(0) = \frac{2\sigma_X^2\sigma_I^2}{\theta+\gamma}$.

A.4 Estimating Sufficient Statistics. By Itô's lemma, transforming the neuron's membrane potential process through the nonlinear sufficient statistics of its membrane potential distribution (assuming they satisfy the requirements) yields new Itô processes that subsequently can be filtered with a causal exponential filter. The resulting exponentially weighted running average approximates the expected values of the sufficient statistics.

Suddenly changing parameters of the membrane potential distribution takes effect on the membrane potential with a certain delay due to continuity of the membrane potential. Filtering a transformation introduces further delay. The dynamics of adaptation variables are in the following analytically derived for the special case of membrane potentials modeled by the Ornstein-Uhlenbeck process defined in lemma 3. Let X_t be an Ornstein-Uhlenbeck with time-varying mean and standard deviation μ_t and σ_t. We consider an adaptation variable $\alpha_t = \int_0^t \gamma \exp(\gamma(s-t))X_s \mathrm{d}s$. For $\tau \neq \gamma > 0$ and for $X_0 = 0$, we can use the deterministic part of the solution X_t as defined in lemma 3 to derive the expectation

$$\mathbb{E}[\alpha_t] = \int_0^t \gamma \exp(\gamma(s-t)) \int_0^s \theta \exp(\theta(r-s))\mu_r \mathrm{d}r\mathrm{d}s \tag{A.30}$$

$$= \int_0^t \int_0^s \theta\gamma \exp(s(\gamma-\theta)) \exp(-\gamma t + \theta r)\mu_r \mathrm{d}r\mathrm{d}s \tag{A.31}$$

$$= \int_0^t \int_r^t \theta\gamma \exp(s(\gamma-\theta))\mathrm{d}s \exp(-\gamma t + \theta r)\mu_r \mathrm{d}r \tag{A.32}$$

$$= \int_0^t \kappa(t-r)\mu_r \mathrm{d}r \tag{A.33}$$

$$= (\kappa \star \mu)(t), \tag{A.34}$$

$$\text{with } \kappa(x) \overset{\text{def}}{=} \frac{\gamma\theta}{\gamma-\theta}\left(\exp(-\theta x) - \exp(-\gamma x)\right). \tag{A.35}$$

The causal filter κ here is a valid probability density with mean $\int_0^\infty x\kappa(x)\mathrm{d}x = \frac{1}{\gamma} + \frac{1}{\theta}$. We thus see that the adaptation variable α_t approximates the membrane potential's true mean value, lagging behind with a delay on the scale of $\frac{1}{\gamma} + \frac{1}{\theta}$. A similar derivation can be done for the second adaptation variable, $\beta_t = \int_0^t \gamma \exp(\gamma(s-t))X_s^2 \mathrm{d}s$. The larger the two time constants θ and γ are chosen, the smoother, yet slower, the resulting adaptation variables approximate the true sufficient statistics. The noise in the adaptation variable α_t can be estimated by considering its variance, which, as calculated as in section A.3, is proportional to $\frac{2\sigma_X^2}{\theta+\gamma}$ and approaches 0 for a sufficiently slow adaptation time constant γ. We conclude that the proposed mechanism of estimating sufficient statistics via filtering is valid and asymptotically correct, and it follows the dynamics derived here. See Figure 2 to verify the match between derived and simulated adaptation parameter dynamics.

A.5 Approximation Quality of Adaptive Neurons. To measure the quality of an adaptive neuron's temporary estimate $f_X^{\bar{\phi}_t}$ of the current stationary membrane potential distribution f_X^∞, the Kullback-Leibler

divergence $D(f_X^{\bar{\phi}_t} || f_X^{\infty})$ between the two can be used. This measure is very informative, since it simultaneously represents the mismatch between the neuron's produced and desired activation distributions:

Lemma 5. *Let f_X^{∞} denote the stationary probability density of an adaptive neuron's membrane potential with parameters $\bar{\eta}_t$, and let $f_X^{\bar{\phi}_t}$ denote the neuron's estimate of the same density, instead parameterized by the neuron's adaptation variables $\bar{\phi}_t$. Further, let f_Y^{∞} denote the neuron's desired stationary distribution of activation and $f_Y^{\bar{\phi}_t}$ the distribution achieved by the adaptive neuron. Then the neuron's activation distribution approaches the desired distribution if and only if its estimated membrane potential distribution approaches the true membrane potential distribution and*

$$D(f_X^{\bar{\phi}_t} || f_X^{\infty}) = D(f_Y^{\infty} || f_Y^{\bar{\phi}_t}).$$

Proof. Let ν be the monotonous, continuous activation function of the neuron with support $(a, b) = \lim_{x\to\infty}(\nu(-x), \nu(x))$ that maps $f_X^{\bar{\phi}_t} \to f_Y^{\infty}$ and $f_X^{\infty} \to f_Y^{\bar{\phi}_t}$; then:

$$D(f_X^{\bar{\phi}_t} || f_X^{\infty}) = \int_{-\infty}^{\infty} f_X^{\bar{\phi}_t}(x) \cdot \log\left(\frac{f_X^{\bar{\phi}_t}(x)}{f_X^{\infty}(x)}\right) \mathrm{d}x \tag{A.36}$$

$$= \int_a^b f_X^{\bar{\phi}_t}(\nu^{-1}(y)) \cdot \log\left(\frac{(\nu^{-1})'(y) f_X^{\bar{\phi}_t}(\nu^{-1}(y))}{(\nu^{-1})'(y) f_X^{\infty}(\nu^{-1}(y))}\right) (\nu^{-1})'(y)\mathrm{d}y \tag{A.37}$$

$$= \int_a^b f_Y^{\infty}(y) \cdot \log\left(\frac{f_Y^{\infty}(y)}{f_Y^{\bar{\phi}_t}(y)}\right) \mathrm{d}y \tag{A.38}$$

$$= D(f_Y^{\infty} || f_Y^{\bar{\phi}_t}). \tag{A.39}$$

□

How well the desired distribution of activation can be achieved thus crucially depends on how well the neuron's adaptation parameters can approximate the sufficient statistics of the membrane potential distribution. Since the divergence on both membrane potential and activation side of the neuron is identical, we just refer to the divergence between neurons.

Using lemma 5, the divergence between two gaussian neurons can be calculated based on the membrane potential distribution's sufficient statistics and the corresponding approximation by the adaptation variables,

$$\bar{\eta} = \begin{pmatrix} \mathbb{E}[X] \\ \mathbb{E}[X^2] \end{pmatrix} = \begin{pmatrix} \mu \\ \sigma^2 + \mu^2 \end{pmatrix} \overset{?}{\approx} \bar{\phi}_t = \begin{pmatrix} \alpha_t \\ \beta_t \end{pmatrix}.$$

The Kullback-Leibler divergence is then defined as follows:

$$D(f_X^{\bar{\phi}_t} || f_X^{\infty}) = \int_{-\infty}^{\infty} f_X^{\bar{\phi}_t}(x) \cdot \log\left(\frac{f_X^{\bar{\phi}_t}(x)}{f_X^{\infty}(x)}\right) \mathrm{d}x \tag{A.40}$$

$$= \int_{-\infty}^{\infty} f_X^{\bar{\phi}_t}(x) \cdot \log\left(\frac{\sigma \exp\left(-\frac{(x-\alpha)^2}{2(\beta_t-\alpha_t^2)}\right)}{\sqrt{\beta_t-\alpha_t^2}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)}\right) \mathrm{d}x \tag{A.41}$$

$$= \log\left(\frac{\sigma}{\sqrt{\beta_t-\alpha_t^2}}\right) + \int_{-\infty}^{\infty} f_X^{\bar{\phi}_t}(x) \cdot \left(-\frac{(x-\alpha)^2}{2(\beta_t-\alpha_t^2)} + \frac{(x-\mu)^2}{2\sigma^2}\right) \mathrm{d}x \tag{A.42}$$

$$= \log\left(\frac{\sigma}{\sqrt{\beta_t-\alpha_t^2}}\right) - \frac{1}{2} + \frac{1}{2\sigma^2}\int_{-\infty}^{\infty} f_X^{\bar{\phi}_t}(x) \cdot (x^2 - 2x\mu + \mu^2)\mathrm{d}x \tag{A.43}$$

$$= \log(\sigma) - \frac{1}{2}\log(\beta_t - \alpha^2) + \frac{(\mu-\alpha_t)^2 + ((\beta_t-\alpha_t^2)-\sigma^2)}{2\sigma^2}. \tag{A.44}$$

We see that this converges to 0 for $\alpha_t \to \mathbb{E}[X] = \mu$ and $\beta_t \to \mathbb{E}[X^2] = \sigma^2 + \mu^2$.

A.6 Effects of the Choice of Activation Distribution. The choice of membrane potential distribution and activation distribution together determines the resulting activation function. While the computational role of the neuron is often explained based on the nonlinearity used, the distributions allow a different perspective. Consider first a discrete, bimodal probability distribution of activations, where the activation is either at a high, fixed value υ with probability p or a 0 with probability $1 - p$. The expected activation or firing rate then is $\upsilon \cdot p$, and the neuron switches between periods of high firing rates (bursts) and periods of low firing rates.

Due to the continuous nature of the membrane potential process, the periods of bursting can be expected to be dense, measurable intervals, which allows us to understand the neuron as a bursting neuron. Depending on the time constants of the membrane potential dynamics, these intervals could represent longer or shorter bursts of spikes. For sufficiently high rates υ and low p with a constant mean firing rate υp, the timing of spikes can be made precise, in particular if renewal processes with refractoriness are used to sample spikes, limiting the number of spikes per interval. The shape of the

distribution of firing rates thus allows influencing the temporal precision of spiking and controls the amount of noise introduced into spike trains sampled by an inhomogeneous Poisson process. For this example, the corresponding cumulative distribution function, and thus the nonlinearity, are step functions, and the neuron switches between stochastic bursting and silence. For continuous activation distributions, similar observations can be made. Highly curtotic distributions, where the mean is much larger than the median, are dominated by rare yet disproportionately high firing rates and can thus best be viewed as encoding such rare events that elicit high membrane potentials (coincidences). Depending on the spike sampling mechanism used, different distributions might yield optimal information transmission through spiking output.

Appendix B: Copulas and Joint Distributions

B.1 Preliminaries.

Lemma 6 (Copulas, Sklar's theorem; see Embrechts et al., 2001). *For an N-variate continuous random variable X with joint cumulative distribution function F and invertible marginal cumulative distribution functions F_i, the copula function $C : [0, 1]^N \to \mathbb{R}^+$ can be defined as follows:*

$$C(u) = \mathbb{P}(F_1(X_1) \le u_1, \ldots, F_N(X_N) \le u_N) \tag{B.1}$$

$$= \mathbb{P}(X_1 \le F_1^{-1}(u_1), \ldots, X_N \le F_1^{-1}(u_N)) \tag{B.2}$$

$$= F(F_1^{-1}(u_1), \ldots, F_1^{-1}(u_N)) \tag{B.3}$$

$$\Leftrightarrow F(x) = C(F_1(x_1), \ldots, F_N(x_N)). \tag{B.4}$$

Except information about the marginals, the copula thus contains all information about the joint distribution.

Corollary 3 (copulas and densities). *Using lemma 6 and the chain rule, a continuous N-dimensional probability distribution with joint density f, marginal densities $f_i > 0$, and marginal continuous distribution functions F_i can be decomposed into the product*

$$f(x) = c(F_1(x_1), \ldots, F_N(x_N)) \prod_i f_i(x_i), \tag{B.5}$$

where $c(x) = \frac{\partial^N C(x)}{\partial x_1 \cdots \partial x_N}$ is the copula density.

Lemma 7 (marginal invariance of copulas). *Let X be an N-variate continuous random variable with copula C_X and let $\nu_1, \ldots, \nu_N$ be monotone functions. Then $Y = (\nu_1(X_1), \ldots, \nu_N(X_N))$ and X have the same copula $C_Y = C_X$.*

Proof. Let C_X, F^X and F_i^X be the copula, joint, and marginal CDFs of X, respectively, and let C_Y, F^Y, and F_i^Y be the copula, joint, and marginal CDFs of

Y, respectively. Then $F_i^X(x) = \mathbb{P}(X_i \leq x) = \mathbb{P}(Y_i \leq \nu(x)) = F_i^Y(\nu(x))$ and applying lemma 6 twice yields

$$C_X(u) = \mathbb{P}(F_1^X(X_1) \leq u_1, \ldots, F_N^X(X_N) \leq u_N) \tag{B.6}$$
$$= \mathbb{P}(F_1^Y(\nu(X_1)) \leq u_1, \ldots, F_N^Y(\nu(X_N)) \leq u_N) \tag{B.7}$$
$$= \mathbb{P}(F_1^Y(Y_1) \leq u_1, \ldots, F_N^Y(Y_N) \leq u_N) \tag{B.8}$$
$$= C_Y(u). \tag{B.9}$$

□

B.2 Dependency of the Copula of the Joint Distribution of Activation on the Covariance of Gaussian Inputs. For a population of neurons driven by jointly gaussian input, the joint membrane potential distribution is determined by the stochastic input, particularly the covariance matrix R of its stationary distribution. The variance of each neuron i's membrane potential is equal to the variance of the signal driving the neuron, scaled down by a constant that depends on the dynamics of the input and the membrane potential process (see section A.3 for a derivation). Due to the invariance of the copula to marginal transformations, as guaranteed by lemma 7, the stationary distribution of the neurons' joint activation thus has the same copula C as the joint membrane potential distribution and the joint input distribution. The stationary, multivariate gaussian input process with covariance matrix R has a gaussian copula (Embrechts et al., 2001) of the following form, where Θ denotes the standard gaussian CDF:

$$c_R(u) = \frac{1}{\sqrt{|R|}} \exp\left(-\frac{1}{2} \begin{pmatrix} \Theta^{-1}(u_1) \\ \vdots \\ \Theta^{-1}(u_N) \end{pmatrix}^T (R^{-1} - I) \begin{pmatrix} \Theta^{-1}(u_1) \\ \vdots \\ \Theta^{-1}(u_N) \end{pmatrix} \right). \tag{B.10}$$

Given c_R, which depends on only the covariance matrix R of the stationary joint distribution of the signals driving a population of neurons and marginal probability densities of the neurons activation, the joint distribution of activation in a population is thus fully specified.

B.3 Inducing Structure through Input Weights. In a population, where N neurons receive input signals via weighted synaptic connections W from different source signals, which are jointly gaussian processes with stationary covariance matrix $\tilde{R}$ of rank $\geq N$, the signals driving the neurons are also jointly gaussian with stationary covariance matrix $R = W\tilde{R}W^T$. Decomposing the (pos. def.) matrix $\tilde{R} = \tilde{R}' \cdot \tilde{R}'^T$ (e.g. using the Cholesky-decomposition), the weight matrix could be chosen as $W = \tilde{R}'^{-1}$ (if more input signals than neurons are available, a pseudo-inverse could be used

here). The resulting covariance of the signals driving the population is then given by

$$R = W\tilde{R}W^T = \tilde{R}'^{-1}(\tilde{R}'\tilde{R}'^T)\tilde{(}R'^{-1})^T = (\tilde{R}'^{-1}\tilde{R}')(\tilde{R}'^{-1}\tilde{R}')^T = I, \tag{B.11}$$

that is, the neurons are completely decorrelated. For a desired covariance matrix Q, the same decomposition into $Q = Q' \cdot Q'^T$ can be used to define a weight matrix $W = Q'^{-1}\tilde{R}'^{-1}$, which results in the stationary covariance matrix:

$$R = (Q'\tilde{R}'^{-1})\tilde{R}(Q'\tilde{R}'^{-1})^T = Q'(\tilde{R}'^{-1}\tilde{R}\tilde{R}'^{-T})Q'^T = Q'Q'^T = Q. \tag{B.12}$$

By choosing the synaptic connection weights accordingly, arbitrary covariance structures can thus be induced in the input signals driving the neurons, assuming that at least as many uncorrelated input signals are available as there are neurons in the population. Using the results from section B.2, this implies that for gaussian inputs, any activation distribution with a gaussian copula can be realized by an appropriate choice of marginal distributions and synaptic weights.

The same reasoning can be applied across time, rather than neurons, when considering the autocovariance of a single neuron instead of the covariance between different neurons. If the autocovariance structure of a gaussian membrane potential process across different points in time is modeled by a multivariate gaussian distribution, it can again be decomposed into a (gaussian) copula and marginal distributions. The copula is preserved under the nonlinear transformation through the neuron's activation function and thus captures the autocovariance structure of its activation process as well. By adding multiple weighted synaptic connections with different delays for a single source signal, the autocovariance can be controlled in the same way as the covariance is controlled above.

Combining these two approaches, a set of weighted, delayed synaptic connections can be used to induce cross-covariance structures in the neurons' inputs and thus membrane potentials and activations.

Appendix C: PCA and ICA in Single Neurons

C.1 Sensitivity of Nonlinear Neurons to Higher Moments. Let X be a random variable with mean μ, and let $\nu : \mathbb{R} \to \mathbb{R}^+$ be a monotonically increasing function that is infinitely differentiable at μ. We consider the expected value of the output random variable $Y = \nu(X)$. Then by using the Taylor-series expansion of ν around μ, we can see that the expected value

$\mathbb{E}[Y]$ depends on the centered moments m_i of X:

$$\mathbb{E}[Y] = \mathbb{E}\left[\sum_{i=0}^{\infty} \frac{f^{(i)}(\mu)}{i!}(X-\mu)^i\right] \tag{C.1}$$

$$= \sum_{i=0}^{\infty} \alpha_i m_i^X \tag{C.2}$$

$$\text{where } \alpha_i = \frac{f^{(i)}(\mu)}{i!} \tag{C.3}$$

$$m_i^X = \mathbb{E}[(X-\mu)^i]. \tag{C.4}$$

For a linear function, higher-order coefficients are $\alpha_{i\geq 2} = 0$, and the expected value of the output depends on only the first and zero-order moment. For polynomials of order M, all moments up to order M influence the expected output of the function. For a general exponential function $\nu(x) = \exp(ax+b)$ with parameters a and b, all coefficients $\alpha_i = \frac{a^i}{i!}\exp(a\mu+b)$ are positive and form a decreasing sequence that converges to 0. In this case, the expected value of the output Y is a linear combination of all centered moments m_i of X with rapidly decreasing weights. For a gaussian random variable X, where the mean and all moments beyond the second are fixed at zero, the expected value of the output Y is thus dominated by the second moment, the variance, whereas for a zero mean, unit covariance Laplacian distribution the third and higher moments influence the expected value of Y. The relative dependence of a function's expected output Y on the moments of X is determined by the coefficients of its Taylor expansion (if it exists).

C.2 Fixed-Point Analysis of the PCA and ICA Neurons. A neuron that receives input from a multivariate stochastic process I_t with stationary covariance matrix Σ through synaptic connections with weight vector w_t has a membrane potential process X_t with standard deviation $\sigma_{w_t} = c \cdot \sqrt{w_t \Sigma w_t^T}$, where the proportionality constant c depends on the dynamics of the processes X_t and I_t (see also section A.3). We assume here that the membrane potential exhibits very fast adaptation dynamics and closely follows the much slower input process I_t (we choose time constants $\theta_X = 100$, $\theta_I = 10$), such that $X_t \approx c \cdot w_t I_t^t$. The activation of a neuron is given by $Y_t = \nu_{\phi_t}(X_t)$, where for a nonadaptive neuron, ϕ_t is constant, and we just write $Y_t = \nu(X_t)$. An adaptive neuron that compensates only for the changes in the scale of its membrane potential such as the gaussian or Laplacian models used here (the mean is fixed in these experiments) can thus be modeled by $Y_t = \nu_{\sigma_{w_t}}(X_t) = \nu(\frac{X_t}{\sigma_{w_t}})$. Using equation 2.15, the vector field of expected weight

changes can be calculated for an adaptive neuron:

$$\mathbb{E}[\frac{\mathrm{d}w_t^{(i)}}{\mathrm{d}t}] = \mathbb{E}\left[\delta(I_t^{(i)} \cdot Y_t - w_t^{(i)})\right] \tag{C.5}$$

$$\approx \delta\left(\mathbb{E}[I_t^{(i)} \cdot \nu(\frac{w_t I_t^T}{\sqrt{w_t \Sigma w_t^T}})] - w_t^{(i)}\right). \tag{C.6}$$

Similarly, a nonadaptive neuron yields

$$\mathbb{E}\left[\frac{\mathrm{d}w_t^{(i)}}{\mathrm{d}t}\right] \approx \delta\left(\mathbb{E}[I_t^{(i)} \cdot \nu(c \cdot (w_t I_t^T)] - w_t^{(i)}\right). \tag{C.7}$$

Since the stationary joint distribution of I_t is explicitly given in both experiments (a multivariate gaussian or a product distribution of two independent Laplace distributions, rotated by 0.3π), these expectations can be analytically calculated for the two neuron types (adaptive and nonadaptive) in both conditions (gaussian and Laplacian inputs). Locations of fixed points can be found where expected weight changes vanish to zero. The bottom panels of Figures 3 and 4 show the calculated vector field for both neuron types in both experimental setups, respectively.

Acknowledgments

We thank our anonymous referees for their constructive feedback, which greatly helped to improve this article.

References

Abbott, L. F., Varela, J. A., Sen, K., & Nelson, S. B. (1997). Synaptic depression and cortical gain control. *Science*, *275*(5297), 221–224. doi:10.1126/science.275.5297.221

Ashby, W. R. (1954). *Design for a brain*. New York: Wiley. http://archive.org/details/designforbrain00ashb

Bell, A. J., & Sejnowski, T. J. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, *7*(6), 1129–1159.

Bibby, B. M., Skovgaard, I. M., & Sørensen, M. (2005). Diffusion-type models with given marginal distribution and autocorrelation function. *Bernoulli*, *11*(2), 191–220. doi:10.3150/bj/1116340291

Bienenstock, E. L., Cooper, L. N., & Munro, P. W. (1982). Theory for the development of neuron selectivity: Orientation specificity and binocular interaction in visual cortex. *Journal of Neuroscience*, *2*(1), 32–48.

Brito, C. S. N., & Gerstner, W. (2016). Nonlinear Hebbian learning as a unifying principle in receptive field formation. *PLOS Computational Biology*, *12*(9), e1005070. doi:10.1371/journal.pcbi.1005070

Brown, T. H., Kairiss, E. W., & Keenan, C. L. (1990). Hebbian synapses: Biophysical mechanisms and algorithms. *Annual Review of Neuroscience*, *13*, 475–511. doi:10.1146/annurev.ne.13.030190.002355

Buesing, L., & Maass, W. (2010). A spiking neuron as information bottleneck. *Neural Computation*, *22*(8), 1961–1992. doi:10.1162/neco.2010.08-09-1084

Burrone, J., & Murthy, V. N. (2003). Synaptic gain control and homeostasis. *Current Opinion in Neurobiology*, *13*(5), 560–567. doi:10.1016/j.conb.2003.09.007

Capasso, V., & Bakstein, D. (2015). An introduction to continuous-time stochastic processes. New York: Springer. doi:10.1007/978-1-4939-2757-9

Embrechts, P., Lindskog, F., & McNeil, A. (2001). *Modelling dependence with copulas and applications to risk management*. Amsterdam: Elsevier.

Fairhall, A. L., Lewen, G. D., Bialek, W., & de Ruyter Van Steveninck, R. R. (2000). Multiple timescales of adaptation in a neural code. In *Proceedings of the 13th International Conference on Neural Information Processing Systems* (pp. 115–121). Cambridge, MA: MIT Press. http://dl.acm.org/citation.cfm?id=3008751.3008769

Fairhall, A. L., Lewen, G. D., Bialek, W., & de Ruyter Van Steveninck, R. R. (2001). Efficiency and ambiguity in an adaptive neural code. *Nature*, *412*(6849), 787–792. doi:10.1038/35090500

Földiák, P. (1990). Forming sparse representations by local anti-Hebbian learning. *Biological Cybernetics*, *64*(2), 165–170. doi:10.1007/BF02331346

Genest, C., & Favre, A.-C. (2007). Everything you always wanted to know about copula modeling but were afraid to ask. *Journal of Hydrologic Engineering*, *12*(4), 347–368. doi:10.1061/(ASCE)1084-0699(2007)12:4(347)

Gerstner, W., & Brette, R. (2009). Adaptive exponential integrate-and-fire model. *Scholarpedia*, *4*(6), 8427. doi:10.4249/scholarpedia.8427

Hromádka, T., DeWeese, M. R., & Zador, A. M. (2008). Sparse representation of sounds in the unanesthetized auditory cortex. *PLOS Biology*, *6*(1), e16. doi:10.1371/journal.pbio.0060016

Hyvärinen, A., & Oja, E. (1998). Independent component analysis by general nonlinear Hebbian-like learning rules. *Signal Processing*, *64*(3), 301–313. doi:10.1016/S0165-1684(97)00197-7

Hyvärinen, A., & Oja, E. (2000). Independent component analysis: Algorithms and applications. *Neural Networks*, *13*(4), 411–430. doi:10.1016/S0893-6080(00)00026-5

Isomura, T., Kotani, K., & Jimbo, Y. (2015). Cultured cortical neurons can perform blind source separation according to the free-energy principle. *PLOS Computational Biology*, *11*(12), e1004643. doi:10.1371/journal.pcbi.1004643

Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, *14*(6), 1569–1572. doi:10.1109/TNN.2003.820440

Jolivet, R., Lewis, T. J., & Gerstner, W. (2004). Generalized integrate-and-fire models of neuronal activity approximate spike trains of a detailed model to a high degree of accuracy. *Journal of Neurophysiology*, *92*(2), 959–976. doi:10.1152/jn.00190.2004

Keziou, A., Fenniri, H., Messou, K., & Moreau, E. (2013). Blind source separation of independent/dependent signals using a measure on copulas. In *Proceedings of the 21st European Signal Processing Conference* (pp. 1–5). Piscataway, NJ: IEEE.

Klambauer, G., Unterthiner, T., Mayr, A., & Hochreiter, S. (2017). Self-normalizing neural networks. arXiv:1706/02515 [cs, stat]

LeCun, Y., Cortes, C., & Burges, C. (N.d.). *MNIST handwritten digit database.* http://yann.lecun.com/exdb/mnist/

Ma, J., & Sun, Z. (2011). Mutual information is copula entropy. *Tsinghua Science and Technology*, *16*(1), 51–54. doi:10.1016/S1007-0214(11)70008-6

Oja, E. (1982). Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, *15*(3), 267–273. doi:10.1007/BF00275687

Øksendal, B. (2003). Stochastic differential equations. In S. Axler, V. Capasso, C. Casacuberta, A. MacIntyre, K. Ribet, C. Sabbah, . . . W. Woyczyński (Eds.), *Stochastic differential equations* (pp. 65–84). Berlin: Springer. doi:10.1007/978-3-642-14394-6_5

Ostojic, S., & Brunel, N. (2011). From spiking neuron models to linear-nonlinear models. *PLOS Computational Biology*, *7*(1), e1001056. doi:10.1371/journal.pcbi.1001056

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.*, *12*, 2825–2830. http://dl.acm.org/citation.cfm?id=1953048.2078195

Ricciardi, L. M., & Lánský, P. (2006). Diffusion models and neural activity. In L. Nodel (Ed.), *Encyclopedia of Cognitive Science*. Hoboken, NJ: Wiley. doi:10.1002/0470018860.s00365

Savin, C., Joshi, P., & Triesch, J. (2010). Independent component analysis in spiking neurons. *PLOS Computational Biology*, *6*(4), e1000757. doi:10.1371/journal.pcbi.1000757

Shinomoto, S., Sakai, Y., & Funahashi, S. (1999). The Ornstein-Uhlenbeck process does not reproduce spiking statistics of Neurons in prefrontal cortex. *Neural Computation*, *11*(4), 935–951. doi:10.1162/089976699300016511

Stemmler, M., & Koch, C. (1999). How voltage-dependent conductances can adapt to maximize the information encoded by neuronal firing rate. *Nature Neuroscience*, *2*(6), 521–527. doi:10.1038/9173

Stepanov, S. S. (2013). *Stochastic world*. Heidelberg: Springer. doi:10.1007/978-3-319-00071-8

Sweeney, Y., Kotaleski, J. H., & Hennig, M. H. (2015). A diffusive homeostatic signal maintains neural heterogeneity and responsiveness in cortical networks. *PLoS Computational Biology*, *11*(7), e1004389. doi:10.1371/journal.pcbi.1004389

Toyoizumi, T., Kaneko, M., Stryker, M., & Miller, K. (2014). Modeling the dynamic interaction of Hebbian and homeostatic Plasticity. *Neuron*, *84*(2), 497–510. doi:10.1016/j.neuron.2014.09.036

Toyoizumi, T., Pfister, J.-P., Aihara, K., & Gerstner, W. (2005). Generalized Bienenstock Cooper Munro rule for spiking neurons that maximizes information transmission. *Proceedings of the National Academy of Sciences of the United States of America*, *102*(14), 5239–5244. doi:10.1073/pnas.0500495102

Triesch, J. (2007). Synergies between intrinsic and synaptic plasticity mechanisms. *Neural Computation*, *19*(4), 885–909. doi:10.1162/neco.2007.19.4.885

Truccolo, W., Eden, U. T., Fellows, M. R., Donoghue, J. P., & Brown, E. N. (2005). A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects. *Journal of Neurophysiology*, *93*(2), 1074–1089. doi:10.1152/jn.00697.2004

Tsymbal, A. (2004). *The problem of concept drift: Definitions and related work* (Tech. Rep.). Dublin: Department of Computer Science, Trinity College Dublin.

Turrigiano, G. G. (2008). The self-tuning neuron: Synaptic scaling of excitatory synapses. *Cell*, *135*(3), 422–435. doi:10.1016/j.cell.2008.10.008

Turrigiano, G. G., & Nelson, S. B. (2004). Homeostatic plasticity in the developing nervous system. *Nature Reviews Neuroscience*, *5*(2), 97–107. doi:10.1038/nrn1327

Winder, S. A., & Impressive Machines LLC. (N.d.). *Natural image patch database.* http://simonwinder.com/2015/08/natural-image-patch-database/

Zucker, R. S., & Regehr, W. G. (2002). Short-term synaptic plasticity. *Annual Review of Physiology*, *64*(1), 355–405. doi:10.1146/annurev.physiol.64.092501.114547

Received June 19, 2017; accepted November 8, 2017.

EVENT-BASED PATTERN DETECTION IN ACTIVE DENDRITES

PREPRINT

Johannes Leugering[*]
Osnabrück University, Germany
jleugeri@uni-osnabrueck.de

Pascal Nieters[*]
Osnabrück University, Germany
pnieters@uni-osnabrueck.de

Gordon Pipa
Osnabrück University, Germany
gpipa@uni-osnabrueck.de

August 17, 2020

ABSTRACT

Many behavioural tasks require an animal to integrate information on a slow timescale that can exceed hundreds of milliseconds. How this is realized by neurons with membrane time constants on the order of tens of milliseconds or less remains an open question. We show, how the interaction of two kinds of events within the dendritic tree, *excitatory postsynaptic potentials* and locally generated *dendritic plateau potentials*, can allow a single neuron to detect specific sequences of spiking input on such slow timescales. Our conceptual model reveals, how the morphology of a neuron's dendritic tree determines its computational function, which can range from a simple logic gate to the gradual integration of evidence to the detection of complex spatio-temporal spike-sequences on long timescales. As an example, we illustrate in a simulated navigation task how this mechanism can even allow individual neurons to reliably detect specific movement trajectories with high tolerance for timing variability. We relate our results to conclusive findings in neurobiology and discuss implications for both experimental and theoretical neuroscience.

Author Summary

The recognition of patterns that span multiple timescales is a critical function of the brain. This is a conceptual challenge for all neuron models that rely on the passive integration of synaptic inputs and are therefore limited to the rigid millisecond timescale of post-synaptic currents. However, detailed biological measurements recently revealed that single neurons actively generate localized plateau potentials within the dendritic tree that can last hundreds of milliseconds. Here, we investigate single-neuron computation in a model that adheres to these findings but is intentionally simple. Our analysis reveals how plateaus act as memory traces, and their interaction as defined by the dendritic morphology of a neuron gives rise to complex non-linear computation. We demonstrate how this mechanism enables individual neurons to solve difficult, behaviorally relevant tasks that are commonly studied on the network-level, such as the detection of variable input sequences or the integration of evidence on long timescales. We also characterize computation in our model using rate-based analysis tools, demonstrate why our proposed mechanism of dendritic computation cannot be detected under this analysis and suggest an alternative based on plateau timings. The interaction of plateau events in dendritic trees is, according to our argument, an elementary principle of neural computation which implies the need for a fundamental change of perspective on the computational function of neurons.

Introduction

The ability to detect long-lasting sequences of neural activity is crucial for complex behavior, but poses a serious challenge for most established neuron models. Consider a rodent navigating through an environment in search for food. Receptive fields of place and grid cells tile a spatial map of the environment and encode the current position by their respective population activities [1, 2]. But in order to find its way back, the animal needs to know not only its present location, but also which path it took to get there. Decoding this path from the sequential activation of place and grid cells requires the integration of information on behavioural timescales that can span hundreds of milliseconds or

[*]Both authors contributed equally.

more [3, 4]. Relevant patterns on such long timescales may prove to be a ubiquitous phenomenon, and have already been documented for a wide range of sensory processing tasks, such as olfaction [5, 6] or cortical auditory processing [7].

This raises the puzzling question, how such long sequences of neural activity can be processed by volatile neurons with membrane time constants on the timescale of tens of milliseconds or less [8]. While this problem is typically addressed on a network level, e.g. by relying on effects of fast-acting synaptic plasticity [9] or slow emergent dynamics due to recurrent connections [10], we argue that it can be solved on the level of individual neurons by active processes within their dendritic trees. These localized processes endow neurons with internal memory traces on the timescale of hundreds of milliseconds, and can be captured in a simple, conceptual model that adheres to recent biological evidence not accounted for in integrate-and-fire neuron models.

By investigating the computational properties of neurons with active dendrites, we draw three conclusions. Firstly, active dendritic processes can implement complex spatio-temporal receptive fields for ordered sequences of synaptic inputs. Secondly, active dendritic processes enable the robust integration of weak signals over timescales much longer than post-synaptic responses. Thirdly, when analyzed from a rate-coding perspective, active dendritic processes implement sophisticated non-linear computations that are characterized by the neuron's dendritic morphology.

We demonstrate these propositions in a general computational framework for event-based, active dendritic sequence processing (ADSP), which offers an elegant solution to the problem of detecting highly variable, long lasting patterns in a neuron's input.

The functional role of active dendritic processes.

We derive our abstract model of dendritic computation from a few basic biological observations: Most of a cortical pyramidal neuron's excitatory synaptic inputs terminate on dendritic spines [11], where post-synaptic ion channels are activated via the stochastic, pre-synaptic release of glutamate-carrying vesicles [12, 13]. The activated channels, primarily controlled by α-amino-3-hydroxy-5-methyl-4-isoxazolepropionic acid receptors (AMPARs) [14], become conductive to a mixture of ions, which leads to a brief depolarization in the corresponding spine, referred to as the *excitatory post-synaptic potential* (EPSP) [15]. These voltage changes in nearby spines induce a modest depolarization in the local dendritic membrane potential [16], which passively propagates along the dendrite as described by neural cable theory (**Fig. 1c**). For very specific branching patterns, the passive propagation of activity along a neuron's dendrite can be simplified to an equivalent model of a cylinder, in which the contribution of individual synaptic inputs sum (sub-)linearly [17]. Since propagation along the cylinder is very fast, abstract point-neuron models such as leaky integrate-and-fire neurons ignore the spatial dimension of the dendritic tree entirely and model the neuron as if it were a single electric compartment [18]. However, in this purely passive model of dendritic integration, the attenuation of signals along the dendritic cable is so strong, that synaptic input onto thin apical dendrites should have little, if any, measurable effect on the membrane potential at the soma far away [19, 20]. A synaptic plasticity mechanism that proportionally up-scales synaptic efficacies depending on the synapses' distance to the soma may counteract this phenomenon. Aptly termed "dendritic democracy"[21], it has been shown in hippocampal pyramidal neurons [22], where it results in a similar contribution of synaptic inputs onto the somatic membrane potential — regardless of the synapse's position along the dendrite. We instead look at a different mechanism to boost weak synaptic inputs, which relies on localized depolarizations that are actively generated and maintained within the dendritic tree.

Such active dendritic processes are ubiquitous [23, 24] and largely rely on N-methyl-D-asparate receptor (NMDAR) gated ion-channels [14] (see **Fig. 1c** for a schematic representation of this mechanism). NMDAR gated channels, like their AMPAR gated counterparts, are activated in the presence of glutamate, but do not become conductive unless a channel-blocking Mg^{+} ion is first displaced by a sufficiently strong depolarization [25, 26]. This depolarization can be achieved by the coactivation of multiple AMPAR channels on nearby spines within a short time-window. Experimental as well as simulation studies report that this requires a volley of 4-20 or even up to 50 spikes within 1-4ms, depending on the location along the dendritic tree [16, 27, 28, 29]. The opening of NMDAR channels triggers a massive influx of different ionic currents that lead to a complete depolarization of a small segment of the dendritic arbor. While the isolated NMDAR response itself is reported to last on the order of at least 25ms [30], in vivo recordings reveal that voltage-gated channels in the dendritic membrane [20] prolong this effect, resulting in a depolarization that can last from tens to hundreds of milliseconds [31]. We focus on these longer lasting events, which we collectively refer to as *dendritic plateau potentials*, and argue, that they provide useful memory traces within the dendritic tree that can last hundreds of milliseconds.

The much larger depolarization during a plateau potential propagates further along the dendrite than the weaker effect of individual EPSPs and thus extends the range at which they can contribute to somatic action potential generation. This may even be required for generating or spiking [32] or bursting [33] output. Just like EPSPs, however, plateau potentials are still subject to considerable attenuation along the dendritic cable and thus have a strong effect only in

their direct neighbourhood[2]. This leads to a division of complex dendritic arbors into functional subunits [34, 35, 36], which we here refer to as *dendritic segments*. How local plateau potentials in these segments interact within a dendritic tree depends on its morphology. In particular, the depolarizing effect on other directly connected dendritic segments is effectively raising their resting potential for the whole duration of the plateau potential, thus lowering the amount of coinciding spikes required to initiate a plateau potential there [37]. As [38] demonstrates, this local nonlinear interaction of dendritic segments due to NMDAR-gated channels can allow neural dendrites to become selective to specific sequences of synaptic inputs. While their work uses a biophysical, spatially extended neuron model to explain this behaviour, we instead derive a much simplified model composed of discrete dendritic segments. This helps explain how local interactions between connected segments lead to cascades of plateau potentials, which in turn allow the detection of specific long-lasting sequences within the dendritic tree.

Each segment of a dendritic tree tends to receive strongly correlated volleys of spikes on clustered synaptic inputs from some subpopulation of neurons [39, 40]. We suppose, that such incoming spike volleys constitute elementary events that convey relevant information. Then, the morphology of the dendritic tree then determines how this information is processed and retained in memory, and thereby endows the ADSP neuron with an intricate computational function.

The interaction of active dendritic processes realizes event-based computation.

We construct an abstract mathematical model of active dendritic sequence processing, that is firmly rooted in the previous biological observations. Conceptually, the complex dynamics of dendritic membrane potentials are reduced to the interactions of two kinds of events, EPSPs and actively generated plateau potentials, in a tree structure of dendritic segments. Since both of these events result in localized stereotypical effects on the dendritic membrane potential, we abstractly model them simply as rectangular pulses of unit magnitude and fixed duration τ_{synapse} and τ_{dendrite}, respectively. Because the qualitative behaviour of the dendritic arbor is thus explained purely in terms of the locations and times at which EPSPs and plateau potential are initiated in its dendritic segments, our model concisely describes dendritic computation.

Only those incoming spikes that are successfully transmitted by the probabilistic synapses induce EPSPs in the postsynaptic segment, which sum up and constitute the total *synaptic input* into the segment. This input is particularly strong when a *volley* of multiple spikes occurs in a time-window short enough for their EPSPs to overlap. In addition to synaptic input, the electric coupling between directly connected dendritic segments provides another source of *dendritic input*.

When both the *synaptic* and *dendritic input* into a segment exceed critical thresholds, the segment enters a prolonged *plateau* state. For the whole duration of the plateau, all other directly connected segments receive depolarizing dendritic input. Segments of the dendritic tree therefore act as coincidence detectors that respond to highly synchronized volleys of spikes with plateau potentials. The precise thresholds for synaptic and dendritic input depend on the segment's location within the dendritic tree. While a large volley of spikes alone suffices to trigger a plateau in the outermost segments of the dendritic tree, internal segments require the additional dendritic input due to plateau potentials in connected segments. For segments that lie at branching points in the dendritic tree, more than one of their neighbours may have to be in a plateau state concurrently to have a sufficient effect. If the soma, which lies at the root of the dendritic tree, receives sufficient synaptic and dendritic input, a somatic action potential, rather than a plateau potential, is generated.

Since the small effects of EPSPs remain confined to the postsynaptic dendrite segment, they only affect the neuron's behaviour indirectly by contributing to the generation of local plateau potentials. It is the plateau potentials and their interaction across neighbouring segments that drives the dendritic membrane potential, and therefore implements an event-based framework of dendritic computation on two distinct timescales orders of magnitude apart. On a fast timescale, the combined effect of a volley of coincident spikes initiates a localized plateau potential. On a much slower timescale, the interaction of these plateaus provides an ephemeral memory of the recent history. The computation we have described here is fully formalized in terms of synaptic spikes and plateau events as provided in the Methods section.

In **Fig. 1** we describe an exemplary ADSP neuron that receives input from five populations of neurons on five segments (**Fig. 1a**). Each segment, if sufficiently excited, responds to a spike volley in its respective input populations by emitting a plateau event at the time of the volley (**Fig. 1b**). The morphology of the dendritic tree determines how these plateaus interact along the dendritic tree. For example, segment C will only activate if both segments A and B are already active once segment C receives a spike volley. We formalize the relative timing requirement for these three segments by the expression $(A + B) \to_2 C$, which indicates that all two child branches A and B must be simultaneously active to enable the parent segment C, allowing it to emit a plateau in response to a spike-volley. We read this as "A and B,

[2]Unlike EPSPs, this attenuation cannot be circumvented by synaptic scaling as for dendritic democracy.

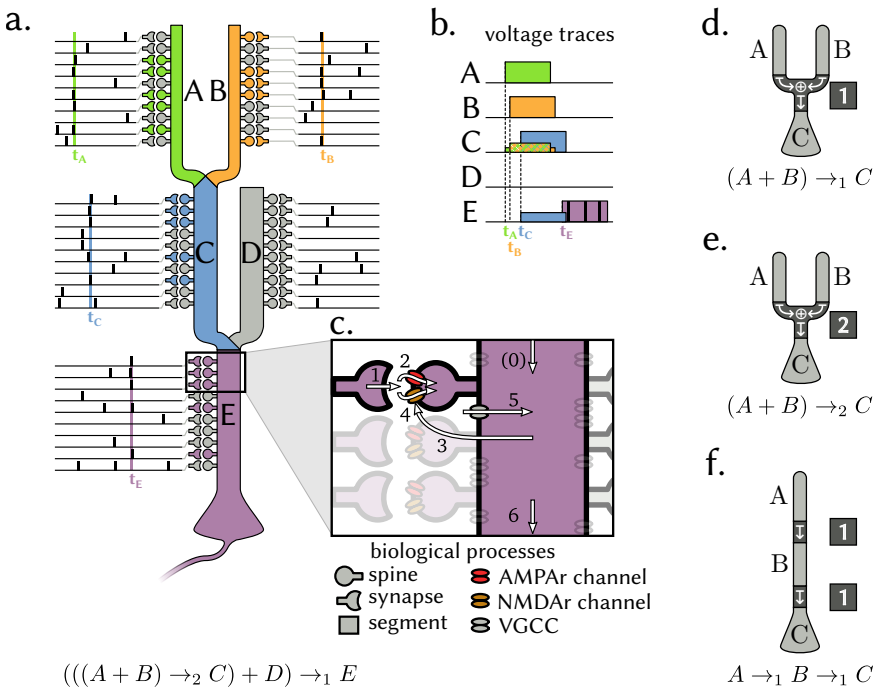


Figure 1: Schematic representation of a complex dendritic tree and its function. **a** A neuron receives on each of its 5 dendritic segments 10 synaptic connections from a corresponding neural population. Sufficiently many coincident spikes (here ≥ 6 out of 10) from population A lead the corresponding dendritic segment to generate a plateau potential (t_A). Similarly, coincident spikes from population B induce a plateau in a parallel branch (t_B). A third segment requires simultaneous input from both of these segments in addition to coincident synaptic input from population C, in order to fire a plateau of its own (t_C). On another branch, a fourth segment receives its input from population D but does not trigger a plateau. A somatic spike is triggered when coincident synaptic input from population E arrives (t_E) during dendritic input from either of its two upstream segments (in this case C). **b** Local membrane potentials show a cascade of plateau potentials. **c** The steps involved in the generation of a plateau: The membrane potential is already elevated due to a plateau potential in a neighbouring segment (0). Presynaptic input arrives at a synapse (1), which leads to a postsynaptic EPSP via AMPAr mediated ion channels (2). Once the local membrane potential is sufficiently depolarized due to coincident EPSPs and prior depolarization, voltage gated, NMDAr mediated ion channels open, causing additional depolarization (4) which can be further facilitated by the opening of voltage gated calcium channels (5). This strong depolarization initiates a longer lasting plateau potential in the dendritic segment, which has a modest depolarizing effect on other neighbouring segments (6). Different dendritic morphologies correspond to different computed functions, indicated in the respective formula under each schematic illustration. **d** If activating one of two dendritic branches with input from either population A or B, followed by a somatic spike initiated by input from population C, is sufficient to produce a spike, the neuron implements the operation $(A + B) \rightarrow_1 C$, which constitutes an "or"-operation between population A and B. **e** If simultaneous input from A and B is required, the neuron calculates an "and"-operation between inputs A and B. **f** A simple neuron that requires sequential activation of first A "and then" B before C.

and then C" (see also **Fig. 1d**). If the threshold was lowered, such that input from either segment A or B alone would suffice, the expression would correspondingly become $(A + B) \rightarrow_1 C$, which translates to "A or B, and then C" (see also **Fig. 1 e**). Generally, the expression $(X_1 + X_2 + \ldots + X_n) \rightarrow_m Y$ translates to "At least m out of the n segments $X_1, X_2, \ldots, X_n$ must be simultaneously active to enable segment Y". By chaining multiple segments together, these timing relations and nonlinear combinations can be arbitrarily nested, as for example in **Fig. 1f** that shows a neuron implementing $A \rightarrow_1 B \rightarrow_1 C$, which we read as "A, and then B, and then C". Using this formal notation, we express the complex ADSP neuron example in **Fig. 1a** as $(((A + B) \rightarrow_2 C) + D) \rightarrow_1 E$, a computation on spike volleys originating from the input populations associated with segments $A, \ldots, E$.

The interaction between connected dendritic segments facilitates cascades of plateau potentials along the dendritic tree, as illustrated in **Fig. 1b**. Starting in a distal segment, a leaf-node in our diagrams, a spike volley can initiate a plateau, which then provides dendritic input for the parent segment. Next, that segment responds to an incoming spike volley with a plateau of its own, in turn providing dendritic input to yet another segment. Whenever such a continuous chain of plateau potentials proceeds all the way to the soma, it culminates in a somatic action potential.

This signals to other neurons, that a specific sequence of spike volleys has been detected – on a timescale that may be as long as the number of segments times the plateau duration, i.e. hundreds of milliseconds. The precise timing between spike volleys is not prescribed exactly, as long as the distance between two successive volleys does not exceed the duration of one plateau potential. This invariance is critical whenever the precise timing of the individual events can vary, e.g. due to external circumstances such as varying movement speeds along a path in navigation tasks or due to neural mechanisms such as sleep replay [41], because it allows the neuron to generalize over all such perturbations. The branching morphology of a dendritic tree therefore determines the computation performed by the neuron, which allows even single neurons to detect complex compositions of sequential patterns. This event-based computation is what we call active dendritic sequence processing (ADSP).

Results

Dendritic processing allows the rapid detection of long, time-invariant patterns

To demonstrate the implications of such neuronal sequence detection, we return to the example of a rat navigating an environment. We assume that the rat has an internal representation of its environment, tiled by the receptive fields of distinct populations of place cells. While the animal resides within such a receptive field, the corresponding population emits spike volleys with a magnitude that is largest when the animal is close to the center of the receptive field. Different paths lead the animal through some of these receptive fields in different order, and result in different sequences of spike volleys.

Each individual spike volley consists of several coincident spikes, the EPSPs of which have to be integrated and thresholded on a millisecond time-scale to detect sufficiently significant events in the presence of noise. To detect whether the animal has taken a specific path through the environment, only specific sequences of such significant spike volleys must be detected on a much slower behavioural time-scale. These two distinct timescales pose a challenge for conventional spiking neuron models, which is further exacerbated by the fact, that the precise timing of the spike-volleys can vary substantially, depending e.g. on the speed with which the animal traverses its environment. While a solution to this problem may be found on a population level, we illustrate in **Fig. 2** how a single neuron can implement a solution very elegantly with just three active dendritic segments.

To simulate the rat's behaviour, we generate random movement trajectories through the environment by a stochastic process (see Methods section). Each place-cell population fires spike-volleys with a magnitude determined by the population's tuning-curve, a two-dimensional Gaussian function centered at the population's preferred location on a hexagonal grid. In this example, we are interested in paths that traverse three specific receptive fields, respectively color-coded in blue, orange and purple, and hence look at a neuron that consists of a chain of three dendritic segments, each receiving input from just one of these place-cell populations (**Fig. 2b**). The only trajectories that effectively drive the neuron to spike are those that sequentially traverse the three receptive fields in the correct order Blue $\rightarrow_1$ Orange $\rightarrow_1$ Purple (**Fig. 2a**).

During the example path shown in solid black, the three place cell populations are activated in the correct order over the course of 200ms and emit sufficiently large spike volleys to trigger a cascade of plateau potentials that lead the neuron to emit a somatic spike **Fig. 2b**. To illustrate how reliable of a detector an individual neuron can be — even when its synaptic inputs are stochastic with a transmission probability of 0.5 —, we systematically evaluate the probability of the neuron to fire in response to different paths with varying directions and lateral offsets. For an ideal straight 200ms long path through the center of all three place cell populations, the firing probability of the neuron is around 75%. When the orientation of the path is varied, this probability sharply decreases to 0%, indicating that the neuron is both highly sensitive and highly specific for paths with this orientation (**Fig. 2c**). Similarly, when the path is shifted orthogonally to the movement direction, the response probability falls quickly, confirming that the neuron is sensitive to the absolute location of the path as well as its direction (**Fig. 2d**).

A remarkable feature of this mechanism is, that it is invariant to changes in the precise timing of the individual volleys as long as two consecutive segments are activated within one plateau duration τ_{dendrite} of each other. The ADSP Neuron can therefore detect paths of any duration from 0ms to $N\tau_{\text{dendrite}}$ms, where $N = 3$ is the number of consecutive segments. We believe this source of timing-invariance to be a highly beneficial feature for generalization that helps explain phenomena, where the same sequence of events must be detected across multiple timescales.

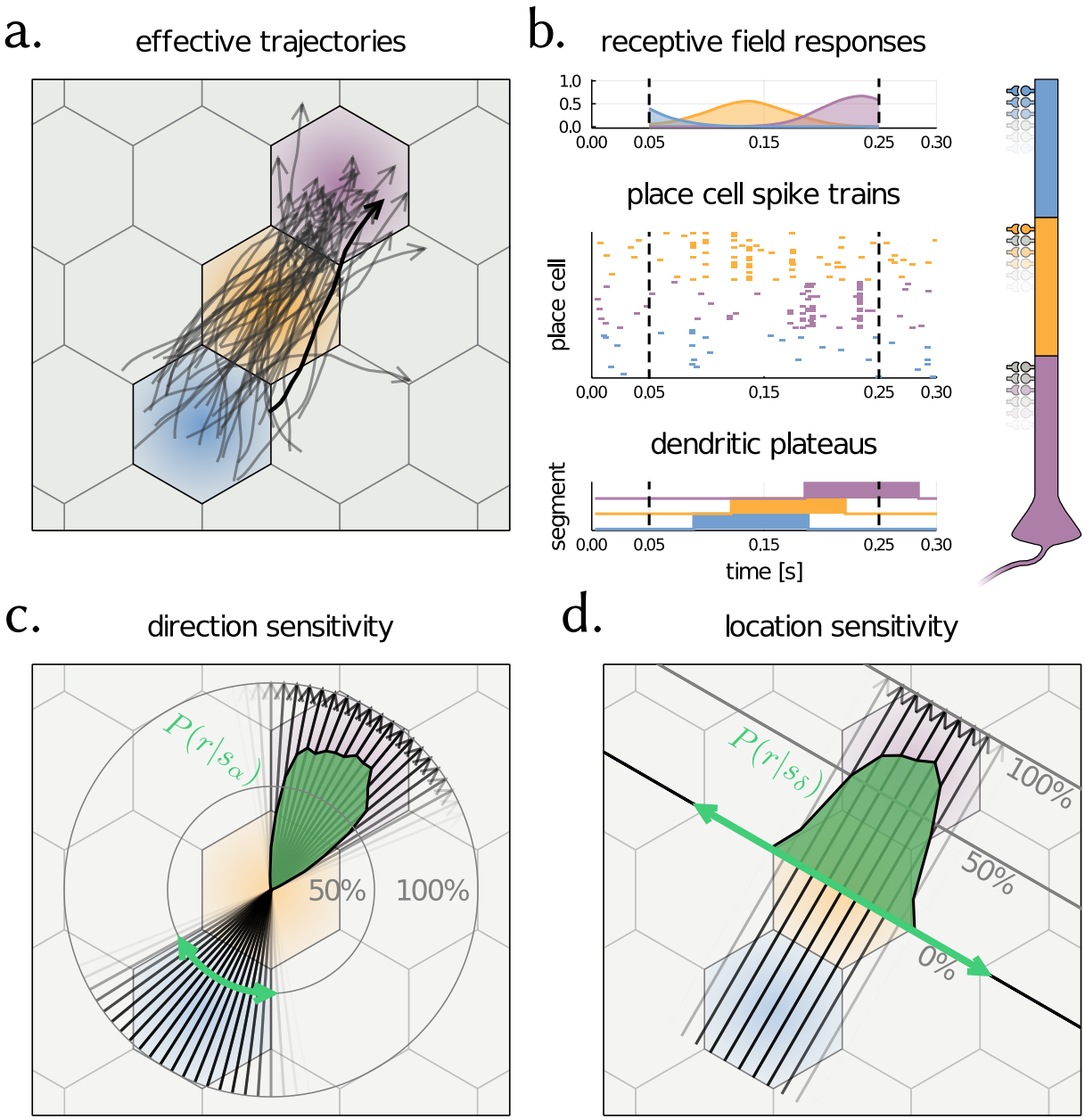


Figure 2: A simple neuron with three dendritic segments arranged as shown to the right of panel b can detect directed paths on a timescale of $300ms$. **a.** The receptive fields of place cell populations tile the environment through which the animal moves in a hexagonal grid. Random trajectories are generated through a stochastic process with randomized initial positions, velocities and angular heading to simulate the animal's movements. **b.** While the animal follows the black trajectory through space, the response of the place cell populations' tuning curves show the sequential activation of the populations over time (top panel). The stimulus s are generated spikes (middle panel) that lead to a temporal sequence of dendritic plateaus (bottom panel) and results in a somatic spike as the response r. **c. and d.** The neuron responds with high probability to exactly those paths that traverse the desired receptive fields in the correct direction and with little lateral offset. In the experiment, a change of rotation leads to paths s_α (**c** black) or a shift orthogonal to the movement direction leads to paths s_δ (**d** black) as indicated by the green arrows. The empirical probability of firing responses $P(r|s_\alpha)$ and $P(r|s_\delta$ respectively are shown in superimposed density plots in green in polar (**c**) and Cartesian (**d**) coordinates and show a highly specific pattern detector.

Plateaus integrate evidence on long timescales

In the previous example, specific paths are recognized by memorizing the sequential activation of different neural populations on a slow behavioural time-scale. A seemingly different, yet in fact closely related problem is the integration of individually unreliable bits of evidence over time. Consider, for example, a population of neurons that extract some relevant feature of a stimulus, such as the local movement direction in a visual moving dots stimulus. If we assume a retinotopic mapping, neighbouring neurons are highly correlated, and whenever the local movement direction is apparent, we expect a couple of neighbouring neurons coding for that direction to produce a volley of spikes. However, these events are unlikely to occur at the exact same point in time throughout the entire input space. The decision, whether or not the visual flow is in a certain direction, therefore requires that a neuron can integrate many such pieces of evidence, each indicated by a spike volley event, over a longer time-scale. Despite the all-or-none response of dendritic plateaus, a neuron with sufficiently many dendritic segments can in fact approximate such a smooth integration of evidence on timescales of hundreds of milliseconds!

We give an example of evidence integration using dendritic plateau potentials in a simplified experiment, in which a neuron with 1000 dendritic compartments receives input from a population of 1000 input neurons through a total of $20,000$ stochastic synapses (**Fig. 3**). The weak signal to be integrated by the ADSP neuron is encoded into spike volleys of 10 simultaneous spikes from adjacent neurons of the input population. Each dendritic segment of the ADSP neuron is connected to a different set of 20 adjacent neurons in the input population, and a total of 300 dendritic segments are required be in simultaneous plateau states for the neuron to emit a somatic spike.

Because each spike volley is likely to activate a different dendritic segment, we expect the number of simultaneously active dendritic compartments to reflect the average rate of incoming spike volleys during a time-interval of one plateau duration. This corresponds to a filtering of the time-varying rate by a rectangular filter, and, for a brief interval after stimulus onset, represents an ideal integrator. We observe this exact behavior by driving the rate, at which spike volleys are generated by the input population, to three different levels for brief time-intervals (**Fig. 3b**, orange line). The number of co-activated dendritic segments (blue line) closely follows the theoretical prediction of an ideal rectangular filter (black dashed line) until saturation. In particular, during the rising flanks right after stimulus onset (**Fig. 3c, d and e.**), we see the number of co-active segments rise with a slope proportional to the intensity of the stimulus until it saturates after 100ms. The neuron begins firing spikes once sufficiently many segments are active (red line). This is exactly the behavior expected for evidence integration: The ADSP neuron will fire sooner if the amount of evidence encoded in the stimulus is stronger, and will not fire at all if it remains sub-critical.

Interestingly, the stochasticity of synaptic transmission helps to further decorrelate the partially overlapping input to different dendritic segments, and can regulate the total amount of evidence required to reach the neuron's physiologically fixed spiking threshold. Also, while the example here makes use of just a single "layer" of dendritic segments directly driving the soma, this idea can be extended to deeper chains of multiple segments, such as in the previous example, to allow for the integration of evidence and non-linear combination thereof on timescales even longer than one plateau duration.

Dendritic morphology determines computational function

In the two previous examples, we assume that each dendritic segment is driven by well-timed volleys of coincident spikes, the magnitudes of which represent the magnitude of an underlying signal. But in theoretical neuroscience, the function of a neuron is often analyzed in a rate-based framework, which relates only the average firing rate of a neuron to the average firing rates of its spiking inputs.

Applying this sort of analysis to our proposed neuron model reveals, how different morphologies of dendritic arbors give rise to different non-linear computations. A dendritic segment driven by independent Poisson spike-trains originating from some population A of 25 neurons respond by triggering plateau potentials at a rate $\varrho(r_A)$ that continuously depend on the fixed firing-rate r_A of the populations' neurons. Here, 8 coincident spikes are required to trigger a plateau. As each plateau lasts for 100ms, ϱ saturates at a rate of 10 plateaus per second for large inputs (**Fig. 4a**). In more complex neurons composed of three dendritic segments, each of which is driven by an identical but independent population of neurons, we analyze the relative contributions of the populations B and C in the same way. In these experiments, we hold the firing rate $r_A = 25$ constant. For a neuron $C \to_1 B \to_1 A$, whose segments are sequentially chained together, a spike is generated if and only if both C and B are activated, and in the correct order. The resulting contour-plot, which shows how the output firing rate of this neuron scales with both r_C and r_B, illustrates that both a high firing rate of population C and B are required to result in a high firing rate of the neuron (**Fig. 4d**). This is similar to the neuron with two parallel segments $(C + B) \to_2 A$ (**Fig. 4e**), only that simultaneous activation of both segments, not sequential activation, is required. The shape of this function closely matches an idealized "and" operation (**Fig. 4b**), the firing rate of which can be derived as just the product of the rates at which plateaus are triggered in all dendritic segments:

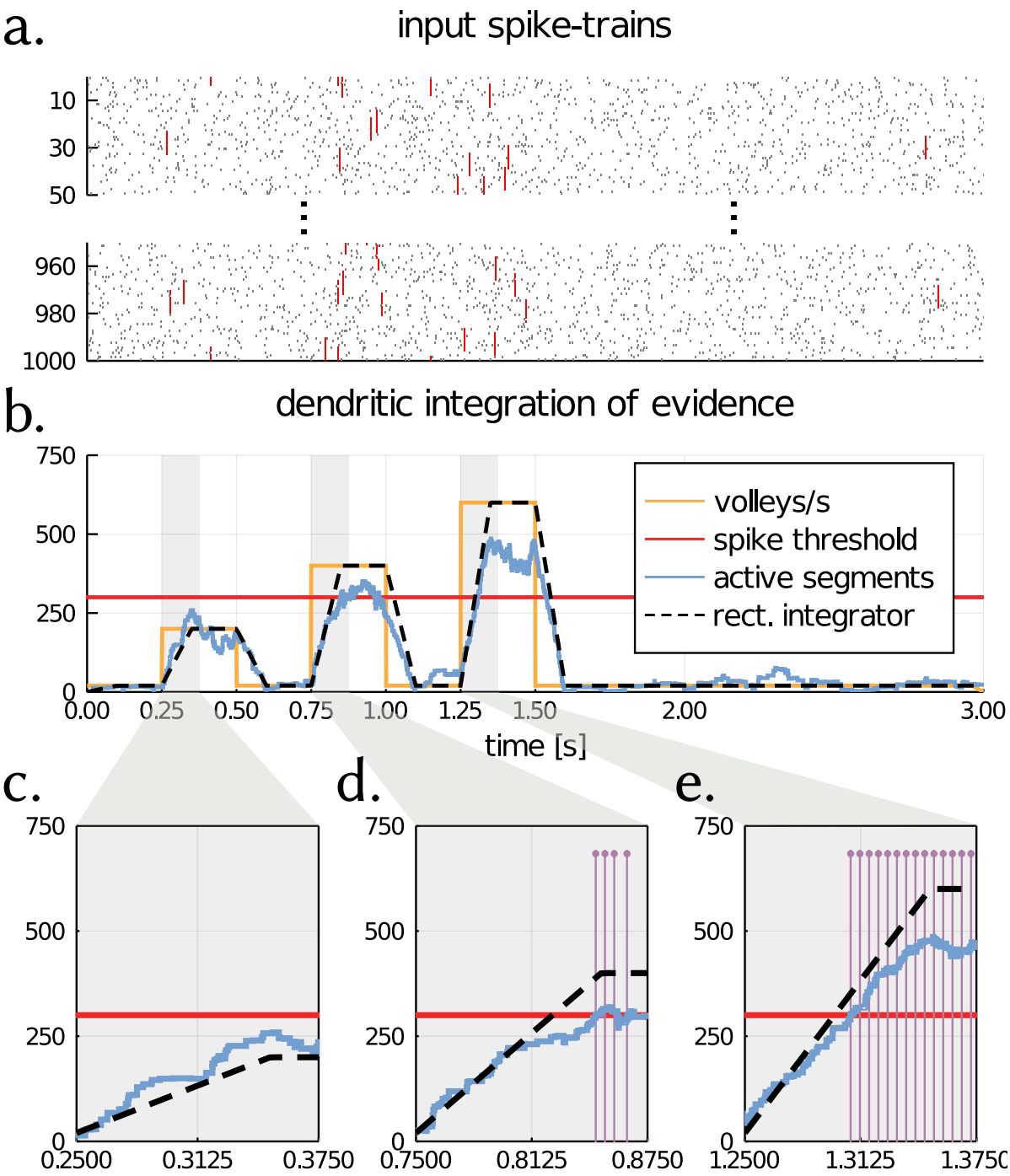


Figure 3: Dendritic plateaus can be used to gradually integrate evidence over long time periods. **a.** A neuron with 1000 dendritic segments is driven by 1000 incoming spike-trains. Embedded in these spike-trains are spike volleys of 10 coincident spikes each, spread across 10 neighbouring neurons (shown in red). **b.** The rate of spike volleys is determined by an input signal (organge line). Each segment receives input from 20 consecutive neurons through stochastic synapses with transmission probability $p = 0.5$, and requires 5 coincident spikes to trigger a plateau potential. The total number of co-activated dendritic segments (blue line) follows the convolution of the stimulus signal with a rectangular filter of length 100ms (black dashed line). **c-e.** For increasing levels of stimulation, the number of co-activated segments rises faster and saturates at a higher level, crossing the threshold required for spike initiation (horizontal red line) at an earlier point in time or not at all, resulting in a sequence of spikes (vertical purple lines).

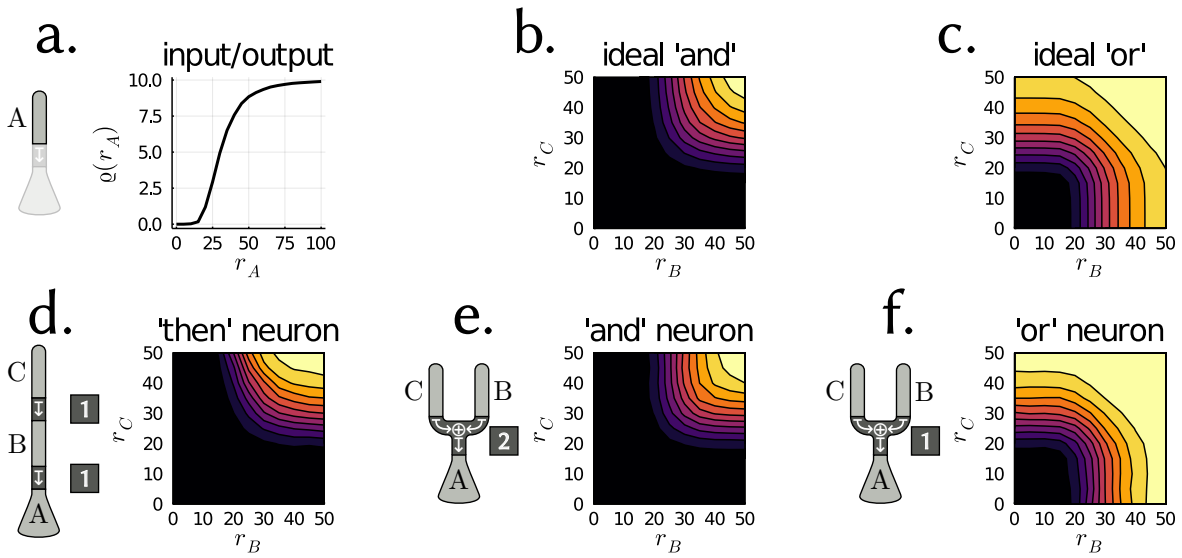


Figure 4: A rate-based analysis reveals well-known computational primitives. **a.** A single dendritic compartment that receives independent Poisson-spike trains at a fixed rate r_A from a population of 25 neurons responds with plateaus at a rate that can be expressed as a non-linear sigmoidal function $\varrho(r_A)$. For multiple dendritic segments, each of which receives input from an identical but independent population A,B or C, the neuron's computation depends on the dendritic morphology. **d and e.** If both segments C and B are required to enable a somatic spike, the neuron's firing rate is proportional an **b** idealized "and" operation between the two inputs. **f.** If either of the two segments suffices, **c** the firing rate instead resembles an idealized "or" operation.

$$\varrho(A,B,C) \propto \tau_{\text{dendrite}}\varrho(r_A) f_{\text{and}}(B,C) \quad \text{where} \quad f_{\text{and}}(B,C) = \tau^2_{\text{dendrite}}\varrho(r_C)\varrho(r_B)$$

Here, $\varrho(A,B,C)$ is the firing rate of the neuron, and $f_{\text{and}}(B,C)$ is the factor due to the segments B and C.

For a different dendritic morphology $(C+B) \to_1 A$, where a plateau in either segment C or B is sufficient (**Fig. 4f**), we see a response that closely resembles an idealized "or" operation (**Fig. 4c**):[3]

$$f_{\text{or}}(B,C) \propto \tau_{\text{dendrite}}\varrho(C) + \tau_{\text{dendrite}}\varrho(B) - f_{\text{and}}(B,C)$$

For a derivation of f_{and} and f_{or} see the Methods section. This rate-based functional description offers a very useful abstraction of the neurons' behaviours, but it necessarily neglects questions of timing. As we saw in the previous sections, depending on the morphology, a dendritic arbor can impose stringent requirements on the order in which different segments can be activated. For example, while both neurons $C \to_1 B \to_1 A$ and $(C+B) \to_2 A$ require strong input from both input population B and C and hence show the same "and"-like response in the rate-coding paradigm, the former imposes the constraint that the input from population C must arrive *before* that from population B while the latter does not. Rather than an "and"-like operation, neuron $C \to_1 B \to_1 A$ in fact implemented an "and then" operation. This is apparent when looking at the joint probability density of the relative timing of dendritic plateaus in the respective segments directly preceding a somatic spike (**Fig. 5**). In particular, if we only consider the unambiguous cases of one dendritic plateau each occurring in each segment within a brief window before a somatic spike (shown by the white dots (**Fig. 5**)), we observe that for neuron $C \to_1 B \to_1 A$, a dendritic plateau in segment B can occur at most 100ms before the somatic spike and is preceded by a dendritic plateau in segment C by at most another 100ms for a maximum total delay of 200ms. In contrast for neuron $(C+B) \to_2 A$, both segments must trigger a plateau within 100ms to elicit a somatic spike. For neuron $(C+B) \to_1 A$, a plateau in either segment within a 100ms window suffices to trigger a somatic spike.

[3] As the last equation shows, referring to this operation as an "or" is justified in the sense that the resulting rate is proportional to the addition of the segments' individual plateau-firing-rates minus the "and" operation applied to both, which generalizes the Boolean operation to real values.

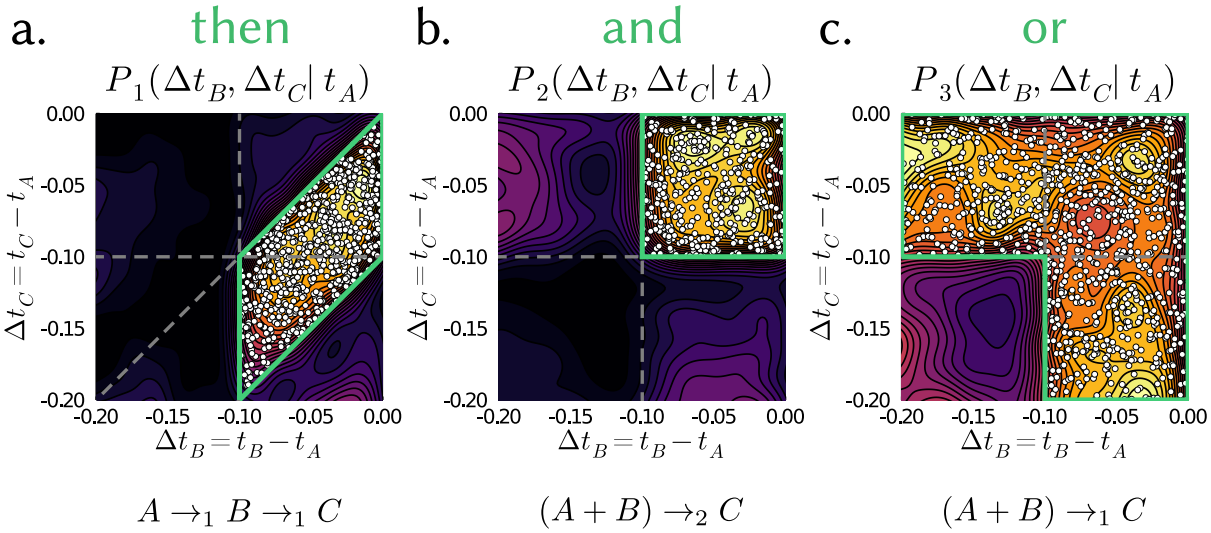


Figure 5: Dendritic morphology imposes timing constraints not revealed by rate-based analysis. For the neurons shown in figure 4, the joint probability distribution of relative timings Δt_B,Δt_C of dendritic plateaus directly preceding a somatic spike at t_A show a distinct temporal structure (contour-plots). **a.** For the "then" neuron, a plateau in segment C must precede a plateau in segment B by at most 100ms, which in turn must occur at most 100ms before a somatic spike can be triggered. This is evident by the fact that all unambiguous cases, where exactly one plateau in each segment C and B was observed before a somatic spike, fall into the corresponding parallelogram-shaped domain (white dots). **b.** The "and" neuron shows a similar rate-response to the "then" neuron, but requires both inputs to occur within 100ms before the somatic spike. **c.** The "or" neuron only requires either of the populations B or C to trigger a plateau within 100ms before a somatic spike.

Discussion

In this theoretical study we showed how a well-known biological phenomenon, dendritic plateau potentials, can drastically improve the computational capabilities of spiking neurons, turning them into powerful spatio-temporal pattern detectors. Due to the long-lasting memory provided by these plateau potentials, it becomes possible for individual neurons to integrate evidence or distinguish specific sequences of input on a timescale of hundreds of milliseconds – an order of magnitude larger than commonly observed membrane time constants [42]. In our model, the morphology of a neural dendrite determines its computational function and, when viewed in a conventional rate-coding paradigm, allows an individual neuron to implement a wide range of nonlinear behaviours in a modular and intuitive way.

This is in line with the two-layer neuron model proposed in [43], which used a detailed biophysical simulation of a pyramidal neuron to investigate the nonlinear effect on the neuron's firing rate due to synaptic input at different dendritic branches. Using a diverse array of stimuli, they showed that a two-layer network of sigmoidal subunits provides a substantially better approximation of the neuron's firing rate than a linear point-neuron. They speculated, however, that the prediction could be improved further, if the nonlinear interactions between the branches were considered, which we did here. We also investigated the use of dendritic plateau potentials as long-lasting memory traces, which our results revealed to be particularly important for evidence integration and the detection of temporal sequences. Remarkably, our drastically simplified and inherently event-based model could qualitatively reproduce properties of the model in [43], such as the sigmoidal input-output firing rate response of each dendritic segment and the linear-nonlinear combination thereof at the soma (see methods section).

But on the fast time-scale of individual spikes, our model differs substantially from this and other rate-based point-neuron models, since it relies on the detection of volleys of coincident spikes on a millisecond time-scale as the basic units of information, which are then integrated on the slower time-scale of dendritic plateau potentials. Our model is more closely related to recent work by [44], which proposed the use of active coincidence detection in dendritic segments to model prolonged effects of basal dendrites on the soma. A similar line of reasoning can also be found in [45], which presented a very elegant two-compartment neuron model and corresponding learning rule with one somatic and one dendritic compartment. Both models assign a specific functional role to the (basal) dendrite segments, namely to predict subsequent activation at the soma from their local synaptic inputs, which allows individual neurons to learn to predict state-transitions ("prospective coding"). Longer sequences are then detected by networks of such

laterally connected neurons, endowing the networks with a form of temporal sequence-memory ("hierarchical temporal memory"). In our work, we have focused on a more mechanistic model that heavily relies on biological phenomena observed in single neurons. This allowed us to describe a neuron's computational capability concretely as that of a sophisticated pattern detector with long-lasting memory, and to illustrate how these mechanisms at play would appear under a rate based analysis. We believe our results offer a very appealing explanation of spike-based computation that has wider implications in neuroscience and raises several important questions, which we briefly discuss in the following:

What is the role of inhibition for dendritic computation?

Our model only takes into account excitatory synapses, but has clear implications for the role of inhibition. The all-or-none response of dendritic plateau potentials in our model implies that the only significant effect an inhibitory synapse can have on the far-away soma is by either reducing the likelihood of plateaus, preventing the generation of plateaus altogether, or by disrupting already ongoing plateau potentials. In the first two cases, an inhibitory synapse's post-synaptic potential must be either well-timed to coincide with the volley of excitatory spikes or exhibit a longer time-scale. Experiments suggest that inhibition can affect the ability of dendrites to generate active plateaus and prevent them [46]. The disruption of ongoing plateaus has also been reported and analyzed [47] and requires no such precise timing a-priori, as long as the spike occurs within the plateau's duration. Inhibition may, however, exhibit different effects depending on when during the plateau processes it is received. In all cases, the likely effect is shunting, rather than substractive, inhibition.

Shunting inhibition can provide an efficient mechanism to improve the computational capabilities of the neurons described above, for example as it would allow individual neurons to exclusively respond to a sequence $a \rightarrow b$ but not to the sequence $a \rightarrow b \rightarrow c$, which is impossible for a neuron with purely excitatory synapses. Inhibition may therefore play an important and distinct role in ADSP neuron that warrants further investigation.

What are the implications of this model for plasticity?

We discussed a fundamental mechanism of dendritic computation and its capabilities, but did not cover the important topic of learning and plasticity. Nevertheless, the model presented here imposes constraints on potential plasticity mechanisms. Due to the long-lasting plateau potentials, a synaptic input can have a relevant causal effect for a somatic spike at a much later time. This makes the temporal assignment of credit for spiking outputs to synaptic inputs fundamentally difficult. The timing-invariance shown by our model and the dependency on the complex nonlinear dynamics within a dendritic tree further exacerbate this problem.

The most prominent example of synaptic learning is spike-time dependent plasticity [48], which tunes synaptic efficacy based on the relative timing of pre- and post-synaptic activity. Since the active dendritic processes discussed here both dominate the post-synaptic membrane potential as well as local Ca^{2+} concentration, they have a major effect on Hebbian plasticity [49, 50].

This is at odds with the common assumption, that backpropagating action potentials (bAPs) from the soma into the dendrite act as the primary post synaptic signal driving synaptic plasticity [51]. Since dendritic plateau potentials strongly depolarize dendrite segments for an extended period of time and should similarly "backpropagate" throughout the dendritic tree, it seems unlikely to us that bAPs are the primary factor for synaptic plasticity in neurons with active dendritic processes. Resolving this inconsistency is an important, but open research question.

Additionally, our model is based on binary stochastic synapses, and which segment the synapse terminates on plays a more important role than its efficacy. We therefore believe that structural plasticity mechanisms are particularly relevant for this kind of model. Furthermore, homeostatic plasticity mechanisms, e.g. scaling synaptic transmission probabilities[52], are in our view important to ensure that only sufficiently large spike-volleys, but not randomly correlated inputs, can reliably trigger plateau potentials.

Is neuronal computation based on plateau processes?

Dendritic processes are thought to implement solutions to a number of specific computational problems in neurons [53], often distributed across many functional dendritic compartments [54, 55]. Based on convincing biological evidence for the mechanism of plateau generation and the interaction of such plateaus, we have argued that they are indeed the primary building block for the implementation of behaviorally highly relevant computations. How can this claim be experimentally verified or falsified?

Direct experimental verification, that computation in single neurons is well described by our proposed ADSP neuron model requires simultaneous measurement of synaptic inputs and local membrane potentials along a single neuron's dendrite on a fine temporal and spatial resolution over a long-time span.

As a first step, since our model is driven by incoming spike volleys from multiple intact neuron populations, *in vivo* measurements could verify the existence of patterns of spike-volleys over different timescales using newly developed

statistical techniques [56, 57].
Secondly, a key part of the model, the detection and integration of information across two timescales, one on the order of a few milliseconds, the other on the order of a hundred milliseconds or more, can be refuted for any type of neuron that achieves this without reliance on active dendritic processes. This may be the case either for neurons incapable of generating plateaus in the first place, or if plateau-generating processes have been pharmacologically disabled.
Thirdly, we predict single neurons that use active dendritic sequence processing to have spatio-temporal receptive fields on long temporal timescales, but with high tolerance to variations in the precise timing of individual plateaus, qualitatively described in **Fig. 2**. Because of this invariance, we propose to go beyond linear analysis such as spike-triggered averages and instead measure both somatic response, as well as the timing of plateaus across the dendritic tree to find structures in the joint distributions as demonstrated in **Fig. 5**. Experimentally, spatio-temporal receptive fields of this kind could also be found by systematically varying stimuli, and should disappear when plateau-generating processes are disrupted.
While we have based our analysis on NMDAr-mediated plateaus in pyramidal cells [20], the same computational principle may be found in other neuron types, as well. For example, Purkinje cells in the cerebellum also generate localized Ca^{2+} events in response to coincident input on individual dendritic segments [58, 59], and thalamo-cortical neurons respond to strong synaptic input by localized plateaus in distal dendritic branches [60]. This indicates that the underlying ADSP mechanism, possibly implemented through diverse means in a case of convergent evolution, may be very general and ubiquitous in the brain.

In summary, we have presented and analyzed an intentionally simple model of neural computation based solely on the interaction of coincident spikes and dendritic plateau potentials. This revealed, how the morphology of the dendritic tree can implement and compose a wide range of non-linear computational functions. Two key features of this computational mechanism are its invariance to exact timings of inputs and its ability to operate on timescales much longer than post-synaptic potentials. We have highlighted the importance of the combination of these two features in two behaviorally relevant tasks: the detection of sequences and the integration of weak signals on long timescales. However, when analyzed from the usual perspective of rate-coding, these computational properties are hard to detect. To this end, we have therefore an alternative set of analyses to identify whether computation in single neurons is indeed based on dendritic plateaus.

Methods

Formal description of the event-based framework for computation in active dendrites

Mathematically, we approximate both EPSPs and plateau potentials by rectangular pulses with fixed duration τ_{synapse} and τ_{dendrite}, respectively. Here, we chose $\tau_{\text{synapse}} = 5$ms and $\tau_{\text{dendrite}} = 100$ms for all experiments if not stated otherwise. The dynamics of each dendritic segment can then be fully described in terms of the arrival times of incoming spikes as well as the times at which plateau potentials are initiated within the segment itself or in other directly connected segments. For some segment i, the synaptic input X_i and the dendritic input Y_i take the form of equations (1) and (2), respectively:

$$X_i(t) = \sum_{j \in S_i} \sum_k \chi_{i,j,k} \cdot \mathbf{1}_{[s_k^j,\, s_k^j + \tau_{\text{synapse}}]}(t) \quad \text{where } \chi_{i,j,k} \sim \text{Bernoulli}(\omega_{i,j}) \tag{1}$$

$$Y_i(t) = \sum_{j \in D_i} \sum_k \mathbf{1}_{[t_k^j,\, t_k^j + \tau_{\text{dendrite}}]}(t) \tag{2}$$

$$t_{m+1}^i = \min \left\{ t \in \mathbb{R} \;\middle|\; t \geq t_m^i + \tau_{\text{dendrite}}, X_i(t) \geq \theta_i^{\text{syn}} \text{ and } Y_i(t) \geq \theta_i^{\text{den}} \right\}, \tag{3}$$

where $\mathbf{1}_{[a,b]}$ represents a unit pulse during the time interval $[a, b]$, and s_k^j and t_j^i are the times of spikes arriving from some presynaptic neuron j and the plateau onset times on segment i, respectively. The random variable $\chi_{i,j,k}$ represents the independent probabilistic transmission of every spike k from source j via a synapse to dendritic segment i, where the transmission occurs with the synapse specific probability $\omega_{i,j}$. The sets S_i and D_i respectively identify the segment's synaptic connections to other neurons and which other dendritic segments it is directly coupled to, and therefore reflect the morphology of the neuron's dendritic tree. Equation (3) states that, if the segment is not in a plateau state already, a new plateau is initiated as soon as both synaptic and dendritic inputs exceed their respective thresholds θ_i^{syn} and θ_i^{den}.

Implementation of the navigation experiments

To simulate the stochastic movements of a rat, random paths are generated with time-varying location $l(t) = (X(t), Y(t)) \in \mathbb{R}^2$ as solutions of the following system of stochastic differential equations:

$$\begin{aligned} dX &= \cos(2\pi A)V\,dt \\ dY &= \sin(2\pi A)V\,dt \\ dA &= 0.25dW_A \\ dV &= 10.0(0.25 - V)dt + 0.1dW_V \end{aligned}$$

A represents the angular heading of the animal, V represents its velocity in $\frac{m}{s}$ and W_A, W_V represent independent standard Brownian motion processes. Each path is generated with a randomized initial position within a rectangular domain of $10cm \times 9.5cm$, a random angular heading and a random velocity according to the marginal stationary distribution of V in the equation above, and is simulated for a fixed duration of $200ms$. Three populations of place cells, each 20 neurons strong, are centered on a hexagonal grid with center-to-center distance of $r \approx 2.9cm$. Each population randomly emits spike volleys following a homogeneous Poisson process with rate $\lambda = 50Hz$. The magnitude of each spike volley is determined by the population's mean activity at the time, which depends on the animal's location within the environment through a receptive field tuning curve. The tuning curves model the probability of each individual neuron within the population to participate in a given spike volley by the bell-curves $f_i(x) = \exp(-\frac{x-\mu_i}{2\sigma^2})$ with coefficient $\sigma = 9.7mm$, centered on the tiles of the hexagonal grid. The total number of spikes emitted during a volley from population i at time t is therefore a random variable distributed according to a Binomial distribution with population size $n = 20$ and probability $p = f_i(l(t))$. Additionally, each neuron in the population emits random spikes at a rate of $5Hz$ to emulate background activity. Each spike is transmitted through stochastic synapses independently with probability 0.5.

Each of the simulated neuron's dendritic segments receives spiking input from the 20 neurons of one population and requires at least 5 coincident spikes to trigger a plateau potential. The three segments are connected in a chain that requires sequential activation by spike volleys from the input populations in correct order to fire a spike. A random path is considered to be accepted by the neuron, if the neuron responds with a spike at any point in time during the corresponding simulation run.

To evaluate the rotation and location sensitivity of the neuron, we also generate straight paths with constant movement speed $v = \frac{3r}{200\text{ms}} \approx 43^{\text{cm}}/_s$ that are either rotated around the center of the environment by an angle α or offset from the center by a distance Δx orthogonal to the optimal movement direction. For each angle or offset, respectively, the empirical firing probability of the neuron in response to that path is estimated by simulating the path and the neuron's responses 500 times each.

Implementation of the evidence-integration experiments

The input to the evidence-integrating neuron is generated by superimposing spike volleys onto 1000 independent Poisson processes with a constant firing rate of 10Hz. The volleys times are generated by a Poisson process with a time-varying rate $\lambda(t)$ representing the incoming "evidence". Here, $\lambda(t) = 200\text{Hz} \cdot (\mathbf{1}_{[0.25,0.5]}(t) + 2 \cdot \mathbf{1}_{[0.75,1.0]}(t) + 3 \cdot \mathbf{1}_{[1.25,1.5]}(t)) + 20\text{Hz}$. Each volley consists of simultaneous spikes from a randomly chosen set of ten input neurons with consecutive indices (wrapping around from 1000 to 1). Since each EPSP is assumed to last for a duration of 5ms, volleys and individual spikes are discarded if they occur less than 5ms after a preceding volley or spike. Each of the neuron's 1000 dendritic segments receives synaptic input via stochastic synapses with transmission probability 0.5 from 20 consecutive input neurons. As the number of input neurons and dendritic segments matches in this example, there is exactly one dendritic segment for every group of 20 consecutive input neurons, and each input neuron projects to exactly 20 dendritic segments. The total number of the neuron's synapses in this example is therefore 20000. Over time, the number of simultaneously active dendritic compartments as well as the times of generated somatic spikes is recorded. As a reference, the convolution $(\lambda \star \Pi)(t)$ of the time-varying rate-function λ with a rectangular filter Π of length 100ms and unit-integral is calculated.

Implementation of the rate-based analysis

For the rate-based analysis, four different neurons are constructed. First, a neuron consisting of a single dendritic compartment is driven by a total of 25 independent Poisson spike-trains with constant firing rate r_A. As in all

other experiments, the duration of each spike is set to $\tau_{synapse} = 5$ms, the duration of a plateau potential is set to $\tau_{dendrite} = 100$ms. By systematically varying r_A and, for each choice, recording the number of plateau potentials generated during a simulation time-interval of 250s we can estimate the smooth function $\varrho(r_A)$, which relates the firing rate of the input population A to the resulting rate at which plateau potentials are generated.

For each of the three morphologies representing the $C \to_1 B \to_1 A$ neuron, the $(C + B) \to_2 A$ neuron and the $(C + B) \to_1 A$ neuron, we systematically vary the input firing rates of both populations B and C independently while keeping the firing rate of population A fixed at a constant 25Hz. For each combination, we again record the number of somatic spikes generated over a time-interval of 250s. As a reference for these two-dimensional functions, we use an idealized "and" and "or" function defined as:

$$f_{\text{and}}(B, C) = \tau^2_{\text{dendrite}} \varrho(r_C)\varrho(r_B) \tag{4}$$

$$f_{\text{or}}(B, C) = \tau_{\text{dendrite}} \varrho(C) + \tau_{\text{dendrite}} \varrho(B) - f_{\text{and}}(B, C) \tag{5}$$

$$= 1 - (1 - \tau_{\text{dendrite}} \varrho(C))(1 - \tau_{\text{dendrite}} \varrho(B)) \tag{6}$$

At a firing rate r_X, a segment driven by population X is in a plateau state at a given point in time with probability $\tau_{\text{dendrite}} \varrho(r_X)$, therefore the probability that a segment driven by population C is active at the time that an input from population B arrives, which could in turn activate the next segment, is $\tau_{\text{dendrite}} \varrho(r_C)$. The probability that this second segment is still active, when yet another volley from population A arrives to possibly trigger a somatic spike is also $\tau_{\text{dendrite}} \varrho(r_B)$. Therefore the neuron's firing rate is proportional to $\tau^2_{\text{dendrite}} \varrho(r_C)\varrho(r_B)$. Similarly, the probability that two parallel upstream segments driven by populations C and B are simultaneously active at a given point in time is $\tau^2_{\text{dendrite}} \varrho(r_C)\varrho(r_B)$. In contrast, the probability that either upstream segment is active at a given point in time is just the probability that not both are simultaneously inactive, i.e. $1 - (1 - \tau_{\text{dendrite}} \varrho(C))(1 - \tau_{\text{dendrite}} \varrho(B))$. This expression has the nice alternative form $c + b - cb$, where $c = \tau_{\text{dendrite}} \varrho(C)$, $b = \tau_{\text{dendrite}} \varrho(B)$ and $cb = f_{\text{and}}(B, C)$, which generalizes the Boolean "or" operation to real-valued firing rates. When identifying $true$ with 1 and $false$ with 0, the truth-table of this expressions matches that of the logic expression "c or b".

To evaluate timing requirements for each of these three neuron morphologies, we run another simulation at constant input rates $r_A = r_B = r_C = 25$Hz for a duration of 1h of simulated time. We record the time of each plateau-initiation-event in both upstream segments driven by population C and B for a time-interval of 200ms preceding each somatic spike. If there is exactly one plateau-event from each segment in such a time-interval, we record this as an *unambiguous* pair of plateau events. If there is more than one plateau-event on either of the dendritic segments, we record all pairs of plateau-events in that time-interval composed of one plateau event for each segment. We refer to these latter pairs as *ambiguous*. Using these ambiguous pairs, we estimate the joint probability distribution $P_i(\Delta t_B, \Delta t_C | t_A)$ over relative times Δt_B and Δt_C between a plateau triggered by population B or C and a somatic spike triggered at time t_A by population A. For a more reliable estimate of the timing constraints, we consider only the unambiguous pairs, which evidently fall into distinct domains of these joint probability distributions that uniquely characterize the precise timing requirements of the respective neuron morphologies. This can be seen in figure 5. E.g. for the $C \to_1 B \to_1 A$ neuron, all plateaus triggered by population C must precede those triggered by B, but cannot precede them by more than one plateau duration of 100ms, therefore they fall into a parallelogram below the diagonal. For the $(C + B) \to_2 A$ neuron, on the other hand, both plateau events must independently occur within 100ms before a somatic spike, and hence fall into the upper quadrant of the joint density.

Code availability

All simulations are implemented in a custom developed package in the Julia programming language [61], publicly available via the code repository hosted at https://github.com/jleugeri/ADSP.jl. Further documentation of the simulator and implementation details can be found there.

References

[1] J O'Keefe and J Dostrovsky. The hippocampus as a spatial map. preliminary evidence from unit activity in the freely-moving rat. *Brain Res.*, 34(1):171–175, November 1971.

[2] Torkel Hafting, Marianne Fyhn, Sturla Molden, May-Britt Moser, and Edvard I Moser. Microstructure of a spatial map in the entorhinal cortex. *Nature*, 436(7052):801–806, August 2005.

[3] Martin Stemmler, Alexander Mathis, and Andreas V M Herz. Connecting multiple spatial scales to decode the population activity of grid cells. *Sci Adv*, 1(11):e1500816, December 2015.

[4] Howard Eichenbaum. On the integration of space, time, and memory. *Neuron*, 95(5):1007–1018, August 2017.

[5] Brice Bathellier, Derek L Buhl, Riccardo Accolla, and Alan Carleton. Dynamic ensemble odor coding in the mammalian olfactory bulb: sensory information at different timescales. *Neuron*, 57(4):586–598, February 2008.

[6] Bede M Broome, Vivek Jayaraman, and Gilles Laurent. Encoding and decoding of overlapping odor sequences. *Neuron*, 51(4):467–482, August 2006.

[7] Huan Luo and David Poeppel. Phase patterns of neuronal responses reliably discriminate speech in human auditory cortex. *Neuron*, 54(6):1001–1010, June 2007.

[8] Ofer Melamed, Wulfram Gerstner, Wolfgang Maass, Misha Tsodyks, and Henry Markram. Coding and learning of behavioral sequences. *Trends Neurosci.*, 27(1):11–4; discussion 14–5, January 2004.

[9] Gianluigi Mongillo, Omri Barak, and Misha Tsodyks. Synaptic theory of working memory. *Science*, 319(5869):1543–1546, March 2008.

[10] Yulia Sandamirskaya and Gregor Schöner. An embodied account of serial order: how instabilities drive sequence generation. *Neural Netw.*, 23(10):1164–1179, December 2010.

[11] C Beaulieu and M Colonnier. A laminar analysis of the number of round-asymmetrical and flat-symmetrical synapses on spines, dendritic trunks, and cell bodies in area 17 of the cat. *J. Comp. Neurol.*, 231(2):180–189, January 1985.

[12] Katz. *The Release of Neural Transmitter Substances (The Sherrington Lectures)*. Liverpool University Press, December 1969.

[13] C F Stevens. Quantal release of neurotransmitter and long-term potentiation. *Cell*, 72 Suppl:55–63, January 1993.

[14] Michael Hollmann and Stephen Heinemann. Cloned glutamate receptors. *Annual review of neuroscience*, November 2003.

[15] JC Watkins and RH Evans. Excitatory amino acid transmitters. *Annual review of pharmacology and toxicology*, 21(1):165–204, 1981.

[16] Attila Losonczy and Jeffrey C Magee. Integrative properties of radial oblique dendrites in hippocampal CA1 pyramidal neurons. *Neuron*, 50(2):291–307, April 2006.

[17] W Rall. Electrophysiology of a dendritic neuron model. *Biophys. J.*, 2(2 Pt 2):145–167, March 1962.

[18] A N Burkitt. A review of the integrate-and-fire neuron model: I. homogeneous synaptic input. *Biol. Cybern.*, 95(1):1–19, July 2006.

[19] Greg Stuart and Nelson Spruston. Determinants of voltage attenuation in neocortical pyramidal neuron dendrites. *Journal of Neuroscience*, 18(10):3501–3510, 1998.

[20] Nelson Spruston. Pyramidal neurons: dendritic structure and synaptic integration. *Nat. Rev. Neurosci.*, 9(3):206–221, March 2008.

[21] Michael Häusser. Synaptic function: dendritic democracy. *Current Biology*, 11(1):R10–R12, 2001.

[22] Jeffrey C Magee and Erik P Cook. Somatic epsp amplitude is independent of synapse location in hippocampal pyramidal neurons. *Nature neuroscience*, 3(9):895–903, 2000.

[23] Srdjan D Antic, Wen-Liang Zhou, Anna R Moore, Shaina M Short, and Katerina D Ikonomu. The decade of the dendritic NMDA spike. *J. Neurosci. Res.*, 88(14):2991–3001, November 2010.

[24] Katerina D Oikonomou, Mandakini B Singh, Enas V Sterjanaj, and Srdjan D Antic. Spiny neurons of amygdala, striatum, and cortex use dendritic plateau potentials to detect network UP states. *Front. Cell. Neurosci.*, 8:292, September 2014.

[25] H Monyer, N Burnashev, D J Laurie, B Sakmann, and P H Seeburg. Developmental and regional expression in the rat brain and functional properties of four NMDA receptors. *Neuron*, 12(3):529–540, March 1994.

[26] T Götz, U Kraushaar, J Geiger, J Lübke, T Berger, and P Jonas. Functional properties of AMPA and NMDA receptors expressed in identified types of basal ganglia neurons. *J. Neurosci.*, 17(1):204–215, January 1997.

[27] Sonia Gasparini, Michele Migliore, and Jeffrey C Magee. On the initiation and propagation of dendritic spikes in CA1 pyramidal neurons. *J. Neurosci.*, 24(49):11046–11056, December 2004.

[28] Sonia Gasparini and Jeffrey C Magee. State-dependent dendritic computation in hippocampal CA1 pyramidal neurons. *J. Neurosci.*, 26(7):2088–2100, February 2006.

[29] Jacopo Bono and Claudia Clopath. Modeling somatic and dendritic spike mediated plasticity at the single neuron and network level. *Nat. Commun.*, 8(1):706, September 2017.

[30] Paul Rhodes. The properties and implications of NMDA spikes in neocortical pyramidal cells. *J. Neurosci.*, 26(25):6704–6715, June 2006.

[31] Guy Major, Matthew E Larkum, and Jackie Schiller. Active properties of neocortical pyramidal neuron dendrites. *Annu. Rev. Neurosci.*, 36:1–24, July 2013.

[32] Tim Jarsky, Alex Roxin, William L Kath, and Nelson Spruston. Conditional dendritic spike propagation following distal synaptic activation of hippocampal CA1 pyramidal neurons. *Nat. Neurosci.*, 8(12):1667–1676, December 2005.

[33] Christine Grienberger, Xiaowei Chen, and Arthur Konnerth. Nmda receptor-dependent multidendrite ca2+ spikes required for hippocampal burst firing in vivo. *Neuron*, 81(6):1274–1281, 2014.

[34] C Koch, T Poggio, and V Torre. Retinal ganglion cells: a functional interpretation of dendritic morphology. *Philos. Trans. R. Soc. Lond. B Biol. Sci.*, 298(1090):227–263, July 1982.

[35] Alon Polsky, Bartlett W Mel, and Jackie Schiller. Computational subunits in thin dendrites of pyramidal cells. *Nat. Neurosci.*, 7(6):621–627, June 2004.

[36] Tiago Branco and Michael Häusser. The single dendritic branch as a fundamental functional unit in the nervous system. *Curr. Opin. Neurobiol.*, 20(4):494–502, August 2010.

[37] Guy Major, Alon Polsky, Winfried Denk, Jackie Schiller, and David W Tank. Spatiotemporally graded NMDA spike/plateau potentials in basal dendrites of neocortical pyramidal neurons. *J. Neurophysiol.*, 99(5):2584–2601, May 2008.

[38] Tiago Branco, Beverley A Clark, and Michael Häusser. Dendritic discrimination of temporal input sequences in cortical neurons. *Science*, 329(5999):1671–1675, 2010.

[39] Matthew E Larkum and Thomas Nevian. Synaptic clustering by dendritic signalling mechanisms. *Curr. Opin. Neurobiol.*, 18(3):321–331, June 2008.

[40] Naoya Takahashi, Kazuo Kitamura, Naoki Matsuo, Mark Mayford, Masanobu Kano, Norio Matsuki, and Yuji Ikegaya. Locally synchronized synaptic inputs. *Science*, 335(6066):353–356, January 2012.

[41] David R Euston, Masami Tatsuno, and Bruce L McNaughton. Fast-forward playback of recent memory sequences in prefrontal cortex during sleep. *Science*, 318(5853):1147–1150, November 2007.

[42] Debora Ledergerber and Matthew Evan Larkum. Properties of layer 6 pyramidal neuron apical dendrites. *Journal of Neuroscience*, 30(39):13031–13044, 2010.

[43] Panayiota Poirazi, Terrence Brannon, and Bartlett W Mel. Pyramidal neuron as two-layer neural network. *Neuron*, 37(6):989–999, 2003.

[44] Jeff Hawkins and Subutai Ahmad. Why neurons have thousands of synapses, a theory of sequence memory in neocortex. *Frontiers in neural circuits*, 10:23, 2016.

[45] Johanni Brea, Alexisz Tamás Gaál, Robert Urbanczik, and Walter Senn. Prospective coding by spiking neurons. *PLOS Computational Biology*, 12(6):1–25, 06 2016.

[46] Monika Jadi, Alon Polsky, Jackie Schiller, and Bartlett W Mel. Location-dependent effects of inhibition on local spiking in pyramidal neuron dendrites. *PLoS Comput Biol*, 8(6):e1002550, 2012.

[47] Michael Doron, Giuseppe Chindemi, Eilif Muller, Henry Markram, and Idan Segev. Timed synaptic inhibition shapes nmda spikes, influencing local dendritic processing and global i/o properties of cortical neurons. *Cell reports*, 21(6):1550–1561, 2017.

[48] Sen Song, Kenneth D Miller, and Larry F Abbott. Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nature neuroscience*, 3(9):919–926, 2000.

[49] John Lisman and Nelson Spruston. Postsynaptic depolarization requirements for LTP and LTD: a critique of spike timing-dependent plasticity. *Nat. Neurosci.*, 8(7):839–841, July 2005.

[50] Jason Hardie and Nelson Spruston. Synaptic depolarization is more effective than back-propagating action potentials during induction of associative long-term potentiation in hippocampal pyramidal neurons. *J. Neurosci.*, 29(10):3233–3241, March 2009.

[51] Nicola Kuczewski, Cristophe Porcher, Volkmar Lessmann, Igor Medina, and Jean-Luc Gaiarsa. Back-propagating action potential. *Communicative & Integrative Biology*, 1(2):153–155, 2008. PMID: 19704877.

[52] Gina Turrigiano. Homeostatic synaptic plasticity: local and global mechanisms for stabilizing neuronal function. *Cold Spring Harbor perspectives in biology*, 4(1):a005736, 2012.

[53] Michael London and Michael Häusser. Dendritic computation. *Annu. Rev. Neurosci.*, 28:503–532, 2005.

[54] Aaron Kerlin, Mohar Boaz, Daniel Flickinger, Bryan J MacLennan, Matthew B Dean, Courtney Davis, Nelson Spruston, and Karel Svoboda. Functional clustering of dendritic activity during decision-making. *Elife*, 8:e46966, 2019.

[55] Hongbo Jia, Nathalie L Rochefort, Xiaowei Chen, and Arthur Konnerth. Dendritic organization of sensory input to cortical neurons in vivo. *Nature*, 464(7293):1307–1312, April 2010.

[56] Rory G Townsend and Pulin Gong. Detection and analysis of spatiotemporal patterns in brain activity. *PLoS Comput. Biol.*, 14(12):e1006643, December 2018.

[57] Min Song, Minseok Kang, Hyeonsu Lee, Yong Jeong, and Se-Bum Paik. Classification of spatiotemporal neural activity patterns in brain imaging data. *Sci. Rep.*, 8(1):8231, May 2018.

[58] Yunliang Zang, Stéphane Dieudonné, and Erik De Schutter. Voltage- and Branch-Specific climbing fiber responses in purkinje cells. *Cell Rep.*, 24(6):1536–1549, August 2018.

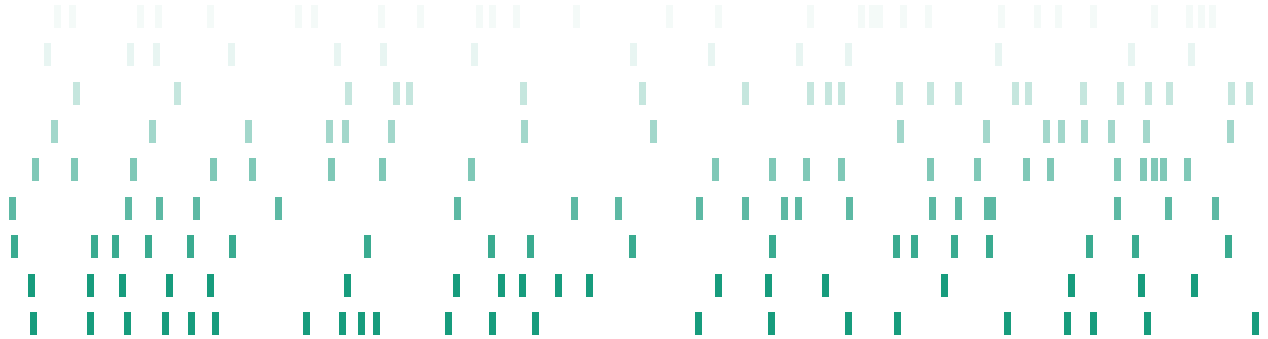
[59] CF Ekerot and O Oscarsson. Prolonged depolarization elicited in purkinje cell dendrites by climbing fibre impulses in the cat. *The Journal of physiology*, 318(1):207–221, 1981.

[60] Sigita Augustinaite, Bernd Kuhn, Paul Johannes Helm, and Paul Heggelund. NMDA spike/plateau potentials in dendrites of thalamocortical neurons. *J. Neurosci.*, 34(33):10892–10905, August 2014.

[61] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

Making spiking neurons more succinct with multi-compartment models

Johannes Leugering
johannes.leugering@iis.fraunhofer.de
Fraunhofer Institute for Integrated Circuits
Erlangen, Germany



ABSTRACT

Spiking neurons consume energy for each spike they emit. Reducing the firing rate of each neuron — without sacrificing relevant information content— is therefore a critical constraint for energy efficient networks of spiking neurons in biology and neuromorphic hardware alike. The inherent complexity of biological neurons provides a possible mechanism to realize a good trade-off between these two conflicting objectives: multi-compartment neuron models can become selective to highly specific input patterns, and thus learn to produce informative yet sparse spiking codes. In this paper, I motivate the operation of a simplistic hierarchical neuron model by analogy to decision trees, show how they can be optimized using a modified version of the greedy decision tree learning rule, and analyze the results for a simple illustrative binary classification problem.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**.

KEYWORDS

spiking neural networks, multi-compartment models, decision tree, random forest, information theory

NICE '20, March 17–20, 2020, Heidelberg, Germany

ACM ISBN 978-1-4503-7718-8/20/03…$15.00
https://doi.org/10.1145/3381755.3381763

ACM Reference Format:
Johannes Leugering. 2020. Making spiking neurons more succinct with multi-compartment models. In *Neuro-inspired Computational Elements Workshop (NICE '20), March 17–20, 2020, Heidelberg, Germany*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3381755.3381763

1 INTRODUCTION

Cortical neurons predominantly use spikes - brief nerve impulses of identical magnitude and shape - to convey information. This mode of communication has been largely ignored in machine learning models of neural networks in favour of *rate coding*, where each neuron produces a real valued quantity as its output - the neuron's *instantaneous firing rate*. In recent years however, driven in large part by the development of *neuromorphic hardware* and advances in training mechanisms[10], spiking neural networks (SNNs) have garnered renewed interest, since their binary pulse-based communication lends itself particularly well to implementation in conventional binary CMOS technology and promises to enable ultra-low-power applications while side-stepping many of the issues associated with purely analog or purely digital designs.

Since in SNNs the information exchange —as well as a substantial amount of energy expense— occurs with spikes, a *sparse code* that minimizes the total number of generated spikes can help to minimize the energy consumption. However, to ensure that the relevant information can still be transmitted even at very low spike-rates, it is critical that each spikes conveys *as much information as possible*. Since the magnitude of the spikes is assumed to be fixed, this leaves two possible approaches: The precise timing of individual spikes could be used to encode information (using a *spike-timing code*), but this approach is limited in practise by timing jitters introduced by thermal noise in analog implementations or by a limited system clock-resolution or the often unpredictable delays due to package

routing in digital implementations. The other approach is to ensure that each spike represents a highly *surprising* event (i.e. carry high information content) that is relevant for the task at hand. This appears to be nature's way, where spike trains convey information in a very effective way [12], so I'll go down the second road and discuss how simple multi-compartment models can pack a lot of information into a single spike.

Consider for example a binary classification task, where a network of spiking neurons should detect whenever its high-dimensional input falls into one specific of two classes. In order to make efficient use of each spike, it is not enough for the neuron to just encode its high-dimensional inputs faithfully. Rather, each neuron has to *actively discard irrelevant information* contained in its own input stimulus while retaining only the information that is relevant for producing the desired output. This challenge, also known as the *information bottleneck*, applies to individual neurons just as much as it does to layers of deep neural networks. Trivially, the best solution would be for the individual neuron to solve the classification task entirely, and just label each target stimulus with a spike — i.e. for the neuron to become a detector for the target class itself. This is practically infeasible for most interesting problems, but it illustrates the relationship between the computational power of the individual neuron and the information content of its output: the more computation can be performed within the neuron, the more informative its outputs can become.

In most commonly used models of artificial neural networks, a neuron's output is a nonlinear transformation of an affine linear operator applied to its inputs. Such neuron models are collectively referred to as linear-nonlinear point-neuron models, and can approximate the somatic response of biological neurons reasonably well. However they fail to account for many of the more recently observed nonlinear effects due the location of a synaptic connection along the dendritic tree, which give rise to highly complex interactions within the dendrite [6]. A mounting body of evidence points to the conclusion, that rather than a single linear operator, the dendritic arbor acts much more like a hierarchical structure of non-linear functionally distinct compartments — not unlike a tree-structured feed-forward neural network! While the experimental characterization of nonlinear dendritic processing *in vivo* is a fairly recent development, there is already empirical evidence to demonstrate that dendritic computation is capable of realizing boolean logic expressions beyond those that can be realized by passive dendrites alone [4, 9]. From this perspective, the atomic unit of computation is not a neuron, but rather a dendritic compartment, and the individual neuron itself is already a complex system designed to extract relevant information from its inputs. This perspective becomes even more intriguing when the temporal (recurrent) dynamics of dendritic processing and regulatory pathways are considered, as well, but for the sake of simplicity I will not discuss questions of timing in this paper.

In the following, I argue that simple multi-compartment models can provide a good trade-off between model complexity and information content of their outputs. To motivate this idea, I relate spiking multi-compartment neuron models to other established concepts from machine learning, namely decision trees and random forests, and show experimentally as well as analytically that such multi-compartment neurons can indeed help to increase the information content transmitted per spike, and thus reduce the required total number of spikes to be transmitted.

2 METHODS

To better understand how an individual multi-compartmental neuron can process information beyond what is possible in a linear-nonlinear model neuron, let's define a simplified mathematical model of spiking neurons with compartmentalized dendritic trees. In order to compare the computation realized by nonlinear dendritic processing to machine learning models such as decision trees, I make a couple of drastic simplifications: First, a *neuron* is composed of a tree-hierarchy of segments. A *segment* of the dendritic tree can have several local synaptic inputs and may recursively branch into several further subordinate child segments. Each segment can only be in either of two states, *active* or *inactive*, and the neuron fires a spike when its root segment becomes active. In order to become active, a segment must be activated sufficiently strongly through its *synaptic inputs*, which is modeled by an affine-linear combination of several input signals followed by a tresholding step-function. In addition, if the segment has any child segments, at least one of them has to be active as well. To simplify the discussion immensely, I treat time as progressing in discrete time-steps, such that each segment receives, within one time-step, a vector-valued input signal as well as its input from the upstream child segments and can either turn on or off in response. This model is a drastic simplification of a model derived from biological characteristics of cortical neurons, and ignores critical aspects timing, but it provides a good intuition for how multi-compartment structures can help to extract meaningful information and efficiently encode it in spikes. A much more in-depth discussion and motivation of the full model can be found in [8]. See figure 1 for a schematic representation of the model.

2.1 Neurons and decision trees

So what does it take to make a neuron model as described above spike? In order for the root segment to emit a spike, its local inputs need to be "satisfactory", as well as the input received from upstream child segments. Those segments in turn need to be activated, which recursively depends on their local synaptic inputs and children, and so on. As figure 1 a.) shows, there are several paths from a dendritic "leaf segment" to the somatic "root segment", and each of these paths represents one set of constraints that, when simultaneously satisfied by the input signal, will lead to a spike. Since each segment involved in such a path requires the previous child segment to be active, the constraints are evaluated by the neuron in a "lazy" fashion, starting from the leaf node continuing towards the soma until one of the constraints is violated, or a spike is fired. This perspective shows a very clear parallel to another well established tool from machine learning: In decision trees, starting from the root node, one constraint on the input signal is evaluated, and, depending on whether it is satisfied or not, the next node is chosen, where another constraint is evaluated, and so on[2]. If an "accepting" leaf node is reached, the input sample is accepted as belonging to the target class, otherwise it is rejected. See figure 1 b.). This similarly leads to a number of possible paths, this time originating at the root node and ending at "accepting" leaf nodes, each of which encodes

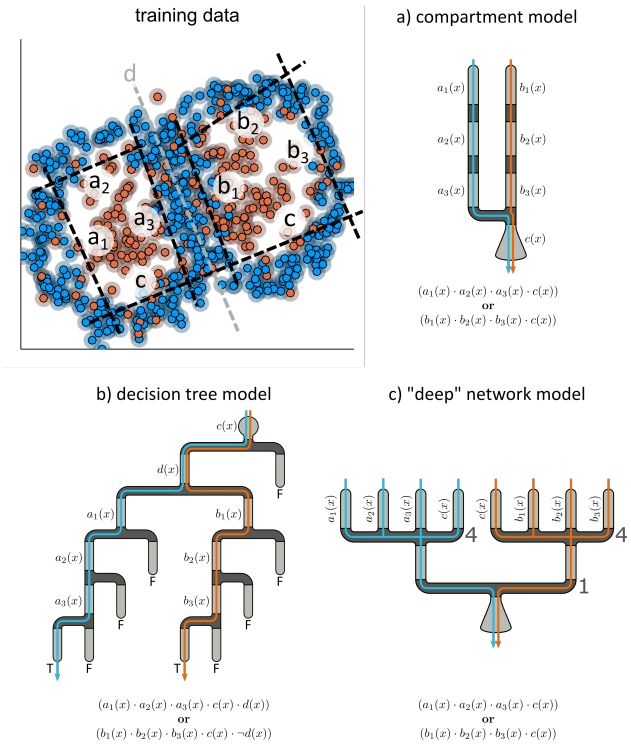


Figure 1: Comparison of different model types. a.) A multi-compartment neuron with two branches composed of three segments each. To fire, either of the two branches has to be fully activated. Each segment evaluates one linear classifier represented by the corresponding dotted line in the top-right panel. b.) and c.) A equivalent decision tree and feed-forward network that each select the same domain in the two-dimensional input space as in a.).

a different sets of constraints that, when simultaneously satisfied, would lead the tree to accept the input. Although the tree structure is inverted, the general concept of hierarchical neuron models and decision trees is therefore fully compatible.

Both of these approaches can represent arbitrary boolean functions of binary feature detectors (implemented by dendritic segments or decision nodes, respectively), since each path represents a conjunction of such features and the neuron or decision tree responds to a disjunction of multiple such paths, effectively representing any boolean expression of the features in its disjunctive normal form. One major difference between both representations is, that the result of a decision at a node of a decision tree merely influences to which of two nodes to evaluate next. Both possible answers are therefore equally informative, and only nodes along a single path are ever evaluated. In the neuron model, on the other hand, segments along all paths are activated simultaneously, but the propagation of dendritic action potentials along each path from a leaf to the root branch ends as soon as a single condition is not satisfied. While the decision tree algorithm uses the outcome of each decision most effectively, the neuron model reflects the un-clocked operation of a biological neuron, which receives its various inputs in parallel and no particular order. This latter approach thus fully utilizes the parallelism of its structure and therefore seems better suited for the design of custom hardware for parallel computing.

Using a sufficiently complex structure, both decision trees and multi-compartment neurons can therefore be configured to respond to very specific input patterns. Consider for example to top-left panel of figure 1, where an artificial two-dimensional input signal is to be classified into two classes, orange and blue. By carefully arranging 7 (or 8 for the decision tree) linear decision boundaries, both models can become selective to the conjunction of the two quadrilateral sets circumscribed by the decision boundaries, and could thus implement a decent solution for the classification problem! Figure 1 a.) and b.) show the corresponding model structures.

2.2 Growing neurons like trees

This perspective is very helpful, since a lot of the theoretical considerations underlying decision trees can be transferred to multi-compartment models of spiking neurons! Among them, the key insight behind the training algorithm for decision trees, and main reason for their success: the sequential greedy optimization algorithm[2]. To train a decision tree, the decision boundaries implemented by each node are optimized such that the result of the decision leads to the best possible expected reduction in uncertainty about the true class label *for exactly those input samples* that successfully propagate all the way up to the node. Each node thus refines the classification made by its ancestors. This concept, the greedy maximization of the so-called *information gain* along each path, can be directly transferred to multi-compartment neuron models as well, and for a supervised binary classification task leads to the following learning rule:

> Starting from terminal segments, a neuron's segments optimize their local synaptic input weights to a) ensure the maximum expected information content about the provided target labels for accepted and rejected inputs, and b.) ensure that the conditional probability of accepted inputs to belong to the target class is higher than that of rejected inputs.

Part a.) of this rule corresponds to the standard training algorithm for decision trees, and part b.) is a slight modification made necessary by the fact that in the multi-compartment neuron model, only the accepted inputs are propagated further to the unique parent branch, whereas in decision trees, each input propagates further to either of two child branches, depending on the classification result. Mathematically, this rule can be expressed as follows:

$$(w_i, b_i) = \text{argmax}_{w,b} \sum_c \sum_s p_{S,C}(s,c) \log\left(p_{C|S}(c|s)\right) \quad \text{where:} \tag{1}$$

$$p_{S|C}(s|c) = \frac{1}{|D_i^c|} \sum_{(x,l)\in D_i^c} f(wx - b) \tag{2}$$

$$p_C(c) = \frac{|D_i^c|}{\sum_\gamma |D_i^\gamma|} \tag{3}$$

$$P_{S,C}(s,c) = p_{S|C}(s|c)pC(c) \tag{4}$$

$$p_{C|S}(c|s) = \frac{p_{S,C}(s,c)}{\sum_\gamma p_{S,C}(s,\gamma)} \tag{5}$$

$$f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases} \text{ (during inference)} \quad \text{or} \tag{6}$$

$$f(x) = (1 + exp(-\alpha x))^{-1} \text{ (during training)} \tag{7}$$

Here D_i^c denotes the set of datapoints x_k with corresponding label $l_k = c$ that propagated all the way to segment i. The output of the function f represents the predicted probability of a datapoint to belong to the target class, and is therefore during inference calculated by a hard decision threshold, but can for training purposes be approximated instead by a differentiable function. For this relaxation, the information gain in equation 1 can be differentiated with respect to the parameter vectors w and b. The gradient information is used in a simple gradient descent algorithm to optimize the model coefficients in a greedy fashion for one dendritic segment at a time. The formalism can be applied recursively to each segment, starting from a terminal segment, and successively refines the subset of data that is propagated further.

2.3 A training example

Let's consider a simple example. The examples in this section are implemented as software simulations in the Julia programming language and are provided in a public code repository (see section 5). A single-compartment and a multi-compartment neuron each receive two-dimensional input signals just like in the top-left panel of figure 1 through their synaptic inputs. The single-compartment model implements a simple linear classifier and responds to those input signals that lie on the appropriate side of the decision boundary, whereas the multi-compartment model as illustrated by figure 1 a.) receives the same input on each segment, and responds only for those inputs that simultaneously satisfy several constraints imposed by the different segments. Training both models with the aforementioned greedy training algorithm provides the solutions that can be seen in the top row of figure 2. Impressively, the single multi-compartmental neuron in fact learned to become selective to the conjunction of either "eye" of the training data set. Results are shown for an independently drawn test dataset from the same distribution as the training dataset.

To verify that this is in fact a robust result, the bottom row of figure 2 shows the averaged classification results of 100 randomly initialized, structurally identical multi-compartment neurons, all trained on the same data. The results clearly show that only the desired regions within the two "eyes" of the dataset are selected for by

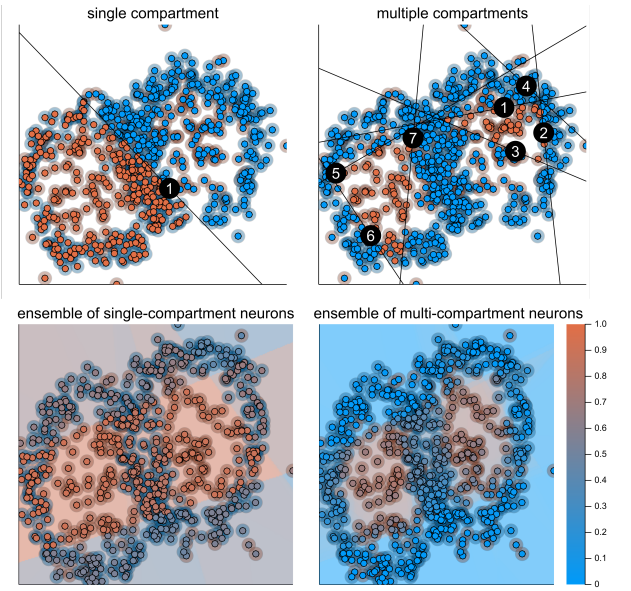


Figure 2: Comparison of trained single- and multi-compartment neurons. Top row: Individual neurons and their classification results are shown. Orange data points lead the respective neuron to spike, blue datapoints are rejected. Bottom row: ensembles of 100 randomly initialized single- and multi-compartment neurons, respectively. The color indicates the probability of a data-point to be selected by a randomly drawn neuron from the ensemble.

neurons (albeit not all of the neurons necessarily become selective to the conjunction of both — some instead only select for one of both "eyes"). For comparison, a similar ensemble of single-compartment models also preferentially selects for the correct region, however a significant number of neurons always also responds to undesirable surrounding region, and different neurons converge to almost identical solutions (this can be inferred from the few, pronounced corners visible in the color-map).

2.4 Untrained feature extraction

While the example given above shows that (greedy) supervised learning can be employed to turn multi-compartment neurons into sophisticated pattern detectors, they can be useful even in the absence of labeled training data. As figure 3 shows, randomly initialized single- and multi-compartment neurons are selective to qualitatively different subsets of the input space: whereas single-compartment models respond to an entire half-space, multi-compartment neurons instead tend to select bounded sub-volumes or cones of the input space. As a consequence of this, the expected response rate of multi-compartment models is much reduced, and any two randomly initialized neurons are more likely to represent two independent subsets of the input space, therefore complementing each other better as independent random feature detectors.

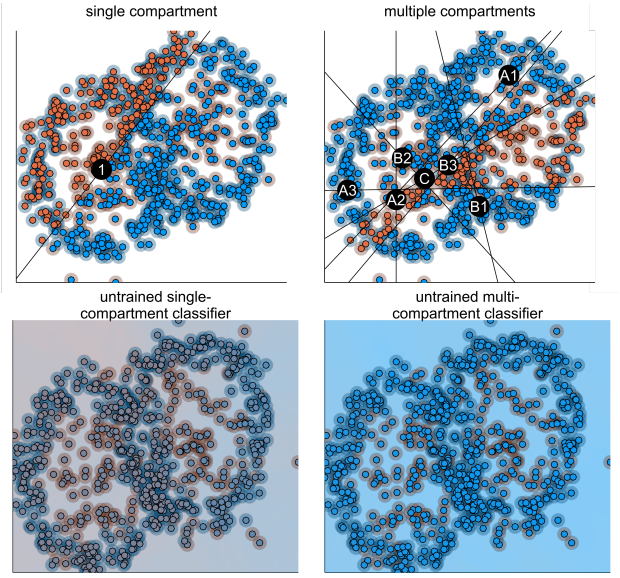


Figure 3: Comparison of untrained single- and multi-compartment neurons. Randomly initialized neurons and ensembles corresponding to those in figure 2 before training.

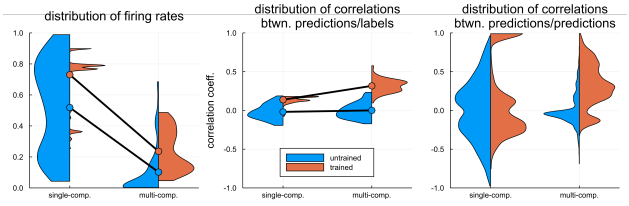


Figure 4: Statistical properties of (un)trained single- and multi-compartment neurons. Left panel: distributions of the response probability of ensemble members for single- and multi-compartment models (left and right, respectively) before and after training (blue and orange, respectively). Middle panel: distribution of correlation coefficients between ensemble members' predictions and the target labels of the test dataset. Right panel: distribution of correlation coefficients between the predictions of pairs of neurons from the ensembles.

2.5 Quantifying the results

To quantify the statements made above, some statistics of the classification results from both enesembles of single- and multi-compartment neurons can be calculated. These results are summarized in figure 4. While untrained single-compartment neurons show a very broad distribution of high firing rates (firing anywhere from 0% to 100% of the data points), they converge to one of a few possible solutions during training, which result in a few narrow bands of relatively high firing rates. The corresponding firing rates for untrained and trained multi-compartment models show a drastic reduction in firing rates, demonstrating that, indeed, these neurons produce a much sparser code. This reduction in firing rate, however, has no negative impact on the predictive power of the resulting spikes; to the contrary, the outputs of the multi-compartment neurons are, as expected, much more correlated with the target labels than the outputs of the single-compartment neurons. Of course, the untrained version of both models is barely correlated by chance (correlation coefficiencts distributed around 0.0). To verify that the multi-compartment neurons are in fact more diverse and independent, the right panel of figure 4 shows a comparison of the distribution of the mutual correlation coefficients between the outputs produced by each pair of two neurons. For untrained single-compartment neurons, this distribution is very broad, whereas for a untrained multi-compartment neurons, it is distributed according to a narrow peak close to 0.0, confirming the expected uncorrelatedness. For trained neurons, the mutual correlation is naturally increased (as all neurons are optimized to produce the same target labels). For single-compartment neurons, however, a striking peak at a correlation coefficient of 1.0 can be observed, which indicates that a lot of neurons tend to approach identical solutions. In an ensemble setting, this indicates that the multi-compartment models represent the better "weak learners", as their errors are more likely to be uncorrelated and can therefore be reduced by averaging.

2.6 How deep is deep enough?

The discussion so far has left two serious questions unanswered: firstly, how deeply nested should the multi-compartment neurons ideally be? And secondly, how do such nested neurons learn in biology? To the first one, there does not seem to be a satisfying answer: just like in decision trees, the choice of depth ultimately constitutes a meta-parameter that is highly task dependent: too simple of a structure, and the neurons would converge to the same, insufficient solution; too complex of a structure, and the neurons become ineffective "grandmother neurons", that barely respond to any input stimuli. Luckily, here some intuitions from decision trees can help, where the effect of depth has been extensively studied. For the running example in this paper, for example, the neuron model has been chosen just barely complex enough to be in principle capable of approximately solving the problem.

Answering the second question requires a more in-depth discussion of biological learning mechanisms and goes beyond the scope of this paper. I'd merely like to stress at this point, that the proposed rule, while implausible as a direct mechanism, only makes use of the locally available, feed-forward information and the supervised training labels, which need to be supplied to all dendritic segments of the neuron by some global signal, but does not require any form of specific feed-back signals to each segment from the soma. A different, yet qualitatively similar, biological mechanism could therefore optimize biological neurons in a greedy fashion, as well, and thus circumvent the conceptual problems (such as frequent propagation failures[11]) surrounding rules based on back-propagating action potentials.

3 RELATED WORK

Hierarchical neuron models have been described before in terms of artificial neural networks, with the complex model resembling a tree-structured feed-forward network of single-compartment neurons. You may at this point wonder: How is this different from the hierarchical models discussed here, and why is the decision tree perspective helpful? There is a subtle, but I believe important, difference between the neuron-as-a-network and the neuron-as-a-decision-tree perspective: In the former, each compartment would be realized by a simple spiking neuron, and the inputs into a segment from its synaptic inputs as well as directly connected child segments would be combined linearly. In that model, the neuron thus corresponds to a regular feed-forward neural network, with each segment corresponding to one neuron of the single-compartment type. This view is appealing, since it merely requires extending our models of neural networks by some additional layers to capture the intrinsic complexity of neurons (and therefore changes little from the perspective of neuromorphic hardware development). As shown in figure 1 c.), such a feed-forward structure could equally select a specific subset of the data and thus act as a sparse pattern detector.

However, despite its appeal, there is a considerable downside when compared to the neuron-as-a-decision-tree perspective: in the former, each segment operates on the input from its child branches, and therefore all information must be transmitted through several levels of the hierarchy, whereas in the latter, each segment operates directly in the input space and only propagates a binary decision ("accept"/"reject") to its parent branch. This also causes a conceptual problem for the neuron-as-a-network perspective, since no simple mechanism for adjusting the relative weights of entire dendritic segments are known. The much simpler structure of the model discussed here also facilitates the analysis, and allows for using a greedy training algorithm in the first place.

Multi-compartment models have also been extensively discussed in various other contexts from neuroscience to machine learning, e.g. in [13] or in [5], which conceptualizes a dendritic arbor as a deep neural network, or in [3], which uses a different motivation to derive a similar conclusion regarding the computational capabilities of nonlinear dendrites to implement boolean functions. Similarly, decision trees and their relation to neural networks have been discussed before[7]. What I believe to be new in this work is the direct application of the decision tree inference and optimization algorithms to a model of individual spiking multi-compartment neurons as well as the discussion and analysis of the impact on sparse coding, specifically in the context of neuromorphic hardware.

4 DISCUSSION

Multi-compartment models as presented here offer a trade-off between the complexity of neurons, which leads to an increased footprint of each neuron, and the amount of information content conveyed per spike, which can lead to a reduction of the required number of spikes per second or the required number of neurons and thus energy consumption. Since the energy and footprint expended on communication grows quadratically with the number of neurons in the network, this trade-off seems favourable for larger networks, specifically for spiking networks in neuromorphic hardware, where sparse codes with low spike-rates can lead to drastic energy savings. To illustrate the potential impact of such models, I presented simulation experiments for the inference stage, while using a modified decision tree training algorithm for optimizing the example neurons. But two difficult challenges, how to train deep hierarchies of such neurons (that are by themselves hierarchical) and how to account for the timing of input signals, remain. Despite these questions left unanswered, I believe that hierarchical neuron models offer a promising enough prospect for computational neuroscience as well as for the development of neuromorphic hardware to warrant much more investigation in the future.

5 CODE AVAILABILITY

All simulations are done using the julia programming language[1]. The code for all simulations is publically available online as a software package in the version control repository under the following address: https://github.com/jleugeri/MultiComp.jl.

ACKNOWLEDGMENTS

This paper has been created as part of my work at Fraunhofer Gesellschaft and has not been funded by any other sources. The author is not aware of any conflict of interest.

REFERENCES

[1] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah. 2017. Julia: a fresh approach to numerical computing. *SIAM Rev.* 59, 1 (Jan. 2017), 65–98. https://doi.org/10.1137/141000671

[2] Leo Breiman (Ed.). 1998. *Classification and regression trees* (repr ed.). Chapman & Hall [u.a.], Boca Raton. OCLC: 247053926.

[3] Romain Cazé, Mark Humphries, and Boris S. Gutkin. 2012. Spiking and saturating dendrites differentially expand single neuron computation capacity. 1070–1078. http://papers.neurips.cc/paper/4738-spiking-and-saturating-dendrites-differentially-expand-single-neuron-computation-capacity

[4] Albert Gidon, Timothy Adam Zolnik, Pawel Fidzinski, Felix Bolduan, Athanasia Papoutsi, Panayiota Poirazi, Martin Holtkamp, Imre Vida, and Matthew Evan Larkum. 2020. Dendritic action potentials and computation in human layer 2/3 cortical neurons. *Science* 367, 6473 (Jan. 2020), 83–87. https://doi.org/10.1126/science.aax6239

[5] Jordan Guerguiev, Timothy P Lillicrap, and Blake A Richards. 2017. Towards deep learning with segregated dendrites. *eLife* 6 (dec 2017), e22901. https://doi.org/10.7554/eLife.22901

[6] Michael Häusser and Bartlett Mel. 2003. Dendrites: bug or feature? *Current Opinion in Neurobiology* 13, 3 (June 2003), 372–383. https://doi.org/10.1016/S0959-4388(03)00075-8

[7] Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulo. 2015. Deep neural decision forests. In *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, Santiago, Chile, 1467–1475. https://doi.org/10.1109/ICCV.2015.172

[8] Johannes Leugering, Pascal Nieters, and Gordon Pipa. 2019. *Event-based pattern detection in active dendrites.* preprint. Neuroscience. https://doi.org/10.1101/690792

[9] Panayiota Poirazi, Terrence Brannon, and Bartlett W. Mel. 2003. Pyramidal neuron as two-layer neural network. *Neuron* 37, 6 (March 2003), 989–999. https://doi.org/10.1016/S0896-6273(03)00149-1

[10] Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. 2019. Going deeper in spiking neural networks: vgg and residual architectures. *Frontiers in Neuroscience* 13 (2019). https://doi.org/10.3389/fnins.2019.00095

[11] Greg Stuart, Nelson Spruston, Bert Sakmann, and Michael Häusser. 1997. Action potential initiation and backpropagation in neurons of the mammalian CNS. *Trends in Neurosciences* 20, 3 (March 1997), 125–131. https://doi.org/10.1016/S0166-2236(96)10075-8

[12] Balázs B Ujfalussy, Judit K Makara, Tiago Branco, and Máté Lengyel. 2015. Dendritic nonlinearities are tuned for efficient spike-based computations in cortical circuits. *eLife* 4 (Dec. 2015), e10056. https://doi.org/10.7554/eLife.10056

[13] Robert Urbanczik and Walter Senn. 2014. Learning by the dendritic prediction of somatic spiking. *Neuron* 81, 3 (Feb. 2014), 521–528. https://doi.org/10.1016/j.neuron.2013.11.030

Anmelder: Universität Osnabrück
Vertreter: Patentanwalt Dr.-Ing. Ch. Holz (VNR 88 88 26)
Datum: 11.12.2019
intern. Aktenz.: EZN 0016-01DE

BESCHREIBUNG

Neuromorpher Musterdetektor und neuromorphe Schaltkreisanordnung hiermit

Die vorliegende Erfindung betrifft einen neuromorphen Musterdetektor gemäß dem Patentanspruch 1 sowie eine neuromorphe Schaltkreisanordnung gemäß dem Patentanspruch 13.

Zur Verarbeitung ihrer Informationen können analog erfasste Signale, welche z.B. sensorisch erfasste Informationen repräsentieren können, in digitale Signale gewandelt und dann verarbeitet werden. Die Erfassung der analogen Signale kann üblicherweise mittels elektrischer Spannung erfolgen, welche einen zeit- und wertkontinuierlichen Verlauf, d.h. eine durchgängigen Verlauf der elektrischen Spannung über der Zeit, aufweist. Ein derartiges elektrisches Analogsignal kann mittels eines Analog-Digital-Umsetzers in ein digitales Signal in Form eines zeit- und wertdiskreten Verlaufs gewandelt werden, um die Information der digitalen Signalverarbeitung zugänglich zu machen. Ein derartiges digitales Signal kann auch als binäres Signal bezeichnet werden und zwei unterschiedliche Zustände in Form von unterschiedlich hohen elektrischen Spannungspegeln aufweisen, so dass über die Länge bzw. Dauer des Signalverlaufs zwischen niedrigen und hohen Spannungspegeln unterschieden werden kann. Die niedrigen Spannungspegel können als „low“-Zustände und die hohen Spannungspegel als „high“-Zustände bezeichnet werden. Hierdurch können die Zustände „0“ und „1“ dargestellt werden.

Die digitale Signalverarbeitung mittels entsprechender elektronischer Bauelemente wie z.B. digitale Signalprozessoren und Mikroprozessoren bietet dabei Vorteile und Möglichkeiten, welche mit analog arbeitender Elektronik gar nicht oder lediglich mit hohem Aufwand umsetzbar wären. Dabei werden die digitalen Signale üblicherweise nicht als die binären Signale eines Verlaufs von Nullen und Einsen in Form von niedrigen und hohen Spannungszuständen verarbeitet sondern als Werte etc. z.B. in Folgen von acht Bits, auch Byte genannt, dargestellt, gespeichert und durch Software verarbeitet. Die entsprechenden Algorithmen, welche die Verarbeitung der digitalen Signale durchführen, werden hierzu als Programmcode einer geeigneten Programmiersprache umgesetzt und z.B. auf einem Mikroprozessor oder auf einer CPU (Central Processing Unit) als serielle Abfolge der Programmierschritte ausgeführt. Mit anderen Worten werden in der digitalen Signalverarbeitung üblicherweise die Instruktionen numerischer Algorithmen von Prozessorarchitekturen auf Binärzahlen implementieren, was die sequentielle Abarbeitung in arithmetisch-logischen Einheiten (ALU) und die Verwendung einer Speichereinheit bedingt.

Zur Verarbeitung digitaler sowie analoger Informationen in Form digitaler Signale können auch sog. künstliche neuronale Netze bzw. Netzwerke verwendet werden, bei denen mittels künstlicher Neuronen die Funktionsweisen biologischer Neuronen bzw. biologischer neuronaler Netze bzw. Netzwerke nachgebildet werden. Die einzelnen künstlichen Neuronen arbeiten dabei zeitlich parallel zueinander, vergleichbar der Vorbilder der biologischen Neuronen. Da eine derartige Arbeitsweise mit den sequentiell arbeitenden Prozessoren strukturbedingt jedoch nicht möglich ist, kann die parallele Arbeitsweise der künstlichen Neuronen auch bei Verwendung mehrerer paralleler Prozessoren bzw. Prozessorkerne nur unzureichend implementiert werden. Dies erschwert die tatsächliche Implementierung parallel arbeitender Verfahren zur digitalen Signalverarbeitung mit künstlichen neuronalen Netzen.

Zur Implementierung von künstlichen neuronalen Netzen bzw. Netzwerken werden daher auch neuromorphe Schaltkreise verwendet, welche jeweils ein biologisches Neuron als elektronische Schaltung abbilden und durch ihr Zusammenwirken das künstliche neuronale Netz bzw. Netzwerk ergeben. Die einzelnen neuromorphen Schaltkreise können dabei tatsächlich parallel zueinander arbeiten und hierdurch die Signalverarbeitung beschleunigen bzw. die als Vorbild dienenden biologischen Neuronen besser nachbilden.

Typischerweise wird das Verhalten des einzelnen künstlichen Neurons dynamischen Systemen aus den theoretischen Neurowissenschaften wie z.B. dem Leaky-Integrate-and-Fire-Modell nachempfunden, durch digitale Arithmetik approximiert und der Datenaustausch zwischen den künstlichen Neuronen durch die Übertragung von Paketen realisiert. Dies erfordert jedoch den Einsatz vieler Recheneinheiten bzw. vieler arithmetisch-logischer Einheiten und stellt hohe Anforderungen an das Paket-Routing zwischen den einzelnen Recheneinheiten. Im speziellen Bereich der Spiking-Neuromorphic-Hardware wird dabei pro künstlichen Neuron und pro Zeitschritt lediglich ein binäres Signal erzeugt.

In einem parallel signalverarbeitenden neuromorphen Netz bzw. Netzwerk der Digitaltechnik sollten somit folgende technische Probleme gelöst bzw. folgende technische Eigenschaften realisiert werden:

- Künstliche Neurone sollte Eingangssignale von vielen anderen künstlichen Neuronen integrieren können. Dies erfordert einen Mechanismus, um Eingangssignale aufzuaddieren und mit einem kritischen Grenzwert vergleichen zu können. In bestehenden Ansätzen der digitalen Signalverarbeitung wird dies mittels ALUs durch Ganzzahlarithmetik realisiert.

- Das Einsatzgebiet von digitaler Signalverarbeitung ist häufig durch das Erfordernis der Echtzeitfähigkeit ausgezeichnet, d.h. durch die Fähigkeit des Betriebssystems der Recheneinheit bzw. der Recheneinheiten, digitale Signale innerhalb einer vorbestimmbaren Frist sicher verarbeiten zu können. Die Einhaltung einer Reaktion auf das digitale Signal innerhalb dieser Frist muss in diesem Fall sichergestellt sein.

So sind die Zeitskalen, auf welche ein analoges Signal in der Außenwelt relevante und zu verarbeitende Charakteristika aufweist, nicht fest und zum Teil auf schnellen oder langsamen Skalen variiert. Daher müssen die Zeitskalen der Verarbeitung digitaler Signal im integrierten Schaltkreis von denen in der Außenwelt entkoppelt werden. Klassische Ansätze der digitalen Signalverarbeitung in z.B. Mikrokontrollern umgehen dieses Problem, indem Zwischenergebnisse im dedizierten Arbeitsspeicher abgelegt werden.

In neuromorphen Ansätzen wird Information meist stattdessen lokal im Zustand der einzelnen Neuronen gehalten. Ggfs. kann die Rate, mit der sich der Zustand des Neurons pro Zeitschritt ändert, skaliert und auf die relevante Zeitskala des Eingangssignals abgestimmt werden. Die Verarbeitung langsamer Signale mit einem schnellen Takt erfordert daher einen hoch aufgelösten internen Zustand der Neuronen.

- Um komplexere Funktionalitäten wie das Erkennen von Mustern mittels neuromorpher Schaltkreise abzubilden, müssen viele Neuronen sinnvoll miteinander verschaltet werden. Dies wird gegenwärtig durch verschiedene Mesh- und Crossbar-Routing-Systeme implementiert, welche bestimmte Konfigurationen zulassen und Output-Signale dem Input verschiedener Neuronen zuordnen. Die Verbindungen zwischen einzelnen Neuronen sind dabei meist unterschiedlich gewichtet, was einen entsprechenden Mechanismus zur verbindungsspezifischen Konfiguration und Signalübertragung erfordert.

- Um mit verrauschten Eingangssignalen umgehen zu können, sollte als Ausgangssignal nicht nur das gewünschte Signal, z.B. ob ein gegebenes Muster erkannt wurde oder nicht, sondern auch ein Maß der zugehörigen Unsicherheit generiert werden. Dies kann von bestehenden Ansätzen lediglich mittelbar unter Rückgriff auf bestimmte Netzwerkarchitekturen realisiert werden, ist aber nicht in der Hardware selbst angelegt.

Somit weisen die bestehenden Ansätze spike-basierter neuromorpher Hardware, welche auf gepulsten neuronalen Netzen (Englisch: spiking neural networks - SNN) beruhen, verschiedene Nachteile auf. So erfordert die Verwendung von Ganzzahlarithmetik und Paket-Routing den Einsatz von Mikroprozessoren, was die technische Komplexität der Hardware erhöhen und aufgrund ihrer sequentiellen Operation zu Latenzen führen kann. Auch kann die Beschränkung auf einfache generische Neuronenmodelle mit gewichteten Verbindungen, welche nicht für die Analyse von kontinuierlichen Signalströmen entwickelt wurden, zur Verwendung von notwendigerweise großen Netzwerken führen, deren interne Kommunikation viel Platz-, Energie- und bzw. oder Zeitressourcen beanspruchen kann.

Eine Aufgabe der vorliegenden Erfindung ist es, einen neuromorphen Schaltkreis bereitzustellen, um die zuvor genannten technischen Probleme zu lösen bzw. die zuvor genannten technischen Eigenschaften

zu realisieren. Insbesondere soll ein zu erkennendes Muster in einem binären Eingangssignal schneller und bzw. oder zuverlässiger als bisher bekannt erkannt werden können. Zumindest soll eine Alternative zu bekannten derartigen neuromorphen Schaltkreisen bereitgestellt werden.

Die Aufgabe wird erfindungsgemäß durch einen neuromorphen Musterdetektor mit den Merkmalen des Patentanspruchs 1 sowie durch eine neuromorphe Schaltkreisanordnung mit den Merkmalen des Patentanspruchs 13 gelöst. Vorteilhafte Weiterbildungen sind in den Unteransprüchen beschrieben.

Somit betrifft die Erfindung einen neuromorphen Musterdetektor, welcher ausgebildet ist, wenigstens zwei 1-Bit Eingangssignale eines zu erkennenden Musters zu erhalten, mit wenigstens zwei Vergleichsschaltungen, welche jeweils ausgebildet sind, eines der 1-Bit Eingangssignale zu erhalten, die Anzahl der „high“-Zustände oder der „low“-Zustände des jeweiligen 1-Bit Eingangssignals innerhalb eines vorbestimmten Zeitraums zu zählen, die Anzahl der gezählten Zustände mit einem vorbestimmten Schwellwert der jeweiligen Vergleichsschaltung zu vergleichen und bei Überschreiten des Schwellwerts auf die erfolgte bzw. auf die erfolgreiche Erkennung des zu erkennenden Musters hinzuweisen. Der neuromorphe Musterdetektor ist vorzugsweise mittels Digitaltechnik umgesetzt.

Mit anderen Worten werden wenigstens zwei 1-Bit Datenströme, welche gemeinsam ein zu erkennendes Muster in Form einer parallelen Bitfolge enthalten, dem erfindungsgemäßen neuromorphen Musterdetektor in Form einer neuromorphen Schaltung zugeführt. Über eine vorbestimmte Anzahl von Bit, welche dem vorbestimmten Zeitraum entsprechen, werden nun die „high“-Zustände oder die „low“-Zustände, d.h. die hohen Signalpegel oder die niedrigen Signalpegel, gezählt. Diese Anzahl wird fortlaufend mit einem Schwellwert verglichen. Wird dieser Schwellwert überschritten, so wird hieraus geschlussfolgert, dass zu erkennende Muster in dem jeweiligen 1-Bit Datenstrom der jeweiligen Vergleichsschaltung erkannt zu haben. Dies wird von dem neuromorphen Musterdetektor nach außen angezeigt, z.B. über ein entsprechendes Ausgangssignal.

Auf diese Art und Weise kann erfindungsgemäß vergleichsweise einfach mittels einer neuromorphen Schaltung eine Mustererkennung in einem digitalen Signal erfolgen.

Gemäß einem Aspekt der Erfindung ist die eine Vergleichsschaltung der anderen Vergleichsschaltung erstrangig untergeordnet, wobei die übergeordnete Vergleichsschaltung ausgebildet ist, nur dann auf die erfolgte Erkennung des zu erkennenden Musters hinzuweisen, falls der Schwellwert der übergeordneten Vergleichsschaltung überschritten und zeitgleich von der erstrangig untergeordneten Vergleichsschaltung auf die erfolgte Erkennung des zu erkennenden Musters hingewiesen wird.

Dies kann es ermöglichen, die Entscheidung der übergeordneten Vergleichsschaltung von der Entscheidung der untergeordneten Vergleichsschaltung, das vorbestimmte Muster erkannt zu haben oder nicht, abhängig zu machen.

Gemäß einem weiteren Aspekt der Erfindung weist der neuromorphe Musterdetektor wenigstens eine weitere Vergleichsschaltung auf, welche parallel zu der untergeordneten Vergleichsschaltung angeordnet ist, wobei die übergeordnete Vergleichsschaltung ausgebildet ist, nur dann auf die erfolgte Erkennung des zu erkennenden Musters hinzuweisen, falls der Schwellwert der übergeordneten Vergleichsschaltung überschritten und zeitgleich von den erstrangig untergeordneten Vergleichsschaltungen jeweils auf die erfolgte Erkennung des zu erkennenden Musters hingewiesen wird.

Dies kann es ermöglichen, die Entscheidung der übergeordneten Vergleichsschaltung von der Entscheidung der beiden untergeordneten Vergleichsschaltungen, das vorbestimmte Muster erkannt zu haben oder nicht, abhängig zu machen.

Gemäß einem weiteren Aspekt der Erfindung weist der neuromorphe Musterdetektor wenigstens eine weitere Vergleichsschaltung auf, welche zweitrangig untergeordnet zu der erstrangig untergeordneten Vergleichsschaltung angeordnet ist, wobei die erstrangig untergeordnete Vergleichsschaltung ausgebildet ist, nur dann auf die erfolgte Erkennung des zu erkennenden Musters hinzuweisen, falls der Schwellwert der erstrangig untergeordneten Vergleichsschaltung überschritten und zeitgleich von der zweitrangig untergeordneten Vergleichsschaltung auf die erfolgte Erkennung des zu erkennenden Musters hingewiesen wird.

Dies kann es ermöglichen, die Entscheidung der erstrangig untergeordneten Vergleichsschaltung von der Entscheidung der zweitrangig untergeordneten Vergleichsschaltung, das vorbestimmte Muster erkannt zu haben oder nicht, abhängig zu machen.

Dabei können die zuvor beschriebenen Möglichkeiten der Anordnung von mehr als zwei Vergleichsschaltungen auch miteinander kombiniert werden, indem wenigstens zwei erstrangige und wenigstens eine zweitrangige Vergleichsschaltung verwendet und wie zuvor beschrieben miteinander und bzw. oder seitens der übergeordneten Vergleichsschaltung in Abhängigkeit gesetzt werden.

Gemäß einem weiteren Aspekt der Erfindung bilden die wenigstens drei Vergleichsschaltungen einen Binärbaum mit wenigstens zwei Ebenen. Unter einem Binärbaum, auch binärer Baum genannt, wird eine besondere Unterart eines Baumes verstanden, wie er in der Informatik für hierarchische Datenstrukturen verwendet wird. Der Ausgangspunkt, wie hier die übergeordnete Vergleichsschaltung, wird als Wurzel oder auch Binärbaumwurzel bezeichnet, von welcher sich der Binärbaum in verschiedenen Ebene

wie hier der erstrangigen und zweitrangigen Vergleichsschaltungen einzeln oder paarweise verzweigt, bis der jeweilige Ast an einem Binärbaumblatt endet.

Entsprechend können die Eigenschaften und Vorteile derartiger hierarchischer Datenstrukturen auf die erfindungsgemäße neuromorphe Schaltung übertragen und dort genutzt werden.

Gemäß einem weiteren Aspekt der Erfindung sind die Vergleichsschaltungen identisch ausgebildet. Dies kann die Umsetzung vereinfachen, da der Entwurf der neuromorphen Schaltung mit geringerem Aufwand ausfallen kann, in dem das Design der Vergleichsschaltung mehrfach verwendet wird. Auch kann dies die Vergrößerung der Schaltung des neuromorphen Musterdetektors vereinfachen und hierdurch eine Skallierbarkeit ermöglichen.

Gemäß einem weiteren Aspekt der Erfindung wird bei Überschreiten des Schwellwerts ein 1-Bit Ausgangssignal der jeweiligen Vergleichsschaltung auf den „high“-Zustand, ansonsten auf den „low“-Zustand, gesetzt, oder umgekehrt. Dies kann es ermöglichen, dass Hinweisen der jeweiligen Vergleichsschaltung auf die erfolgte Erkennung des zu erkennenden Musters einfach umzusetzen.

Gemäß einem weiteren Aspekt der Erfindung sind die Vergleichsschaltungen ausgebildet, jeweils ein 1-Bit Steuersignal zu erhalten und in Reaktion auf einen „high“-Zustand oder auf einen „low“-Zustand des jeweiligen 1-Bit Steuersignals das 1-Bit Ausgangssignal der jeweiligen Vergleichsschaltung auf den „low“-Zustand zu setzen. Hierdurch kann eine Möglichkeit geschaffen werden, die entsprechende Vergleichsschaltung mittels des jeweiligen 1-Bit Steuersignals wieder zurückzusetzen. Mit anderen Worten kann die Vergleichsschaltung von außen resetted werden. Dies kann es insbesondere ermöglichen, alle Vergleichsschaltungen zurückzusetzen, um anschließend mit dem Erkennen eines neuen Musters beginnen zu können, ohne dass der zuvor erfolgte Vorgang auf dessen Ergebnis Auswirkungen haben kann.

Gemäß einem weiteren Aspekt der Erfindung gibt der vorbestimmte Schwellwert der Anzahl der Zustände der jeweiligen Vergleichsschaltung vor, wann das zu erkennende Muster als erkannt angesehen wird. Mit anderen Worten kann durch die Höhe des Schwellwerts in Relation zur Länge bzw. Kürze des vorbestimmten Zeitraums bzw. der vorbestimmten Anzahl von Bit des Eingangssignals vorbestimmt werden, wie deutlich eine Übereinstimmung zwischen dem jeweiligen Eingangssignal und dem vorbestimmten Muster vorliegen muss, um das vorbestimmte Muster im jeweiligen Eingangssignal als erkannt anzusehen. Dies kann für jedes zu erkennende Muster und für jede Vergleichsschaltung vorgegeben werden. Dies kann über die Konfiguration der Vergleichsschaltungen erfolgen.

Gemäß einem weiteren Aspekt der Erfindung weisen die Vergleichsschaltungen jeweils einen Schiebefensterdetektor auf, welcher jeweils ausgebildet ist, das jeweilige 1-Bit Eingangssignal zu erhalten und die Anzahl der „high“-Zustände oder der „low“-Zustände des jeweiligen 1-Bit Eingangssignals innerhalb

des vorbestimmten Zeitraums zu zählen. Dies kann die Umsetzung dieser Funktion der Vergleichsschaltungen einfach und bzw. oder zuverlässig ermöglichen.

Gemäß einem weiteren Aspekt der Erfindung erfolgt das Zählen der Anzahl der „high“-Zustände oder der „low“-Zustände des jeweiligen 1-Bit Eingangssignals innerhalb des vorbestimmten Zeitraums mittels eines bidirektionalen Schieberegisters des jeweiligen Schiebefensterdetektors. Dies kann die Umsetzung dieser Funktion der Vergleichsschaltungen einfach und bzw. oder zuverlässig ermöglichen.

Gemäß einem weiteren Aspekt der Erfindung erhalten die Vergleichsschaltungen, vorzugsweise deren Schiebefensterdetektor, jeweils ein Taktsignal zur Steuerung der Verarbeitung der Pulse und ein Taktsignal zur Steuerung der Länge der Plateaus, wobei die beiden Taktsignale unterschiedlich sind. Unter einem Puls bzw. Spike ist der Zustand eines Signals im Zustand „high“ nach und vor einem Zustand „low“ zu verstehen. Unter einem Plateau eines Signals ist die Zeitdauer bzw. die Signallänge im Zustand „high“ zu verstehen. Mit anderen Worten ist unter eine Plateau eine Funktion vergleichbar einem volatilen Cache-Zwischenspeicher zu verstehen, welcher für eine konfigurierbare Zeit, d.h. die Zeitdauer des Plateaus, ein Zwischenergebnis speichert. Auf diese Art und Weise kann die Mustererkennung der Vergleichsschaltungen und damit auch des neuromorphen Musterdetektors in Abhängigkeit von wenigstens zwei unterschiedlichen Taktsignalen erfolgen.

Die vorliegende Erfindung betrifft auch eine neuromorphe Schaltkreisanordnung mit einer Mehrzahl von neuromorphen Musterdetektoren wie zuvor beschrieben, wobei jeder neuromorphe Musterdetektor ausgebildet ist, das gleiche 1-Bit Eingangssignal zu erhalten, ein unterschiedliches 1-Bit Zufallszahlensignal zu erhalten, das jeweilige 1-Bit Eingangssignal mit dem entsprechenden 1-Bit Zufallszahlensignal zu verändern, und die Anzahl der „high“-Zustände oder der „low“-Zustände des jeweiligen veränderten 1-Bit Eingangssignals innerhalb eines vorbestimmten Zeitraums zu zählen.

Unter einem 1-Bit Zufallszahlensignal ist ein Signal mit einer Bitfolge zu verstehen, welche zufällig erzeugt wurde. Dies kann deterministisch oder nicht-deterministisch erfolgen. Ein deterministisch erzeugtes 1-Bit Zufallszahlensignal kann auch als Pseudo-Zufallszahlensignal bezeichnet werden. Dabei kann die Verwendung eines pseudo-zufälligen 1-Bit Zufallszahlensignals vorteilhaft sein, da dies einfacher als ein nicht-deterministisches 1-Bit Zufallszahlensignal erzeugt werden und zur Erzielung der entsprechenden Eigenschaften und Vorteile ausreichend sein kann.

Somit können mehrere der zuvor beschriebenen neuromorphen Musterdetektoren parallel zueinander angeordnet und verwendet werden, um jeweils das gleiche vorbestimmte Muster in dem gleichen Eingangssignal zu erkennen. Hierbei können die beiden 1-Bit Datenströme jeweils unterschiedlich stochastisch verändert werden, so dass das gleiche Muster jeweils in unterschiedlichen Eingangssignalen der einzelnen neuromorphen Musterdetektoren erkannt werden muss. Dies kann eine Aussage über die

Zuverlässigkeit der Mustererkennung erlauben, da die gleichen Eingangssignal mit dem zu erkennenden Muster durch die 1-Bit Zufallssignale unterschiedliche verfremdet bzw. gestört jeweils identisch durch die neuromorphen Musterdetektoren bearbeitet werden.

Diesbezüglich sei angemerkt, dass ein (pseudo-)zufälliges Maskieren eines Datenstroms in mehrere sich zufällig unterscheidende Datenströme auch unabhängig von einer neuromorphen Schaltkreisanordnung wie zuvor beschrieben und insbesondere unabhängig von einer Mehrzahl von neuromorpher Musterdetektoren wie zuvor beschrieben umgesetzt und angewendet werden kann. Dies kann es ermöglichen, die entsprechenden Eigenschaften und Vorteile auch unabhängig umzusetzen und anzuwenden.

Gemäß einem Aspekt der Erfindung weist wenigstens eine Vergleichsschaltung, vorzugsweise weisen alle Vergleichsschaltungen jeweils, ein Und-Gatter auf, welches ausgebildet ist, das jeweilige 1-Bit Eingangssignal und das entsprechende 1-Bit Zufallszahlensignal zu kombinieren. Hierdurch kann die Veränderung der gleichen Eingangssignal durch die unterschiedlichen stochastischen 1-Bit Zufallszahlensignal umgesetzt werde.

Gemäß einem weiteren Aspekt der Erfindung ist die neuromorphe Schaltkreisanordnung ausgebildet, die Anzahl der 1-Bit Ausgangssignale der jeweiligen Vergleichsschaltung, welche zeitgleich im „high“-Zustand oder im „low“-Zustand sind, zu erfassen und aus dem Verhältnis der Anzahl von 1-Bit Ausgangssignalen im „high“-Zustand oder im „low“-Zustand und der Anzahl der neuromorphen Musterdetektoren einen Grad der Übereinstimmung zwischen 1-Bit Eingangssignal und zu erkennendem Muster zu bestimmen. Hierdurch kann diese Information bestimmt und zur Verfügung gestellt werden.

Gemäß einem weiteren Aspekt der Erfindung weist wenigstens eine Vergleichsschaltung, vorzugsweise weisen alle Vergleichsschaltungen jeweils, einen Zeitmultiplexer auf, welcher ausgebildet ist, parallele Ausgangssignale der neuromorphen Musterdetektoren zu einem 1-Bit-Ausgangssignalder neuromorphen Schaltkreisanordnung zusammenzuführen. Auf diese Art und Weise kann ein einziger resultierenden 1-Bit Datenstrom als Ausgangssignal der neuromorphen Schaltkreisanordnung erzeugt werden.

Ein Ausführungsbeispiel und weitere Vorteile der Erfindung werden nachstehend im Zusammenhang mit den folgenden Figuren rein schematisch dargestellt und näher erläutert. Darin zeigt:

Fig. 1 eine schematische Darstellung eines Symbols eines Schaltkreises einer Population der Fig. 2;
Fig. 2 eine schematische Darstellung eines Schaltkreises der Population der Fig. 1;
Fig. 3 eine schematische Darstellung eines Symbols eines Schaltkreises eines Neurons der Fig. 4;
Fig. 4 eine schematische Darstellung eines Schaltkreises des Neurons der Fig. 3;
Fig. 5 eine schematische Darstellung eines Schaltkreises eines Binärbaumzweigs;
Fig. 6 eine schematische Darstellung eines Schaltkreises eines Abschlusszweigs;

Fig. 7 eine schematische Darstellung eines Symbols eines Schaltkreises eines Segments der Fig. 8;

Fig. 8 eine schematische Darstellung eines Schaltkreises des Segments der Fig. 7;

Fig. 9 eine schematische Darstellung eines Symbols eines Schaltkreises eines Schiebefensterdetektors der Fig. 10;

Fig. 10 eine schematische Darstellung eines Schaltkreises des Schiebefensterdetektors der Fig. 9;

Fig. 11 eine schematische Darstellung eines Symbols eines Schaltkreises eines Zeitmultiplexers der Fig. 12; und

Fig. 12 eine schematische Darstellung eines Schaltkreises des Zeitmultiplexers der Fig. 11.

Fig. 1 zeigt eine schematische Darstellung eines Symbols eines Schaltkreises einer Population 1 der Fig. 2. Fig. 2 zeigt eine schematische Darstellung eines Schaltkreises der Population 1 der Fig. 1.

Unter einer Population 1 im Sinne von Computersoftware wird eine Anordnung von gleichen Computerprogrammen verstanden, welche gemeinsam die Population 1 bilden. Wird dies auf neuromorphe Schaltkreise übertragen, so kann die o.g. Population 1 mittels neuromorpher Schaltkreise als neuromorphe Schaltkreisanordnung 1 gebildet werden, indem mehrere neuromorphe Musterdetektoren 2, welche auch als Neuronen 2 bezeichnet werden können, gleicher Struktur in Form von identisch ausgebildeten neuromorphen Schaltkreisen miteinander zur Population 1 verschaltet werden, siehe Figur 2.

Die Population 1 besteht dabei gemäß dem dargestellten Ausführungsbeispiel aus einer Anzahl K von Neuronen 2, von welchen in der Figur 2 das erste, das zweite und das K-te Neuron 2 von links nach rechts dargestellt sind. Jedes Neuron 2 erhält denselben eingehenden Datenstrom E als Eingangssignal E, welches aus einer Anzahl N von einzelnen 1-Bit Eingangssignalen E_1-E_N besteht. Das Eingangssignal E enthält ein zu erkennendes Muster, welches auch als Pattern bezeichnet werden kann.

Jedes Neuron 2 erhält ferner parallel dasselbe Steuersignal I, welches aus einer Mehrzahl von einzelnen 1-Bit Steuersignalen I_1-I_N besteht. Das Steuersignal I kann zum Zurücksetzen von Vergleichsschaltungen 2, auch Segmente 3 genannt, innerhalb der Neuronen 2 verwendet werden, wie weiter unten näher beschrieben werden wird.

Ferner erhält jedes Neuron 2 eine Anzahl N von binären Zufallszahlensignalen $M_{1,1}$-$M_{K,N}$, welche deterministisch erzeugt und für jedes Neuron 2 unterschiedlich pseudo-zufällig sind. Genauer gesagt wird der Population 1 für jedes der N Eingangssignale E_1-E_N und für jedes der K Neuronen 2 ein zufälliges 1-Bit Zufallssignal $M_{1,1}$-$M_{K,N}$ zur Verfügung gestellt.

Des Weiteren erhält jedes Neuron 2 drei unterschiedliche Taktsignal CLK_{PLT}, CLK_{SPIKE} und CLK_{PROG}. Das Taktsignal CLK_{PLT} ist ein Taktsignal zur Steuerung der Länge der Plateaus der Vergleichsschaltungen 3, wie weiter unten noch näher erläutert werden wird. Das Taktsignal CLK_{SPIKE} ist ein Taktsignal zur Steue-

rung der Verarbeitung von Spikes, d.h. von Pulsen, der Vergleichsschaltungen 3, wie ebenfalls weiter unten noch näher erläutert werden wird. Das Taktsignal CLK_{PROG} ist ein Taktsignal eines Konfigurationssignals D_{PROG} bzw. D_{PROGO}, wie ebenfalls weiter unten noch näher erläutert werden wird.

Ein Konfigurationssignal D_{PROG} der Population 1 wird als Eingangssignal dem ersten Neuron 2 zugeführt, dort zur Konfiguration des ersten Neurons 2 verwendet und als Konfigurationssignal D_{PROGO} von dem ersten Neuron 2 an das zweite Neuron 2 ausgegeben. Das zweite Neuron 2 erhält somit das Konfigurationssignal D_{PROG} als Eingangssignal usw. Das Konfigurationssignal D_{PROGO} als Ausgangssignal des letzten K-ten Neurons 2 ist das Konfigurationsausgangssignal D_{PROGO} der Population 1.

Jedes der K Neuronen 2 erzeugt ein binäres Ausgangssignal P_1-P_K, welche parallel einem Zeitmultiplexer 5 als dessen Eingangssignale S_1-S_K zugeführt werden. Die Verarbeitung dieser Eingangssignale S_1-S_K zu einem Ausgangssignal O des Zeitmultiplexers 5, welches auch das Ausgangssignal O der gesamten Population 1 darstellt, wird weiter unter beschrieben.

Fig. 3 zeigt eine schematische Darstellung eines Symbols eines Schaltkreises eines Neurons 2 der Fig. 4. Fig. 4 zeigt eine schematische Darstellung eines Schaltkreises des Neurons 2 der Fig. 3. Fig. 5 zeigt eine schematische Darstellung eines Schaltkreises eines Binärbaumzweigs 21, 22. Fig. 6 zeigt eine schematische Darstellung eines Schaltkreises eines Abschlusszweigs 20.

Jedes Neuron 2 besteht im Wesentlichen aus einem rekursiv eingebetteten, binären Baum, auch Binärbaum genannt, mit einem ersten Binärbaumzweig 21, einem zweiten Binärbaumzweig 22 sowie dem zuvor bereits erwähnten Segments 3, siehe z.B. Fig. 4. Jeder der beiden Baumzweige 21, 22 kann in jeder Ebene des binären Baums entweder ein weiteres Neuron 2 mit zwei weiteren Binärbaumzweigen 21, 22 und einem Segment 3, siehe Fig. 5, oder ein Abschlusszweig 20 mit lediglich einem Segment 3, siehe Fig. 6, sein. Die beiden Binärbaumzweige 21, 22 können auch als innere Knoten des Binärbaums 21, 22 oder als Nested Branches 21, 22 bezeichnet werden. Der Abschlusszweig 20 kann auch als Binärbaumblatt 20 oder als Terminal Branch 20 bezeichnet werden. Das Neuron 2 selbst kann daher auch als Binärbaumwurzel 2 bezeichnet werden. Mit anderen Worten wird jeder Binärbaumzweige 21, 22 entweder aus einem weiteren Neuron 2, welches seinerseits wieder zwei Binärbaumzweige 21, 22 aufweist, oder aus einem Abschlusszweig 20 gebildet.

Dabei besitzt das jeweilige Segment 3, welches die Wurzel des Binärbaums bildet, die gleiche Struktur wie die Binärbaumzweige 21, 22 der weiteren Ebenen des binären Baums mit dem Unterschied, dass das Segment 3 der Wurzel des Binärbaums statt dem Taktsignal CLK_{PLT} das Taktsignal CLK_{SPIKE} erhält. Die Binärbaumzweige 21, 22 erhalten das Taktsignal CLK_{PLT}. Dies führt dazu, dass das Ausgangssignal P_1-P_k des Neurons 2 als kurze Spikes mit dem Taktsignal CLK_{SPIKE} und nicht lange Plateaus mit dem Taktsignal CLK_{PLT} aufweist.

Das Konfigurationssignal D_{PROG} des Neurons 2 wird jedem Binärbaumzweig 21, 22, jedem Abschlusszweig 20 sowie jedem Segment 3 zugeführt.

Jedes Segment 3 jeder Ebene des binären Baums erhält eines der 1-Bit Eingangssignale E_1-E_N sowie das entsprechende 1-Bit Steuersignale I_1-I_N und das entsprechende Zufallszahlensignal $M_{1,1}$-$M_{K,N}$. Die Funktion des Segments 3 wird weiter unten erklärt werden. Auch erhält jedes Segment 3 jeder Ebene des binären Baums die Taktsignale CLK_{PLT}, CLK_{SPIKE} und CLK_{PROG} zu den zugehörigen Signalen.

Der Abschlusszweig 20 besteht lediglich aus einem Segment 3 mit zwei konstanten Eingangssignalen B_1, B_2, welche beide den Zustand „high“ aufweisen. Ferner erhält das Segment 3 ebenfalls eines der 1-Bit Eingangssignale E_1-E_N sowie das entsprechende 1-Bit Steuersignale I_1-I_N und das entsprechende Zufallszahlensignal $M_{1,1}$-$M_{K,N}$.

Fig. 7 zeigt eine schematische Darstellung eines Symbols eines Schaltkreises eines Segments 3 der Fig. 8. Fig. 8 zeigt eine schematische Darstellung eines Schaltkreises des Segments 3 der Fig. 7.

Das Segment 3, welches wie zuvor beschrieben jeweils identisch in jedem Neuron 2 mehrfach auf verschiedenen Ebenen des binären Baums verwendet wird, erhält stets die Ausgangssignale P der Binärbaumzweige 21, 22 derselben Ebene als Eingangssignal B_1, B_2. Die beiden Eingangssignale B_1, B_2 sind parallel sowohl auf ein erstes Oder-Gatter 30 als auch auf ein erstes Und-Gatter 31 geschaltet. Die Ausgangssignale der beiden ersten Gatter 30, 31 können entweder ein konstantes “high”-Signal oder ein konstantes “low”-Signal sein, welche parallel einem 4-fach Multiplexer 32 zugeführt werden. Zusätzlich zu den beiden Ausgangssignalen der beiden ersten Gatter 30,31 werden ein konstantes „low“-Signal und ein konstantes „high“-Signal parallel dem 4-fach Multiplexer 32 zugeführt. Dabei wird das Ausgangssignal des 4-fach Multiplexers 32 von einem ersten 2-bit SIPO Schieberegister 33 (SIPO: serial-input-parallel-output) gewählt und einem zweiten Und-Gatter 34 zugeführt.

Die zwei 1-Bit Eingangssignale E_1-E_N und $M_{1,1}$-$M_{K,N}$ des jeweiligen Segments 3 werden von einem dritten Und-Gatter 35 verschaltet, dessen Ausgangssignal in einen Schiebefensterdetektor 4 als dessen Eingangssignal D_{IN} geschaltet wird, welcher auch als Slider 4 bezeichnet werden kann und weiter unten näher erläutert werden wird. Der Schiebefensterdetektor 4 wird durch den Datenstrom des Konfigurationssignals D_{PROG} mit dem zugehörigen Taktsignal CLK_{PROG} konfiguriert. Das Ausgangssignal des Konfigurationssignals D_{PROGO} des Schiebefensterdetektors 4 ist wiederum das Eingangssignal des ersten 2-bit SIPO Schieberegisters 33, welches seinerseits durch die steigende Flanke des Taktsignals CLK_{PROG} weitergeschoben wird. Das erste 2-bit SIPO Schieberegister 33 erzeugt dabei parallel zu der zuvor beschriebenen Auswahl des Ausgangssignals des 4-fach Multiplexers 32 das Ausgangssignal des Konfigurationssignals D_{PROGO} des Segments 3.

Der Schiebefensterdetektor 4 erzeugt parallel zu dem Konfigurationssignals D_{PROGO} des Schiebefensterdetektors 4 ferner ein Ausgangssignal D_{OUT}, welches das zweite Eingangssignal des zweiten Und-Gatters 34 ist, dessen Ausgangssignal einen 1-Bit Flipflop 36 zu jeder steigenden Flanke in den „high"-Zustand versetzt. Das Ausgangssignal des 1-Bit Flipflops 36 ist auch das Ausgangssignal P des jeweiligen Segments 3.

Zu jeder steigenden Flanke des Taktsignals CLK_{PLT} wird ein zweites N-bit SIPO Schieberegister 37 um einen Schritt geschoben, wodurch das aktuelle Ausgangssignal des 1-Bit Flipflops 36 ausgelesen wird. Das letzte Bit des parallelen Ausgangssignals des zweites N-bit SIPO Schieberegisters 37 bildet ein Eingangssignal eines zweiten Oder-Gatters 38. Das andere Eingangssignal des zweiten Oder-Gatters 38 ist das entsprechende 1-Bit Steuersignale I_1-I_N. Wenn eines der beiden Eingangssignale des zweiten Oder-Gatters 38 den "high"-Zustand aufweist, ist auch das Ausgangssignal des zweiten Oder-Gatters 38 „high" und die steigende Flanke schaltet den Zustand des 1-Bit Flipflops 36 zurück sowie setzt hierdurch alle Bits des zweites N-bit SIPO Schieberegisters 37 auf „low", d.h. in den „low"-Zustand.

Fig. 9 zeigt eine schematische Darstellung eines Symbols eines Schaltkreises eines Schiebefensterdetektors 4 der Fig. 10. Fig. 10 zeigt eine schematische Darstellung eines Schaltkreises des Schiebefensterdetektors 4 der Fig. 9.

Der Schiebefensterdetektor 4 dient dazu zu erkennen, ob die Anzahl der „high"-Bits, d.h. der Bits im „high"-Zustand, in seinem Eingangssignal D_{IN} innerhalb der letzten N-bits, d.h. innerhalb eines vorbestimmten Zeitraums, welcher durch das Konfigurationssignal D_{PROG} konfigurierbar ist, des 1-Bit Eingangssignals D_{IN} einen konfigurierbaren Schwellwert übersteigt. Hierzu wird das 1-Bit Eingangssignal D_{IN} des Schiebefensterdetektors 4 einem ersten Und-Gatter 40 als dessen erstes Eingangssignal zugeführt.

Das Ausgangssignal des ersten Und-Gatters 40 wird als Eingangssignal in ein erstes N-bit SIPO Schieberegister 41 geleitet, welches das serielle Eingangssignal parallelisiert und mit jeder steigenden Flanke im Taktsignal CLK_{IN} einen Schritt weitergeschoben wird. Das N-te parallele Ausgangssignal des ersten N-bit SIPO Schieberegisters 41 ist das linksschiebende Eingangssignal SL in ein zweites bidirektionales M-bit SIPO-Schieberegister 42. Das Ausgangssignal des ersten Und-Gatters 40 selbst ist das rechtsschiebende Eingangssignal SR des zweiten bidirektionalen M-bit SIPO-Schieberegisters 42.

Zu jeder steigenden Flanke des Taktsignals CLK_{IN} wird das zweite bidirektionale M-bit SIPO-Schieberegister 42 einen Schritt in die Richtung nach rechts geschoben, falls das rechtsschiebende Eingangssignal SR „high" und das linksschiebende Eingangssignal SL „low" ist. Wenn das rechtsschiebende Eingangssignal SR „low" und das linksschiebende Eingangssignal SL „high" ist, wird das zweite bidirektionale M-bit SIPO-Schieberegister 42 hingegen in die Richtung nach links geschoben. Ansonsten bleibt das zweite bidirektionale M-bit SIPO-Schieberegister 42 unverändert.

Falls das zweite bidirektionale M-bit SIPO-Schieberegister 42 in die Richtung nach rechts geschoben wird, wird ein „high”-Bit von links eingefügt. Falls hingegen das zweite bidirektionale M-bit SIPO-Schieberegister 42 in die Richtung nach links geschoben wird, wird ein „low”-Bit von rechts eingefügt. Das letzte Bit des parallelen Ausgangssignals des zweiten bidirektionalen M-bit SIPO-Schieberegisters 42 wird invertiert als zweites Eingangssignal des ersten Und-Gatters 40 genutzt.

Ein M+1fach Multiplexer 43 wird von K 1-Bit Eingangssignalen konfiguriert und generiert so entweder ein konstantes “high”-Ausgangssignal oder selektiert einen der M parallelen Ausgangssignale des zweiten bidirektionalen M-bit SIPO-Schieberegisters 42. Das selektierte Signal ist das Ausgangssignal des M+1fach Multiplexers 43 und des gesamten Schiebefensterdetektors 4.

Welches der M+1 <= 2^K (M plus 1 kleinergleich 2 hoch K) Eingangssignale mittels des M+1fach Multiplexers 43 ausgesucht wird, wird von dem parallelen Ausgangssignals eines dritten K-bit SIPO-Schieberegisters 44 festgelegt, welches von einem Bitstrom des Eingangssignals D_{PROG} mit einem dazugehörigen Taktsignals CLK_{PROG} betrieben wird. Das letzte parallele Ausgangssignal des dritten K-bit SIPO-Schieberegisters 44 ist der zusätzliche Konfigurationssignal D_{PROGO} als Ausgangssignal des Schiebefensterdetektors 4, um mehrere Segmente 3 bzw. Neuronen 2 in Serie verschalten zu können.

Fig. 11 zeigt eine schematische Darstellung eines Symbols eines Schaltkreises eines Zeitmultiplexers 5 der Fig. 12. Fig. 12 zeigt eine schematische Darstellung eines Schaltkreises des Zeitmultiplexers 5 der Fig. 11.

Der Zeitmultiplexer 5 ist in der Lage, eine Folge von K 1-Bit parallelen EIngangssignalen S_1-S_K in ein serielles 1-Bit Ausgangssignal O zu enkodieren. Die steigende Flanke eines der K Eingangssignale S_1-S_K setzt ein korrespondierendes Flipflop 50 einer Anzahl K von identischen und parallel zueinander angeordneten Flipflops 50 in den „high”-Zustand. Die Ausgangssignale der Flipflops 50 sind jeweils eines der beiden Eingangssignale eines jeweils korrespondierenden Und-Gatters 51 einer Anzahl K von identischen und parallel zueinander angeordneten Und-Gattern 51.

Zu jeder steigenden Flanke des Taktsignals CLK_{SPIKE} wird ein selbst initialisierter K-bit Ringzähler 52 weitergeschoben, dessen parallele Ausgangssignale jeweils das zweite Eingangssignal der Und-Gatter 51 sowie das zurücksetzende Signal, d.h. das Reset-Signal, für die Flipflops 50 sind. Zu jedem Zeitpunkt ist genau ein Bit des Ringzählers 52 im Zustand „high” während alle anderen Bit des Ringzählers 52 im Zustand „low” sind. Zur fallenden Flanke des Reset-Signals wird das jeweilige Flipflop 50 in den „low”-Zustand geschaltet.

Während beide Eingangssignale eines der Und-Gatter 51 im Zustand „high” sind, ist auch das Ausgangssignal dieses Und-Gatters 51 im Zustand „hoch”, ansonsten im Zustand „low”. Wenn eines der K Und-

Gatter 50 ein Ausgangssignal im Zustand „hoch“ hat, ist das Ausgangssignal eines Oder-Gatters 53 ebenfalls im Zustand „high”, sonst im Zustand „low”. Zur steigenden Flanke des Taktsignals CLK_{SPIKE} wird das Ausgangssignal des Oder-Gatters 53 für einen Taktzyklus in einem D-Flipflop 54 zwischengespeichert. Das Ausgangssignal des D-Flipflops 54 ist das Ausgangssignal des Zeitmultiplexers 5.

Das Taktsignal CLK_{PROG} ist an den Datenstrom des Konfigurationssignals D_{PROG} zur Konfiguration der Segmente 3 gekoppelt und hat lediglich die Funktion, die Segmente 3 innerhalb des jeweiligen Neurons 2 sowie die Neuronen 2 innerhalb der Population 1 untereinander zu synchronisieren.

Das Taktsignal CLK_{SPIKE} steuert die Verarbeitung von sog. „Spikes“, d.h. von Pulsen als „high“-Zustände. Zum einen wird mit der Frequenz des Taktsignales CLK_{SPIKE} das Ausgangssignal der Population 1 in der Zeit multiplexed. Zum anderen wird das Taktsignal CLK_{SPIKE} im zweiten bidirektionalen M-bit SIPO-Schieberegister 42 des Schiebefensterdetektors 4 genutzt, um dieses Eingangssignal synchronisiert zu verarbeiten. Somit ist der ausgehende Datenstrom des zweiten bidirektionalen M-bit SIPO-Schieberegisters 42 an das Taktsignal CLK_{SPIKE} gebunden. Auch hängen alle eingehenden Datenströme des Eingangssignals E, des Kontrollsignals I sowie der binären Zufallszahlensignal M, welche zur Mustererkennung dienen, an dem Taktsignal CLK_{SPIKE}.

In der Verarbeitung des Eingangssignals E, des Kontrollsignals I sowie der binären Zufallszahlensignal M zur Mustererkennung gilt insbesondere, dass das Zeitfenster des Schiebefensterdetektors 4 $N * 1/f(CLK_{SPIKE})$ ist, also durch den Horizont des n bidirektionalen M-bit SIPO-Schieberegister 42 des Schiebefensterdetektors 4 und durch die Frequenz des Taktsignals gegeben ist. In der Anwendung lässt sich durch die Wahl der Frequenz die Population 1 auf die Zeitskalen anpassen, auf denen Teilmuster erkannt werden sollen, wobei ein Teilmuster das ist, was ein Segment 3 alleine durch den Schiebefensterdetektor 4 erkennt.

Das Taktsignal CLK_{PLT} steuert ausschließlich die Länge der Plateaus in den einzelnen Segmente 3, also die Zeitdauer bzw. Signallänge, für die ein einzelnes Segment 3 sich die Erkennung eines Teilmusters zusammen mit ausreichendem Signal aus dem binären Baum merkt: Cache für das Zwischenergebnis. Im Speziellen wird das asynchron geschaltete „high“-Ausgangssignal des Segments 3 nach mindestens $N * 1/f(CLK_{PLT})$ und nach maximal $(N+1) * 1/f(CLK_{PLT})$ wieder ausgeschaltet. Die Spanne ergibt sich dadurch, dass das Zählen im Schiebefensterregister 4 zum Ausschalten vom Anschalten des Ausgangssignals entkoppelt ist. Damit lässt sich über die Wahl von N die zeitliche Präzision auf Kosten von Bauteilen und über die gemeinsame Wahl von N und der Frequenz des Taktsignals CLK_{PLT} die Zeitskala regeln, auf der Zwischenergebnisse und Teilmuster gespeichert werden. Das Taktsignal CLK_{PLT} stellt somit eine zweite Zeitskala in der Mustererkennung dar.

Die Kombination der Taktsignale CLK_{SPIKE} und CLK_{PLT}, um Teilmuster auf zwei unabhängig wählbaren Zeitskalen zur Mustererkennung zu kombinieren, stellt eine Besonderheit der Neuronen 2 dar. Isoliert kontrolliert jedes Taktsignal CLK_{SPIKE} und CLK_{PLT} wie bisher üblich einen Teil des Neurons 2 über Flankensteuerung. Genauer betrachtet werden jedoch erfindungsgemäß die Segmente 3 innerhalb des Neurons 2 von den verschiedenen Taktsignalen CLK_{SPIKE} und CLK_{PLT} gesteuert und dies zur Implementierung von Algorithmen zur Mustererkennung verwendet.

Die zuvor beschriebene Population 1 kann dazu verwendet werden, mit niedriger Latenz Muster in kontinuierlichen, digitalen Datenströmen (Bitstreams) zu erkennen. Da aufgrund von Störsignalen oder zeitlicher Impräzision niemals dieselben Muster in gleicher Form auftreten, können dabei auch ungefähre Übereinstimmungen erkannt und der Grad der Übereinstimmung quantifiziert werden. Dabei sind die zu erkennenden Muster konfigurierbar, d.h. können vorbestimmt werden.

Hierzu werden die zuvor beschriebenen Neuronen 2 als mehrere Musterdetektoren in Gruppen in Form von Populationen 1 zusammengefasst. Jedes einzelne Neuron 2 ist hier eine hierarchische Struktur der Segmente 3, welche untereinander verknüpft sind und, je nach problemspezifischer Konfiguration, jeweils eigene Eingangssignale verarbeiten. Wenn ein komplexes Muster als Eingangssignal alle Segmente 3 in der richtigen zeitlichen Sequenz aktiviert, erzeugt das jeweilige Neuron 2 in seinem Ausgangssignal ein positives Bit, d.h. ein Ausgangssignal mit dem Zustand „high“; sozusagen „feuert“ das Neuron 2 bzw. das Neuron 2 erzeugt einen Pulse bzw. einen „Spike“.

Die Wahrscheinlichkeit, mit der ein einzelnes Neuron 2 feuert, reflektiert dabei den Grad der Übereinstimmung zwischen dem geforderten, d.h. dem konfigurierten vorbestimmten, und dem gesehenen, d.h. der Population 1 zugeführten, Muster. In einer Population 1 lesen alle K gleichkonfigurierten Neuronen 2 den gleichen Datenstrom als Eingangssignal E und versuchen, das gleiche Muster in dem Eingangssignal E zu erkennen, erhalten jedoch durch eine pseudo-zufällige Maskierung der Eingangssignale E mit den binären Zufallszahlensignalen M stochastisch voneinander verschiedene Eingangssignale E. Dies bedeutet, dass auf jedes Muster W von K Neuronen 2 reagieren, wobei W den Grad der Übereinstimmung zwischen dem zugeführten Muster und dem konfigurierten vorbestimmten Muster abbildet. Die technische Umsetzung kommt hierbei gänzlich ohne Mikroprozessoren aus und ist gänzlich in den zuvor beschriebenen Schaltkreisen umsetzbar.

Hierzu wird der eingehende Datenstrom E, in welchem ein Muster erkannt werden sollen, als getaktetes binäres Signal E in Form von N 1-Bit Eingangssignalen E_1-E_N auf mehreren parallelen Leitungen gelegt und der Population 1 zugeführt. Dies gilt ebenso für das Kontrollsignal I und die binären Zufallszahlensignale M.

Innerhalb der Population 1 werden auf der Eingangsseite die gleichen Eingangssignale E und Kontrollsignale I an jedes Neuron 2 geleitet, wo die Eingangssignale E mit den neuronenspezifischen binären Zufallssignalen M maskiert werden. Die einzelnen Ergebnisse der Neuronen 2 werden dann im Zeitmultiplexer 5 zusammengeführt, um einen einzelnen Datenstrom O als Ausgang der Population 1 zu generieren, welcher wie gefordert die Qualität des erkannten Musters in den eingehenden Datenstrom E widerspiegelt.

Jedem einzelnen Neuron 2 innerhalb Population 1 kommt dabei die Aufgabe zu, das konfigurierbare, vorbestimmte Muster im jeweiligen 1-Bit Eingangssignal E_1-E_N der N 1-Bit Eingangssignale E_1-E_N zu erkennen. Hierzu sind die Neuronen 2 jeweils aus den einzelnen N Segmenten 3 aufgebaut, von denen jedes eines der eingehenden N 1-Bit Eingangssignale E_1-E_N verarbeitet. Jedes Segment 3 reagiert dabei auf ein relevantes Signal in seinem zugeordneten 1-Bit Eingangssignal E_1-E_N, d.h. das k-te Segment 3 auf ein relevantes Signal im 1-Bit Eingangssignal E_k, indem das Segment 3 für eine bestimmte Zeit eingeschaltet, d.h. in den „high“-Zustand versetzt, wird. Untereinander sind diese Segmente 3 in dem binären Baum derart verschaltet, dass jedes einzelne Segment 3 nur dann durch das jeweilige 1-Bit Eingangssignal E_1-E_N eingeschaltet werden kann, wenn – je nach Konfiguration – Null, Eins oder Zwei der untergeordneten Binärbaumzweige 21, 22 oder Abschlusszweige 20 im binären Baum bereits eingeschaltet sind. Wie lange ein Segment 3 eingeschaltet ist, wird durch das Taktsignal CLK_{PLT} festgelegt, welches nicht an das Taktsignal CLK_{SPIKE} des Eingangssignals E gekoppelt ist.

In jedem Neuron 2 ist diese Verschachtelung in dem Binärbaum abgebildet. Ein Neuron 2 hat für jedes Segment 3 ein jeweils zugeordnetes 1-Bit Eingangssignal E_1-E_N der N 1-Bit Eingangssignale E_1-E_N, mit welchem das jeweilige Segment 3 für eine feste Zeit eingeschaltet, d.h. in den Zustand „high“ gebracht, werden kann (Plateau). Jedes Segment 3 hat ebenso ein jeweils zugeordnetes 1-Bit Kontrollsignal I_1-I_N der N 1-Bit Kontrollsignale I_1-I_N, mit welchem das Segment 3, falls es bereits in den Zustand „high“ ist, durch das jeweils zugeordnete 1-Bit Kontrollsignal I_1-I_N als externes Signal wieder ausgeschaltet, d.h. in den „low“-Zustand gebracht, werden kann. Einzelne Segmente 3 bekommen das konfigurierbare Taktsignal CLK_{PLT}, welches die zeitliche Dauer bestimmt, für die ein Segment 3 eingeschaltet ist. Das Segment 3 an der Wurzel der Baumstruktur, d.h. in der obersten Ebene es binären Baums, generiert kurze Pulse, auch Spikes genannt, mit derselben Taktung des Taktsignals CLK_{SPIKE} wie die Eingangssignale E anstatt längere Plateaus zu erzeugen, wie in den übrigen Segmenten 3.

Jedes der Segmente 3 wird zunächst durch die Eingangssignale B1, B2 anderer im binären Baum untergeordneter Segmente 3 getrieben, sofern diese existieren. Hier kann konfiguriert werden, ob Null, Eins oder Zwei Segmente 3 eingeschaltet sein müssen. Weiter wird der eingehende Datenstrom E in dem Schiebefensterdetektor 4 verarbeitet, welches für kurze Zeit eingeschaltet ist, falls die Anzahl der gesetzten Bits in einem festen Zeitfenster einen kritischen Pegel überschreitet. Das Eingangssignal in den

Schiebefensterdetektor 4 wird vorher mit dem binäres Zufallszahlensignal M maskiert. So ist die Antwort jedes Schiebefensterdetektors 4 auf dasselbe Eingangssignal E stochastisch und unterscheidet sich, wie oben beschrieben, von anderen Schiebefensterdetektoren 4 in der Population 1, welche auf das gleiche Eingangssignal E abweichend reagieren.

Falls sowohl das jeweils zugeordnete 1-Bit Eingangssignal E_1-E_N das konfigurierte Kriterium der untergeordneten Binärbaumzweige 21, 22 bzw. Abschlusszweige 20 erfüllt und der Schiebefensterdetektor 4 des Segments 3 im zugeordneten 1-Bit Eingangssignal E_1-E_N ein Signal erkannt hat, schaltet sich das Segment 3 ein. In dem zweiten bidirektionalen M-bit SIPO-Schieberegister 42 wird dieser Zustand für eine feste Anzahl an Takten des Taktsignals CLK_{PLT} gehalten, worauf sich das Segment 3 selbst wieder ausschaltet. Weiter kann das jeweils zugeordnete 1-Bit Kontrollsignal I_1-I_N den zweiten bidirektionalen M-bit SIPO-Schieberegister 42 zurücksetzen und das Segment 3 frühzeitig ausschalten. Intern ist das Ausgangssignal des Segments 3 lediglich indirekt über den Schiebefensterdetektor 4 an ein Taktsignals CLK_{SPIKE} gebunden. Das Segment 3 reagiert ansonsten mit einer zu vernachlässigen Verzögerung der einzelnen Bauteile.

Dem Schiebefensterdetektor 4 kommt dabei die Aufgabe zu, zu detektieren, falls die Anzahl der gesetzten Bits, d.h. der Bits im „high“-Zustand, in einem festen vorbestimmten Zeitfenster des getakteten Datenstroms einen kritischen Wert überschreitet. Hierzu wird seitens des Schiebefensterdetektors 4 pro Segment 3 die Anzahl der eingehenden Pulse, d.h. der Bits im Zustand „high“, in dem festen Zeitfenster in dem jeweiligen Datenstrom D_{IN} gezählt. Jedes eingehende gesetzte Bit, d.h. Bit im Zustand „high“, schiebt das zweite bidirektionale M-bit SIPO-Schieberegister 42 vorwärts und wird gleichzeitig in einer durch das zweite bidirektionale M-bit SIPO-Schieberegister 42 und durch das Taktsignal CLK_{SPIKE} des Datenstrom E implementierten Delayline gespeichert. Nach diesem Delay wird das zweite bidirektionale M-bit SIPO-Schieberegister 42 wieder zurückgeschoben.

Konfigurierbar ist, an welcher Stelle im zweiten bidirektionalen M-bit SIPO-Schieberegister 42 ein Bit gesetzt sein muss, um ein Ausgangssignal zu erzeugen. So signalisiert das zweite bidirektionale M-bit SIPO-Schieberegister 42, wann mehr als ein konfigurierbarer, vorbestimmter Schwellwert 1-Bit-Signale in dem durch das Taktsignal CLK_{SPIKE} und durch die Länge der Delayline festgelegten Zeitfenster im Datenstrom D_{IN} zu finden waren.

Der Schiebefensterdetektor 4 dekodiert somit 1-Bit Signale, welche genau der Kodierung des Ausgangssignals der Population 1 entsprechen. Die Anzahl der Pulse in kurzer Zeit kodiert dabei die Stärke des Signals. Der Schwellwert im zweiten bidirektionalen M-bit SIPO-Schieberegister 42 legt fest, wann ein Signal stark genug war.

Das Konfigurationssignal D_{PROG}, welches sich durch alle Bauteile der Population 1 zieht, ermöglicht die Konfiguration der Population 1 und aller enthaltener Bauteile.

Im Vergleich zu bekannten Lösungen zur digitalen Signalverarbeitung und Mustererkennung, besonders im Bereich neuromorpher Technologien, ergeben sich eine Reihe von Vorteilen aus den zuvor beschriebenen Neuronen 2 sowie der hieraus aufgebauten Population 1.

So ermöglicht der zuvor beschriebene Ansatz die Erkennung von konfigurierbaren Mustern auf verschiedenen Zeitskalen und ist somit tolerant gegenüber Störungen im Signal oder im Timing. Dies erlaubt den Einsatz in erschwerten Bedingungen, z.B. im Verbund mit unpräziser Sensorik oder mit Signalen mit hoher Variabilität.

Auch kann durch die Nutzung stochastischer Eingangssignale als die binären pseudo-zufälligen Zufallszahlensignale M nicht nur ein gegebenes Muster erkannt werden, sondern es kann auch der Grad der Übereinstimmung quantifiziert werden.

Sowohl das Eingangssignal E als auch das Ausgangssignal O der Population 1 sind kompatibel, um mit weiteren Populationen zu kommunizieren, und erlauben somit die Verschaltung zu großen Netzen.

Die Informationsverarbeitung erfolgt gänzlich ohne den Einsatz von Mikroprozessoren oder Paket-Routing, was technisch einfacher umsetzbar ist, ein hohes Maß an Parallelisierung ermöglicht und zu niedrigen Latenzen führt.

Die Kommunikation zwischen Populationen 1 und das An- bzw. Ausschalten von Segmenten 3 ist an zwei verschiedene Taktsignale, nämlich die Taktsignale CLK_{PLT} und CLK_{SPIKE}, gebunden. Hierdurch wird das Taktsignal CLK_{SPIKE}, auf welchem Muster im Datenstrom E als Eingangssignal E erkannt werden sollen, von dem Taktsignal CLK_{PLT} entkoppelt. Auf diese Art und Weise können im bestimmungsgemäßen Gebrauch im Datenstrom E Teilmuster erkannt werden, welche auf einer von dem Datenstrom E entkoppelten Zeitskala, nämlich dem Taktsignal CLK_{SPIKE}, mit anderen Teilmustern des Datenstroms E verbunden werden.

Zum Beispiel können viele Pulse im Datenstrom E in sehr kurzer Zeit übertragen werden und auf ein wichtiges Ereignis wie zum Beispiel das Überschreiten eines kritischen Wertes eines Temperatursensors hinweisen. Ein zweites Ereignis wie zum Beispiel das Überschreiten eines kritischen Wertes eines Beschleunigungssensors kann ebenfalls schnell mittels des Taktsignals CLK_{SPIKE} übertragen werden. Beide Ereignisse können als Teil eines Muster „kritische Temperatur und dann kritische Beschleunigung“ dann aber auf einer Zeitskala, welche von dem Taktsignal CLK_{SPIKE} entkoppelt und durch das Taktsignal CLK_{PLT} zum Beispiel deutlich langsamer definiert ist, kombiniert werden. Durch die Kombination beider Taktsi-

gnale CLK_{SPIKE}, CLK_{PLT} kann ein Segment 3 auf die speziellen externen Timing-Anforderrungen der Anwendung angepasst werden.

Werden die Neuronen 2 mit einer höheren Komplexität in Form eines binären Baums mit zahlreichen Ebenen umgesetzt, so können mehr Informationen im internen Zustand der Neuronen 2 verarbeitet und gespeichert werden. Daher sind für dieselbe Leistung weniger individuelle Neuronen 2 erforderlich, was die Größe der resultierenden Population 1 und damit die Komplexität der notwendigen Kommunikationsinfrastruktur deutlich reduzieren kann.

Weitere Ausgestaltungen der Erfindung, welche von dem betrachteten Ausführungsbeispiel abweichen, sind vorstellbar. Jedes einzelne der oben genannten Bauteile kann in seinem Funktionsumfang erweitert oder in der Umsetzung angepasst werden. Auch können mehrere Populationen 1 zu Netzen verschaltet werden, die eingesetzt werden könnten, um komplexere Probleme zu lösen.

BEZUGSZEICHENLISTE (Teil der Beschreibung)

B_1	Eingangssignal eines Segments 3 seitens eines ersten Binärbaumzweigs 21
B_2	Eingangssignal eines Segments 3 seitens eines zweiten Binärbaumzweigs 22
CLK_{PLT}	Taktsignal zur Steuerung der Länge der Plateaus der Vergleichsschaltungen 3
CLK_{SPIKE}	Taktsignal zur Steuerung der Verarbeitung der Spikes der Vergleichsschaltungen 3
CLK_{PROG}	Taktsignal des Konfigurationssignals D_{PROG}, D_{PROGO}
D_{IN}	Eingangssignal eines Schiebefensterdetektors 4
D_{OUT}	Ausgangssignal eines Schiebefensterdetektors 4
D_{PROG}	Konfigurationssignal als Eingangssignal
D_{PROGO}	Konfigurationssignal als Ausgangssignal
E, E_1-E_N	Eingangssignal der Population 1; eingehender Datenstrom
i	Zählindex
I, I_1-I_N	Kontrollsignal
J	Anzahl der Eingangssignale des Abschlusszweigs 20
K	Anzahl der Neuronen
M, $M_{1,1}$-$M_{K,N}$	1-Bit bzw. binäres Zufallszahlensignale
N	Anzahl der Segmente
O	Ausgangssignal des Zeitmultiplexers 5 bzw. der Population 1
P_1-P_k	1-Bit Ausgangssignale der Neuronen 2, der Abschlusszweige 20, der Binärbaumzweige 21, 22 und der Segmente 3
S_1-S_K	Eingangssignale des Zeitmultiplexers 5
SL	linksschiebendes Eingangssignal des zweiten bidirektionalen (M-bit SIPO-) Schieberegisters 42 des Schiebefensterdetektors 4
SR	rechtsschiebendes Eingangssignal des zweiten bidirektionalen (M-bit SIPO-) Schieberegisters 42 des Schiebefensterdetektors 4
W	Grad der Übereinstimmung zwischen zugeführtem Muster und konfigurierten vorbestimmten Muster
1	neuromorphe Schaltkreisanordnung; Population
2	neuromorpher Musterdetektor; Neuron; Neuronen-Schaltkreis; Binärbaumwurzel
20	Abschlusszweig; Binärbaumblatt; Terminal Branch
21	erster Binärbaumzweig; erster innerer Knoten des Binärbaums; erster Nested Branch
22	zweiter Binärbaumzweig; zweiter innerer Knoten des Binärbaums; zweiter Nested Branch

3	Vergleichsschaltung; Segment
30	erstes Oder-Gatter
31	erstes Und-Gatter
32	(4-fach) Multiplexer
33	erstes (2-bit SIPO-) Schieberegister
34	zweites Und-Gatter
35	drittes Und-Gatter
36	(1-Bit) Flipflop
37	zweites (N-bit SIPO-) Schieberegister
38	zweites Oder-Gatter

4	Schiebefensterdetektor; Slider
40	erstes Und-Gatter
41	erstes (N-bit SIPO-) Schieberegister
42	zweites bidirektionales (M-bit SIPO-) Schieberegister
43	((M+1)-fach) Multiplexer
44	drittes (K-bit SIPO-) Schieberegister

5	Zeitmultiplexer; Time Multiplexer
50	Flipflops
51	Und-Gatter
52	selbst initialisierter (K-bit) Ringzähler
53	Oder-Gatter
54	D-Flipflop

PATENTANSPRÜCHE

1. Neuromorpher Musterdetektor (2),

welcher ausgebildet ist, wenigstens zwei 1-Bit Eingangssignale (E_1-E_N) eines zu erkennenden Musters zu erhalten,

mit wenigstens zwei Vergleichsschaltungen (3), welche jeweils ausgebildet sind,

eines der 1-Bit Eingangssignale (E_1-E_N) zu erhalten,

die Anzahl der „high"-Zustände oder der „low"-Zustände des jeweiligen 1-Bit Eingangssignals (E_1-E_N) innerhalb eines vorbestimmten Zeitraums zu zählen,

die Anzahl der gezählten Zustände mit einem vorbestimmten Schwellwert der jeweiligen Vergleichsschaltung (3) zu vergleichen und

bei Überschreiten des Schwellwerts auf die erfolgte Erkennung des zu erkennenden Musters hinzuweisen.

2. Neuromorpher Musterdetektor (2) nach Anspruch 1, **dadurch gekennzeichnet, dass**

die eine Vergleichsschaltung (3) der anderen Vergleichsschaltung (3) erstrangig untergeordnet ist,

wobei die übergeordnete Vergleichsschaltung (3) ausgebildet ist, nur dann auf die erfolgte Erkennung des zu erkennenden Musters hinzuweisen, falls der Schwellwert der übergeordneten Vergleichsschaltung (3) überschritten und zeitgleich von der erstrangig untergeordneten Vergleichsschaltung (3) auf die erfolgte Erkennung des zu erkennenden Musters hingewiesen wird.

3. Neuromorpher Musterdetektor (2) nach Anspruch 2, **gekennzeichnet durch**

wenigstens eine weitere Vergleichsschaltung (3), welche parallel zu der untergeordneten Vergleichsschaltung (3) angeordnet ist,

wobei die übergeordnete Vergleichsschaltung (3) ausgebildet ist, nur dann auf die erfolgte Erkennung des zu erkennenden Musters hinzuweisen, falls der Schwellwert der übergeordneten Vergleichsschaltung (3) überschritten und zeitgleich von den erstrangig untergeordneten Vergleichsschaltungen (3) jeweils auf die erfolgte Erkennung des zu erkennenden Musters hingewiesen wird.

4. Neuromorpher Musterdetektor (2) nach einem der Ansprüche 2 oder 3, **gekennzeichnet durch**

wenigstens eine weitere Vergleichsschaltung (3), welche zweitrangig untergeordnet zu der erstrangig untergeordneten Vergleichsschaltung (3) angeordnet ist,

wobei die erstrangig untergeordnete Vergleichsschaltung (3) ausgebildet ist, nur dann auf die erfolgte Erkennung des zu erkennenden Musters hinzuweisen, falls der Schwellwert der erstrangig untergeordneten Vergleichsschaltung (3) überschritten und zeitgleich von der zweitrangig untergeordneten Vergleichsschaltung (3) auf die erfolgte Erkennung des zu erkennenden Musters hingewiesen wird.

5. Neuromorpher Musterdetektor (2) nach Anspruch 3 oder 4, **dadurch gekennzeichnet, dass**

die wenigstens drei Vergleichsschaltungen (3) einen Binärbaum mit wenigstens zwei Ebenen bilden.

6. Neuromorpher Musterdetektor (2) nach einem der vorangehenden Ansprüche, **dadurch gekennzeichnet, dass**

die Vergleichsschaltungen (3) identisch ausgebildet sind.

7. Neuromorpher Musterdetektor (2) nach einem der vorangehenden Ansprüche, **dadurch gekennzeichnet, dass**

bei Überschreiten des Schwellwerts ein 1-Bit Ausgangssignal (P_1-P_k) der jeweiligen Vergleichsschaltung (3) auf den „high“-Zustand, ansonsten auf den „low“-Zustand, gesetzt wird, oder umgekehrt.

8. Neuromorpher Musterdetektor (2) nach Anspruch 7, **dadurch gekennzeichnet, dass**

die Vergleichsschaltungen (3) ausgebildet sind, jeweils ein 1-Bit Steuersignal (I_1-I_N) zu erhalten und in Reaktion auf einen „high“-Zustand oder auf einen „low“-Zustand des jeweiligen 1-Bit Steuersignals (I_1-I_N) das 1-Bit Ausgangssignal (P_1-P_k) der jeweiligen Vergleichsschaltung (3) auf den „low“-Zustand zu setzen.

9. Neuromorpher Musterdetektor (2) nach einem der vorangehenden Ansprüche, **dadurch gekennzeichnet, dass**

der vorbestimmte Schwellwert der Anzahl der Zustände der jeweiligen Vergleichsschaltung (3) vorgibt, wann das zu erkennende Muster als erkannt angesehen wird.

10. Neuromorpher Musterdetektor (2) nach einem der vorangehenden Ansprüche, **dadurch gekennzeichnet, dass**

die Vergleichsschaltungen (3) jeweils einen Schiebefensterdetektor (4) aufweisen, welcher jeweils ausgebildet ist, das jeweilige 1-Bit Eingangssignal (E_1-E_N) zu erhalten und die Anzahl der „high"-Zustände oder der „low"-Zustände des jeweiligen 1-Bit Eingangssignals (E_1-E_N) innerhalb des vorbestimmten Zeitraums zu zählen.

11. Neuromorpher Musterdetektor (2) nach Anspruch 10, **dadurch gekennzeichnet, dass**

das Zählen der Anzahl der „high"-Zustände oder der „low"-Zustände des jeweiligen 1-Bit Eingangssignals (E_1-E_N) innerhalb des vorbestimmten Zeitraums mittels eines bidirektionalen Schieberegisters (42) des jeweiligen Schiebefensterdetektors (4) erfolgt.

12. Neuromorpher Musterdetektor (2) nach einem der vorangehenden Ansprüche, **dadurch gekennzeichnet, dass**

die Vergleichsschaltungen (3), vorzugsweise deren Schiebefensterdetektor 4, jeweils ein Taktsignal CLK_{SPIKE} zur Steuerung der Verarbeitung der Pulse und ein Taktsignal CLK_{PLT} zur Steuerung der Länge der Plateaus erhalten,

wobei die beiden Taktsignale CLK_{SPIKE} und CLK_{PLT} unterschiedlich sind.

13. Neuromorphe Schaltkreisanordnung (1)

mit einer Mehrzahl von neuromorphen Musterdetektoren (2) nach einem der vorangehenden Ansprüche,

wobei jeder neuromorphe Musterdetektor (2) ausgebildet ist,

das gleiche 1-Bit Eingangssignal (E_1-E_N) zu erhalten,

ein unterschiedliches 1-Bit Zufallszahlensignal ($M_{1,1}$-$M_{K,N}$) zu erhalten,

das jeweilige 1-Bit Eingangssignal (E_1-E_N) mit dem entsprechenden 1-Bit Zufallszahlensignal ($M_{1,1}$-$M_{K,N}$) zu verändern, und

die Anzahl der „high"-Zustände oder der „low"-Zustände des jeweiligen veränderten 1-Bit Eingangssignals (E_1-E_N) innerhalb eines vorbestimmten Zeitraums zu zählen.

14. Neuromorphe Schaltkreisanordnung (1) nach Anspruch 13, **dadurch gekennzeichnet, dass**

wenigstens eine Vergleichsschaltung (3), vorzugsweise alle Vergleichsschaltungen (3) jeweils, ein Und-Gatter (35) aufweist, welches ausgebildet ist, das jeweilige 1-Bit Eingangssignal (E_1-E_N) und das entsprechende 1-Bit Zufallszahlensignal ($M_{1,1}$-$M_{K,N}$) zu kombinieren.

15. Neuromorphe Schaltkreisanordnung (1) nach Anspruch 13 oder 14, **dadurch gekennzeichnet, dass**

die neuromorphe Schaltkreisanordnung (1) ausgebildet ist,

die Anzahl der 1-Bit Ausgangssignale (P_1-P_k) der jeweiligen Vergleichsschaltung (3), welche zeitgleich im „high“-Zustand oder im „low“-Zustand sind, zu erfassen und

aus dem Verhältnis der Anzahl von 1-Bit Ausgangssignalen (P_1-P_k) im „high“-Zustand oder im „low“-Zustand und der Anzahl der neuromorphen Musterdetektoren (2) einen Grad (W) der Übereinstimmung zwischen 1-Bit Eingangssignal (E_1-E_N) und zu erkennendem Muster zu bestimmen.

16. Neuromorphe Schaltkreisanordnung (1) nach einem der Ansprüche 13 bis 15, **dadurch gekennzeichnet, dass**

wenigstens eine Vergleichsschaltung (3), vorzugsweise alle Vergleichsschaltungen (3) jeweils, einen Zeitmultiplexer (5) aufweist, welcher ausgebildet ist, parallele Ausgangssignale (P_1-P_K) der neuromorphen Musterdetektoren (2) zu einem 1-Bit-Ausgangssignal (O) der neuromorphen Schaltkreisanordnung (1) zusammenzuführen.

ZUSAMMENFASSUNG

Die vorliegende Erfindung betrifft einen neuromorphen Musterdetektor (2), welcher ausgebildet ist, wenigstens zwei 1-Bit Eingangssignale (E_1-E_N) eines zu erkennenden Musters zu erhalten, mit wenigstens zwei Vergleichsschaltungen (3), welche jeweils ausgebildet sind, eines der 1-Bit Eingangssignale (E_1-E_N) zu erhalten, die Anzahl der „high“-Zustände oder der „low“-Zustände des jeweiligen 1-Bit Eingangssignals (E_1-E_N) innerhalb eines vorbestimmten Zeitraums zu zählen, die Anzahl der gezählten Zustände mit einem vorbestimmten Schwellwert der jeweiligen Vergleichsschaltung (3) zu vergleichen und bei Überschreiten des Schwellwerts auf die erfolgte Erkennung des zu erkennenden Musters hinzuweisen.

(Figur 4)

PLOS ONE

RESEARCH ARTICLE

A Bayesian Monte Carlo approach for predicting the spread of infectious diseases

Olivera Stojanović[1☯]*, **Johannes Leugering**[1☯], **Gordon Pipa**[1], **Stéphane Ghozzi**[2], **Alexander Ullrich**[2]*

1 Department of Neuroinformatics, Institute of Cognitive Science, Osnabrück University, Osnabrück, Germany, **2** Department of Infectious Diseases, Robert Koch Institute, Berlin, Germany

☯ These authors contributed equally to this work.
* ostojanovic@uos.de (OS); ullricha@rki.de (AU)

Citation: Stojanović O, Leugering J, Pipa G, Ghozzi S, Ullrich A (2019) A Bayesian Monte Carlo approach for predicting the spread of infectious diseases. PLoS ONE 14(12): e0225838. https://doi.org/10.1371/journal.pone.0225838

Editor: Cathy W.S. Chen, Feng Chia University, TAIWAN

Received: April 29, 2019

Accepted: November 13, 2019

Published: December 18, 2019

Data Availability Statement: All data and code used in this work is available at the public code repository at: https://github.com/ostojanovic/BSTIM.

Funding: The author(s) received no specific funding for this work.

Competing interests: The authors have declared that no competing interests exist.

Abstract

In this paper, a simple yet interpretable, probabilistic model is proposed for the prediction of reported case counts of infectious diseases. A spatio-temporal kernel is derived from training data to capture the typical interaction effects of reported infections across time and space, which provides insight into the dynamics of the spread of infectious diseases. Testing the model on a one-week-ahead prediction task for campylobacteriosis and rotavirus infections across Germany, as well as Lyme borreliosis across the federal state of Bavaria, shows that the proposed model performs on-par with the state-of-the-art *hhh4* model. However, it provides a full posterior distribution over parameters in addition to model predictions, which aides in the assessment of the model. The employed Bayesian Monte Carlo regression framework is easily extensible and allows for incorporating prior domain knowledge, which makes it suitable for use on limited, yet complex datasets as often encountered in epidemiology.

Introduction

Public-health agencies have the responsibility to *detect*, *prevent* and *control* infections in the population. In Germany, the Robert Koch Institute collects a wide range of factors, such as location, age, gender, pathogen, and further specifics, of laboratory confirmed cases for approximately 80 infectious diseases through a mandatory surveillance system [1]. Since 2015, an automated outbreak detection system, using an established aberration detection algorithm [2], has been set in place to help *detect* outbreaks [3, 4]. However, *prevention* and *control* require quantitative *prediction* instead of mere *detection* of anomalies and thus prove more challenging. For logistical, computational and privacy reasons, epidemiological data is typically reported or provided in bulk, often grouped by calendar weeks and counties. Predictions thus have to be made about the number of cases per time-interval and region, based on a history of such measurements.

Since outbreaks can extend over multiple counties, states or even nations, spatio-temporal models are typically employed. Some approaches use scan statistics to identify anomalous spatial or spatio-temporal clusters [5, 6], while others model and predict case counts as time series

or point processes [7, 8]. A major advantage of such predictive models is the additional insight they can provide into the factors contributing to the spread of infectious diseases.

In general, we distinguish four qualitatively different classes of predictive features: *spatial*, *temporal*, *spatio-temporal* and *(spatio-temporal) interaction* effects. The former three are purely functions of space, time or both, modeling *seasonal* fluctuations and *trends*, *geographical* influences or localized time-varying effects, such as *region-specific demographics* or *legislation*, respectively. The latter is an autoregressive variable that captures how an observed infection influences the number of further infections in its neighborhood over time, which depends on differences in *patients' behavior*, *transmission vectors*, *incubation times* and *duration* of the respective diseases. Even in the absence of direct contagion, previously reported cases can provide valuable *indirect* information for predicting future cases through latent variables. The effect on the expected number of cases at a given place and time due to interactions can thus be expressed as a (unknown) function of spatial and temporal distance to previously reported cases. Particularly for regions with less available historic data or those strongly influenced by their neighbors, e.g. smaller counties close to larger cities [9], incorporating the county's and its neighbors' recent history of case counts can improve predictions.

The state-of-the-art spatio-temporal *hhh4* method [7, 10] assumes aggregated case counts to follow a Poisson or Negative Binomial distribution around a mean value determined by "epidemic" and "endemic" components. The epidemic component can capture the influence of previous cases from the same or neighboring counties, e.g. potentially weighted by the counties' adjacency order, while the endemic component models the expected baseline rate of cases.

For *not aggregated* data, the more general *twinstim* method [7] models the interaction effects due to individual cases by a self-exciting point process with predefined continuous spatio-temporal kernel, rather than through a binary neighborhood relation as in the *hhh4* model. Optimizing such a kernel for a specific dataset provides an opportunity to incorporate or even infer information about the infectious spread of the disease at hand. Using such smooth spatial kernel functions in favor of e.g. neighborhood graphs between geographical regions has the additional benefit, that it can also be applied in domains where the shape and neighborhood relation between such regions is complex. For example within Germany counties can contain enclaves, e.g. cities that represent a county of their own, or even be composed of disjoint parts.

In the following, we present a Bayesian spatio-temporal interaction model (referred to as BSTIM), as a synthesis of both approaches: a probabilistic generalized linear model (GLM) [11] predicts aggregated case counts within spatial regions (counties) and time intervals (calendar weeks) using a history of reported cases, temporal features (seasonality and trend) and region-specific as well as demographic information. Like for the *twinstim* method, interaction effects are modeled by a continuous spatio-temporal kernel, albeit parameterized with parameters inferred from data. Since the aggregated reporting of case counts per calendar week and county leaves residual uncertainty about the precise time and location of an individual case, we model times within the respective week and locations within the respective county as latent random variables. Monte Carlo methods are employed to evaluate posterior distributions of parameters as well as predictions, which are subsequently used to assess the quality of the model.

For three different infectious diseases, *campylobacteriosis*, *rotaviral enteritis* and *Lyme borreliosis*, the interpretability of the inferred components, specifically the interaction effect kernel, is discussed and the predictive performance is evaluated and compared to the *hhh4* method.

Materials and methods

We evaluate both the proposed BSTIM as well as the *hhh4* reference model on a one-week-ahead prediction task, where the number of cases in each county is to be predicted for a specific week, given the previous history of cases in the respective as well as surrounding counties. Instead of point estimates, we are interested in a full posterior probability distribution over possible case counts for each county and calendar week—capturing both aleatoric uncertainty due to the stochastic nature of epidemic diseases as well as epistemic uncertainty due to limited available training data. The data for this study is provided by the Robert Koch Institute, and consists of weekly reports of case counts for three diseases, campylobacteriosis, rotavirus infections and Lyme borreliosis. They are aggregated by county and collected over a time period spanning from the 1st of January 2011 (2013 for borreliosis) to the 31st of December 2017 via the *SurvNet* surveillance system [1]. We use the term "county" to generally refer to rural counties (*Landkreise*) and cities (*kreisfreie Städte*) as well as the twelve districts of Berlin (*Bezirke*). Aggregated case counts of diseases with mandatory reporting in Germany can be downloaded from https://survstat.rki.de. For each of the three diseases, the data preceding 2016 is used for training the model, while the remaining two years are used for testing. A software implementation of the BSTI Model presented here is available online at https://github.com/ostojanovic/BSTIM.

The BSTI Model

The proposed model is optimized to predict the number of reported cases in the future (e.g. the next week), based on prior case counts. Since epidemiological count data is often overdispersed relative to a Poisson distribution [12], i.e. the variance exceeds the mean, we assume counts are distributed as a Negative Binomial random variable around an expected value $\mu(t, x)$ that varies with time (t) and space (x), and with a scale parameter $\alpha \geq 0$. Due to its common use in combinatorics, the Negative Binomial distribution is often formalized in terms of parameters r, representing the number of failures in a hypothetical repeated coin flip experiment, and p, representing the success probability in each trial. This can be trivially extended to real valued coefficients, and reparameterized in terms of μ and α by setting $\mu \to {}^{pr}/_{1-p}$ and $\alpha \to {}^{1}/_{r}$. The Negative Binomial distribution has been successfully used in epidemiology [12–14], since its variance $\mathbb{V} = \mu + \alpha\mu^2$ allows to model overdispersion in the data for $\alpha > 0$, while including the Poisson distribution as a special case for $\alpha \to 0$.

We further assume that the relationship between each feature $f_i(t, x)$ and the expected value $\mu(t, x)$ can be expressed in a generalized linear model of the Negative Binomial random variable $Y(t, x)$ using the canonical logarithmic link function. A half-Cauchy distribution is used as a weakly informative prior [15] to enforce positivity of the dispersion parameter of the residual Negative Binomial distribution. For all other parameters, Gaussian priors with zero mean and standard deviation 10 are chosen. Since the linear predictor of the generalized linear model combines qualitatively different types of data, specifically interaction effects and exogenous features such as temporal or demographical information, we employ sensitivity analysis to verify that the chosen (relative) scales for the priors do not unduly influence the inferred parameters. To this end, we systematically vary the standard deviation of the prior distribution for the interaction effect coefficients over the values 0.625, 2.5, 10, 40 and 160. Since we only observe negligible changes in the posterior parameter distributions (see S4 Fig through S6 Fig) and resulting predictions (not shown here) for standard deviations 10 and above, we conclude that the chosen Normal distribution with standard deviation 10 constitutes an adequate weekly informative prior. The full probabilistic model for training can thus

be summarized as follows:

$$\alpha \sim \text{HalfCauchy}(\gamma = 2) \tag{1}$$

$$W_i \sim \text{Normal}(\mu = 0, \sigma = 10) \tag{2}$$

$$\mu(t,x) = \exp\left(\sum_{i=1}^{N} W_i f_i(t,x)\right) \cdot \epsilon(t,x) \tag{3}$$

$$Y(t,x) \sim \text{NegBin}(\mu(t,x), \alpha) \tag{4}$$

where:
α is a dispersion parameter
N is the total number of used features
W_i are model weights
$f_i(t, x)$ are features varying in time and space
$\epsilon(t, x)$ is the exposure varying in time and space
t refers to a time-interval (i.e. one calendar week)
x refers to a spatial region (i.e. one county)

For prediction, the priors over the dispersion parameter and weights are replaced by the corresponding posterior distribution inferred on the training set.

A schema of our model is shown in Fig 1. To capture the interaction effects between different places over time, a continuous spatio-temporal kernel is estimated through a linear combination of 16 basis kernels. The individual contribution due to each of these basis kernels is included into the model as a feature. Four temporal periodic *basis functions* are used to capture seasonality and five sigmoid *basis functions* (one for each year of available training data) to capture temporal trends. Four region-specific features (ratio of population in a county belonging to three age groups and one political component) are used, which results in 29 features. In addition, the logarithm of the population of each county in the respective year is used as a scaling parameter (exposure) ϵ.

For example, given one parameter sample $w = [w_1, \ldots, w_{29}]$, inferred from the training set of campylobacteriosis case counts, the conditional mean prediction within county x during

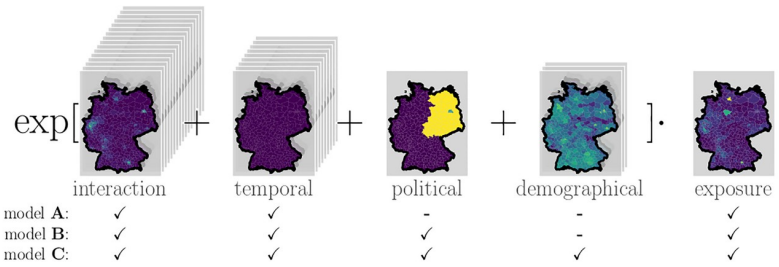


Fig 1. Model scheme. Exemplary contributions from different features, grouped into interaction, temporal, political and demographical components, each evaluated in all counties in Germany for campylobacteriosis in the week 30 of 2016. Each county's total population is always included as an exposure coefficient. We consider three models of increasing complexity, A, B and C, that differ in whether features are included (✓) or not (-). Information about the shape of counties within Germany is publicly provided by the German federal agency for cartography and geodesy (Bundesamt für Kartographie und Geodäsie) (GeoBasis-DE / BKG 2018) under the dl-de/by-2-0 license.

https://doi.org/10.1371/journal.pone.0225838.g001

week t is determined as follows:

$$\mu(t,x) = \exp\left(\underbrace{\sum_{i=1}^{16} w_i f_i(t,x)}_{\text{interaction}} + \underbrace{\sum_{i=17}^{20} w_i f_i(t)}_{\text{periodic}} + \underbrace{\sum_{i=21}^{25} w_i f_i(t)}_{\text{trend}} + \underbrace{\sum_{i=26}^{29} w_i f_i(t,x)}_{\text{region-specific}}\right) \cdot \underbrace{\epsilon(t,x)}_{\text{exposure}} \tag{5}$$

Monte Carlo sampling procedure

The model described above determines the posterior distribution over parameters by the data-dependent likelihood and the choice of priors. We want to capture this parameter distribution in a fully Bayesian manner, rather than summarize it by its moments (ie. mean, covariance, etc.) or other statistics. Since an analytic solution is intractable, we use Markov Chain Monte Carlo (MCMC) methods to generate unbiased samples of this posterior distribution. These samples can be used for evaluation of performance measures (here deviance and Dawid-Sebastiani score; cf. section *Predictive performance evaluation and model selection*), visualization or as input for a superordinate probabilistic model.

Our model combines features that can be directly observed (e.g. demographic information) with features that can only be estimated (e.g. interaction effects, due to uncertainty caused by data aggregation). To integrate the latter into the model, we generate samples from the distribution of interaction effects features as outlined in section *Interaction effects*.

The sampling procedure generates samples from the *prior* distribution over parameters and combines them with training data and our previously generated samples of the interaction effect features to produce samples of the *posterior* parameter distribution. These samples from the inferred joint distribution over *parameters* are then used to generate samples of the posterior distribution of model *predictions* for testing data.

We employ a Hamiltonian Monte Carlo method, No-U-Turn-Sampling [16], implemented in the probabilistic programming package *pyMC3* [17]. To evaluate proper convergence of the sampling distribution to the desired (but unknown) posterior distribution, four independent Markov chains are generated and their marginal distributions compared using the Gelman-Rubin diagnostic $\hat{R}$ [18], which assesses the relation between the within-chain and the between-chains variance.

Interaction effects

Each reported case provides valuable information about the expected number of cases to come in the near future and close proximity. We suppose that this effect of an individual reported infection on the rate of future (reported) infections in the direct neighborhood can be captured by some unknown function $\kappa(d_{\text{time}}(t_\star, t_k), d_{\text{geo}}(x_\star, x_k))$, which we refer to as *interaction effect kernel* in the following, where (t_k, x_k) refer to the time and location of the k-th reported case and $(t_\star, x_\star)$ represent the time and location of a hypothetical future case. Here, $d_{\text{geo}}(x, y)$ represents the geographical distance between two locations x and y, whereas $d_{\text{time}}(t, s)$ denotes the time difference between two time points t and s. Thus, $\kappa(\cdot, \cdot)$ is a radial, time- and location-invariant kernel, depending only on the spatial and temporal proximity of the two (hypothetical) cases. For the sake of simplicity, we assume that interaction effects due to individual infections add up linearly.

Since κ is not known a-priori for each disease, we wish to infer it from data. To this end, we approximate it by a linear combination of spatio-temporal basis kernels $\kappa_{i,j}$ with coefficients w_i

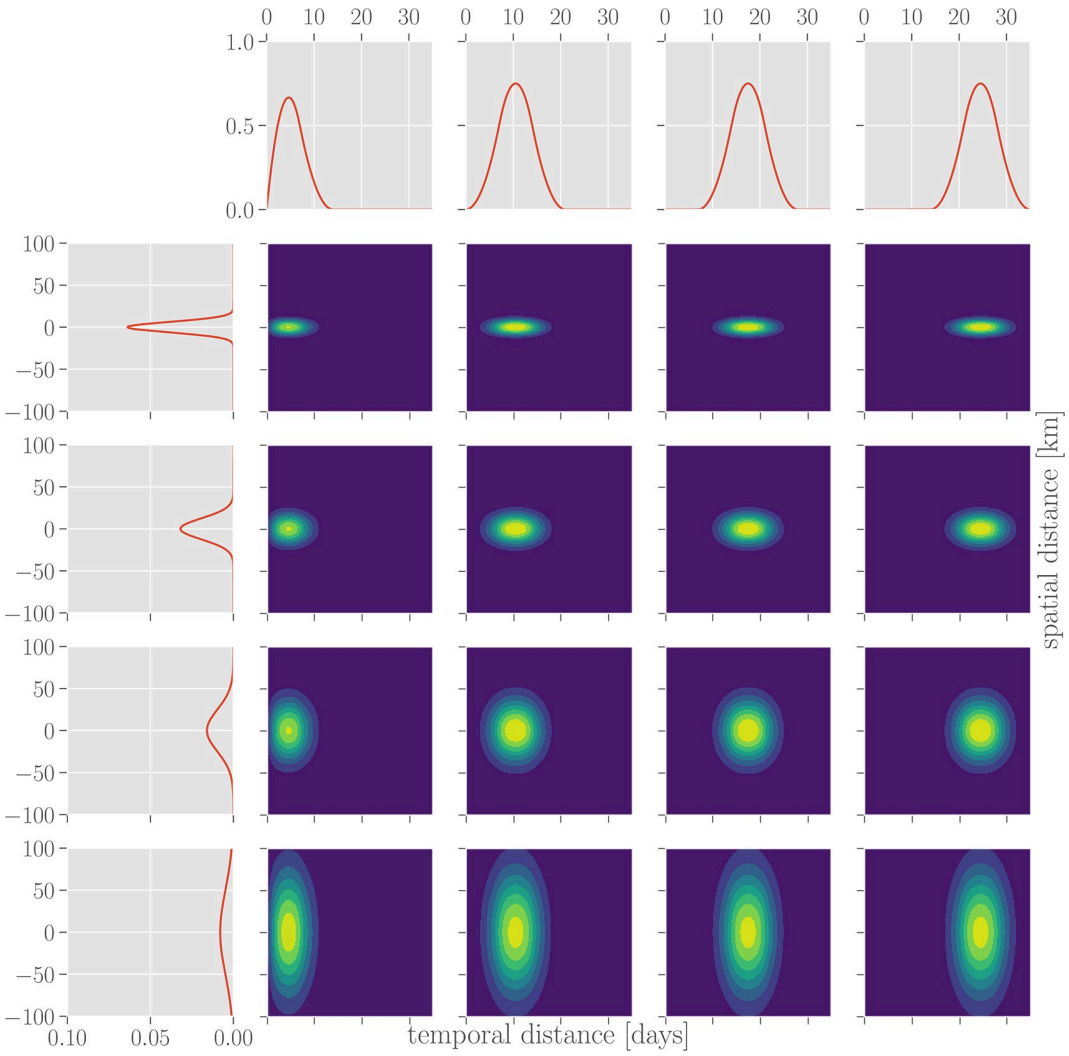


Fig 2. Spatial and temporal basis functions for interaction kernel. The inferred interaction kernel is composed of a linear combination of spatio-temporal basis functions (four-by-four grid of contour plots), each of which is a product of one spatial (left column) and one temporal factor (top row).

https://doi.org/10.1371/journal.pone.0225838.g002

that can be inferred from training data:

$$\kappa(\triangle t, \triangle x) \approx \quad \hat{\kappa}(\triangle t, \triangle x) \coloneqq \sum_i w_i \kappa_{I_i J_i}(\triangle t, \triangle x) \qquad (6)$$
$$\text{where} \qquad I_i \coloneqq \lceil {}^{i}/_{4} \rceil, \quad J_i \coloneqq (i-1) \bmod 4 + 1$$

As the basis functions for the interaction effect kernel, we choose the products $\kappa_{i,j}(\triangle t, \triangle x) \coloneqq \kappa_i^T(\triangle t) \cdot \kappa_j^S(\triangle x)$ between one temporal (κ_i^T) and one spatial factor (κ_j^S), each (cf. Fig 2). As temporal factors, we use the third order B-spline basis functions $\kappa_i^T = N_{i,3}$ for $i = \{1, 2, 3, 4\}$ as defined in [19], with the knot vector [0, 0, 7, 14, 21, 28, 35] (measured in days). The multiplicity 2 of the first knot enforces $\kappa_1^T(0) = 0$. This results in four smooth unimodal functions, spanning the overlapping time interval from zero to two weeks, zero to three weeks, one to four weeks and two to five weeks after a reported case, respectively.

Outside these intervals, the functions are identically zero. Acausal effects (i.e. the influence of a reported case on hypothetical other cases reported at an earlier time) as well as effects more than five weeks after a reported case are thus excluded. This accounts for the typical incubation times for campylobacteriosis [20] and rotavirus infections [21], and early symptoms of Lyme Borreliosis [22], as well as potential reporting delays. As spatial factors, we use exponentiated quadratic kernels (i.e. univariate Gaussian functions) centered at a distance of 0km to a reported case, with shape parameters σ of 6.25km, 12.5km, 25.0km, and 50.0km. These spatial kernels are wide enough to cover the typical daily commuting distances within Germany, which amount to 25km or less for the majority of commuters [23], while being narrow enough to capture only local effects. See Fig 2 for an illustration of how the basis functions $\kappa_{i,j}$ are constructed.

Since the contributions of individual cases are assumed to sum up linearly, the total influence of all cases that were previously reported at times and places (t_k, x_k), $k \in 1 \ldots n$ onto the expected rate of cases reported at a later time t and location x is given by:

$$\sum_{i=1}^{16} w_i f_i(t,x) \quad \text{where}$$
$$f_i(t,x) := \sum_{k=1}^{n} \kappa_{I_i,J_i}(d_{\text{time}}(t,t_k), d_{\text{geo}}(x,x_k)) \tag{7}$$

Each $f_i(t, x)$ for $i \in \{1, \ldots, 16\}$ is a spatio-temporal function that depends on all cases reported prior to t, providing us with a total of 16 features for modeling interaction effects. By determining the corresponding coefficients w_i, the fitting procedure thus allows us to infer an interaction effect kernel $\hat{\kappa}$ in a 16-dimensional parameterized family from data. It should be noted, however, that since the basis functions $\kappa_{i,j}$ capture strongly correlated and possibly redundant information, the effective number of degrees of freedom may be well below 16. Since we work with aggregated data at a spatial resolution of counties and a temporal resolution of calendar weeks, the exact time and location of an individual case report, as well as time and location of a hypothetical future case, are conditionally independent random variables given the county and week in which they occur. Because of this epistemic uncertainty, the features $f_i(t, x)$ derived in Eq 7 are thus random variables themselves. To deal with this uncertainty, the *twinstim* model proposed in [7] suggests to replace these features by their expected values, which can be numerically approximated efficiently. Here, instead of using such point-estimates, which might lead the model to underestimate its uncertainty, we want to incorporate the features $f_i(t, x)$ directly into our probabilistic model and thus need to account for their full probability distribution.

While this distribution is intractable to calculate analytically, we can generate unbiased samples from it through rejection sampling: For a case reported in a given calendar week and county, possible sample points of a precise time and location can be independently generated by uniformly drawing times from within the corresponding week and locations from a rectangle containing the county, rejecting points that fall outside the county's boundary. By randomly drawing a sample time and location for each reported case, we can thus generate an unbiased sample of the (unavailable) data prior to aggregation that accurately reflects the uncertainty caused by the aggregation procedure. Using these resulting sample times and locations in place of t_k and x_k in Eq 7 yields unbiased samples of the features $f_i(t, x)$, which are in turn used when generating samples of the model's posterior parameter distribution (cf. section *Monte Carlo sampling procedure*).

It bears repeating that what we refer to as interaction effect features in this paper are thus in fact latent random variables due to the epistemic uncertainty caused by aggregated reporting of infections by counties and calendar weeks.

Additional features

Infection rates vary in time due to natural processes, such as seasons and climate trends, evolution of pathogens and immunization of the population, as well as societal developments such as scientific and technological advancement and medical education. Within Germany these effects may not differ much across space and can thus be included into the model as feature functions $f_i(t)$ that only depend on time. For modeling yearly seasonality, four sinusoidal basis functions (ie. $\sin(2\pi \cdot t \cdot \omega_{\text{yearly}})$, $\sin(4\pi \cdot t \cdot \omega_{\text{yearly}})$, $\cos(2\pi \cdot t \cdot \omega_{\text{yearly}})$, $\cos(4\pi \cdot t \cdot \omega_{\text{yearly}})$) are used as temporal periodic components, where $\omega_{\text{yearly}} = (1\ \text{year})^{-1}$. Slower time-varying effects are subsumed in a general trend modeled by a linear combination of one logistic function (ie. $(1 + \exp(-\frac{t-\tau_i}{2} \cdot \omega_{\text{weekly}}))^{-1}$) centered at the beginning of each year (τ_i) with slope ½ ω_{weekly}, where $\omega_{\text{weekly}} = (1\ \text{week})^{-1}$.

Due to the historical division between eastern and western Germany, and their different developments, some structural differences remain, such as unemployment rate, density of hospitals and doctors, population density, age structure etc. [24, 25] To account for such systematic differences, a political component, which we refer to as the *east/west component* in the following, is introduced which labels all counties that were part of the former German Democratic Republic as 1 and counties that were part of the Federal Republic of Germany as 0. Since Berlin itself was split into two parts, yet todays counties don't accurately reflect this historic division, counties within Berlin are labeled with an intermediate value of 0.5.

Since diseases can affect children and elderly in different ways, yearly demographic information about each county is incorporated into the model. The logarithm of the fraction of population belonging to three age groups (ages $[0-5)$, $[5-20)$ and $[20-65)$) is used. The age group of 65 years and above accounts for the remaining share of the population and thus is a redundant variable with respect to the other three age groups and the total population. The total population of each county acts as a scaling factor for the predicted number of infections.

Predictive performance evaluation and model selection

To evaluate the predictive performance of the model, forecasts of the number of infections are made one calendar week ahead of time for each disease and each county. To determine the relevance of different features, model selection is performed on the training dataset between three models of different complexity [Fig 1]:

model A—includes interaction and temporal (periodic and trend) components,

model B—includes interaction, temporal and political components,

model C—includes interaction, temporal, political and demographic components.

The Widely Applicable Information Criterion [26](WAIC, also referred to as *Watanabe-Akaike information criterion*, is applied to the posterior distribution over parameters and predictions from the training set to determine which combination of features (i.e. model A, B or C) minimizes the generalization error. Similar to the deviance information criterion, WAIC assesses the model's ability to generalize by estimating the out-of-sample expectation, while penalizing a large *effective* number of parameters. This is relevant here since modeling interaction effects introduces multiple features that capture redundant information. However, rather than evaluating the log-posterior at a parameter point-estimate, the WAIC calculates the

empirical mean over the entire posterior distribution, which leads to a better estimate of the out-of-sample expectation [27], and is therefore ideally suited for sampling-based approaches.

Different error measures are applied to evaluate the fit of the predictive distribution for the test set to observations. Deviance of the Negative Binomial distribution (i.e. the expected difference between the log-likelihood of observations and the log-likelihood of the predicted means) is used as a likelihood-based measure and the Dawid-Sebastiani score (a covariance-corrected variant of squared error, cf. [28]) is included as a distribution-agnostic proper scoring rule.

To evaluate the performance of the model presented here as well as an *hhh4* model implementation for reference, we compare the resulting distributions of scores across counties.

The *hhh4* model reference implementation

We use an *hhh4* model for Negative Binomial random variables, implemented in the R package "surveillance" [7], with a mean prediction composed of an epidemic and an endemic component. The epidemic component is a combination of an autoregressive effect (models reproduction of the disease within a certain region) and a neighborhood effect (models transmission from other regions). The endemic component models a baseline rate of cases due to the same features as described above. The reference model is trained and evaluated on the same datasets as the BSTIM.

Results and discussion

Testing models of varying complexity (see Fig 1) reveals that the most complex model (model complexity C, including interaction effects, temporal, political as well as demographical features) generalizes best as measured by WAIC (see Table 1) for all three different tested diseases (campylobacteriosis, rotavirus and borreliosis). For the remainder of this text, we thus focus only on the full model variety C. The posterior parameter distribution inferred from the training data can be analyzed in itself, which provides valuable information about the disease at hand as well as the suitability of the model. Subsequently, it is used to generate one-week-ahead predictions for the test data.

For each model configuration and disease, the sampling procedure is run until a total of 1000 valid samples of the joint posterior distribution have been generated, which each requires approximately four hours of run-time on a conventional desktop machine (utilizing 4 cores of an AMD Ryzen 5 1500x processor). The sampling procedure converges to the same posterior for all independent chains, as can be seen by inspecting the posterior marginal distributions of each parameter in S1 to S3 Figs, which is quantified by the Gelman-Rubin diagnostics shown in S10 Fig.

Table 1. Training set WAIC scores for the three tested diseases and the three levels of model complexity.

model	campylobacteriosis	rotavirus	borreliosis
A	423279.3	349182.37	31359.62
B	420172.1	339143.27	(31359.62)
C	420010.64	338219.46	30643.49

Since for borreliosis the model is trained and evaluated only within the western state of Bavaria, the east/west feature is constantly zero, and the models A and B thus coincide.

https://doi.org/10.1371/journal.pone.0225838.t001

The inferred model

The procedure outlined above produces samples from the posterior parameter distribution, which in turn provides a probability distribution over interaction kernels. Due to the large number of free parameters (16) involved (see Fig 2), the family of parameterized kernels is flexible enough to capture different disease-specific interactions in time and space. Despite the fact that much more complex interaction effect kernels could be learned, the kernels inferred from data appear to factorize into a specific spatial and temporal profile for each disease. The mean interaction kernel for campylobacteriosis (see Fig 3, 1A) shows the furthest spatial influence over up to 75 km, whereas rotavirus (see Fig 3, 2A) and borreliosis (see Fig 3, 3A) are more localized within a radius of up to 25 km. Borreliosis exhibits longer lasting interaction effects, extending up to four weeks. Despite the fact that borreliosis is not contagious between humans, this is consistent with a pseudointeraction effect due to a localized, slowly changing latent variable such as the prevalence of infected ticks or other seasonal factors. The kernel for campylobacteriosis shows a clear drop in the third week after an infection, which might indicate recovery from the disease, but we advise caution against overinterpretion of this negative interaction.

Looking at individual samples from the respective kernel distributions (see Fig 3, rows B and C) reveals a degree of uncertainty over the precise kernel shape for the different diseases:

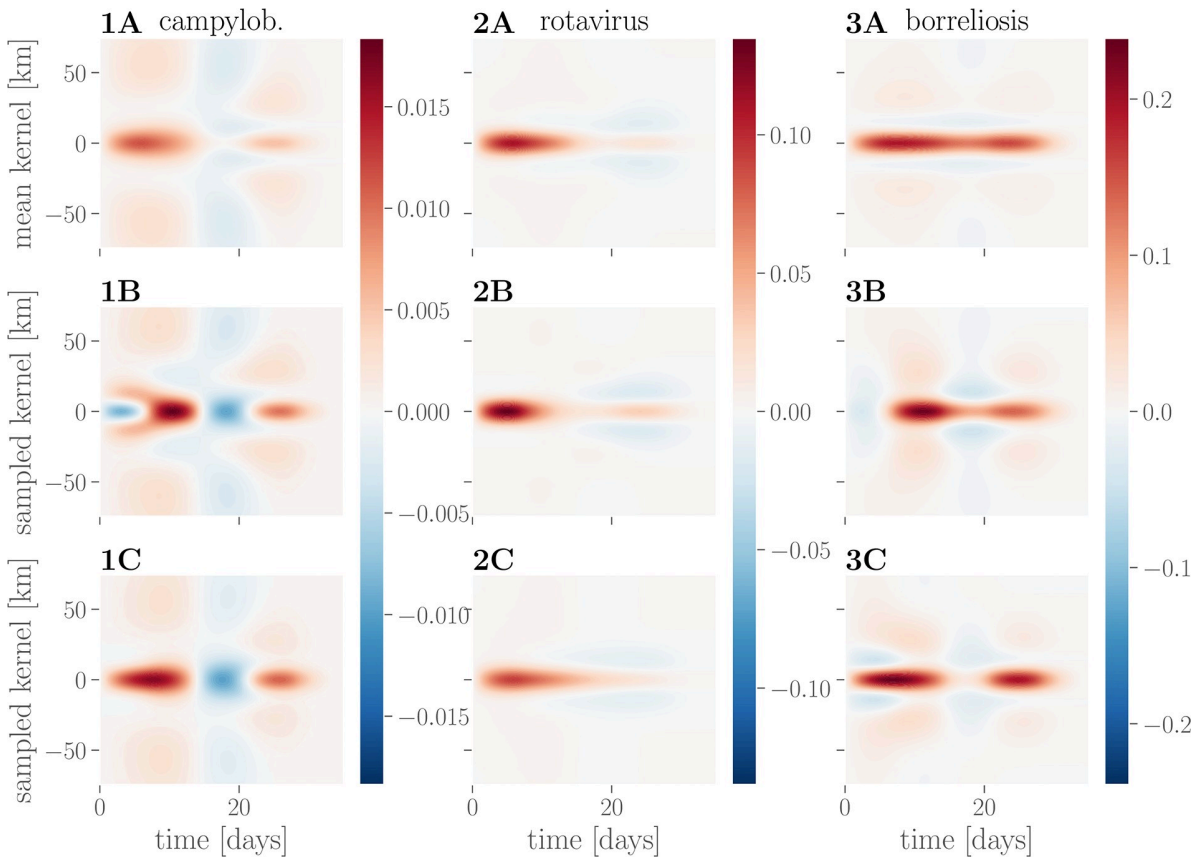


Fig 3. Learned interaction effect kernels. Kernels for campylobacteriosis are shown in **1A-C**, for rotavirus in **2A-C** and for borreliosis in **3A-C**. Mean interaction kernels are shown in the **row A**, while **rows B and C** show two random samples from the inferred posterior distribution over interaction kernels.

https://doi.org/10.1371/journal.pone.0225838.g003

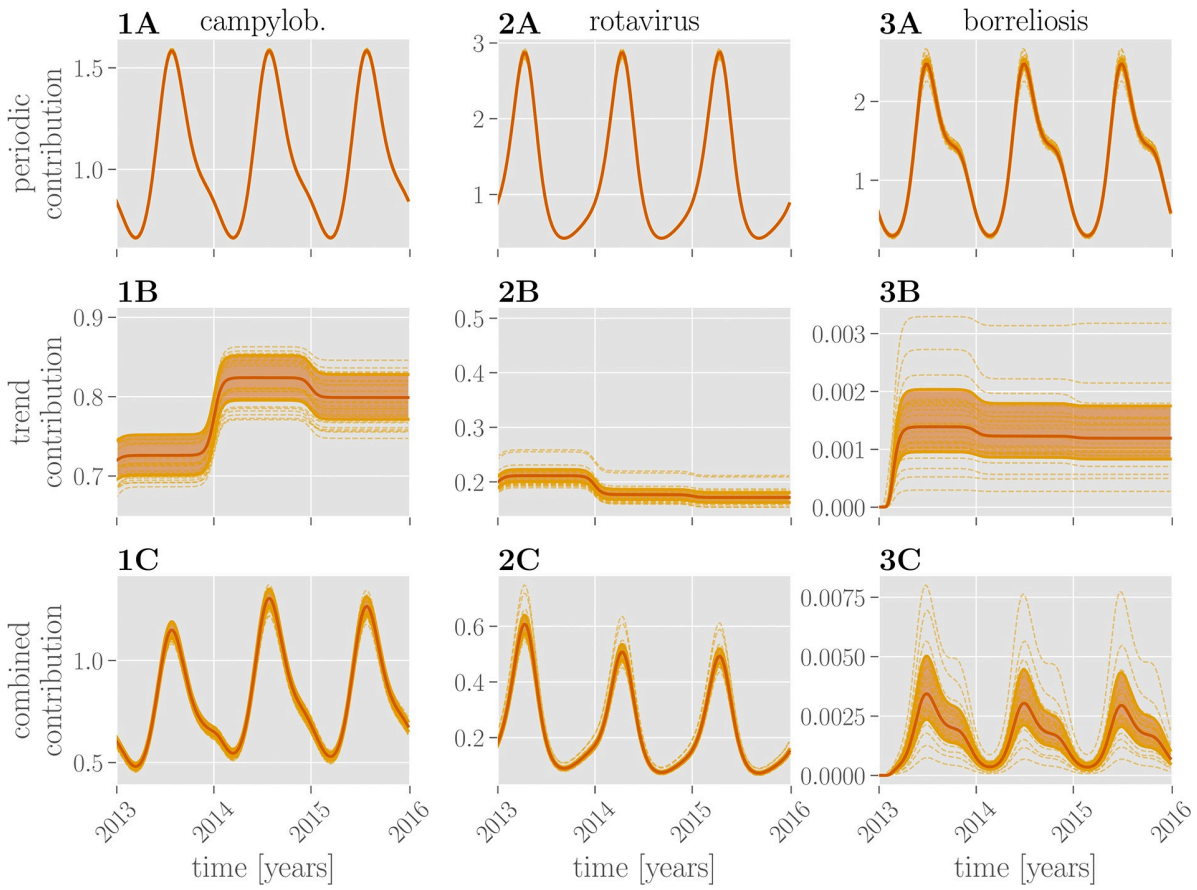


Fig 4. Learned temporal contributions. Periodic contributions over the course of three years (2013-2016) for all three diseases are shown in the **row A**, trend contributions in the **row B** and their combination in the **row C**. Red lines show the mean exponentiated linear combination of periodic or trend or both features through the respective parameters. Dashed lines show random samples thereof; the shaded region marks the 25%-75% quantile.

https://doi.org/10.1371/journal.pone.0225838.g004

while there is little variation in the kernel shape inferred for rotavirus, there is uncertainty about the temporal profile of interactions for campylobacteriosis.

The seasonal components (see Fig 4) for campylobacteriosis and borreliosis show a yearly peak in July and June, respectively. In the case of rotavirus the incidence rate is higher in spring with a peak from March to April. The learned trend components capture the disease-specific baseline rate of infections, which remains stable throughout the years 2013 to 2016. While there is little uncertainty in the seasonal component, there is a high degree of uncertainty in the constant offset of the trend component. The effect of combining both contributions within the model's exponential nonlinearity results in higher uncertainty around larger values.

For campylobacteriosis and, to a lesser extent, rotavirus reported incidence rates are higher in regions formerly belonging to eastern Germany (see Fig 5). The parameters inferred for demographic components (see Fig 5) show the role that age stratification plays for susceptibility. For all three diseases, a larger share of children and adolescents (ages 5-20 years) in the general population is indicative of increased incidence rates. Additionally, working-age adults (ages 20-65 years) appear to increase the incidence rate of borreliosis. It should be noted that this does not necessarily imply an increased susceptibility of the respective groups themselves,

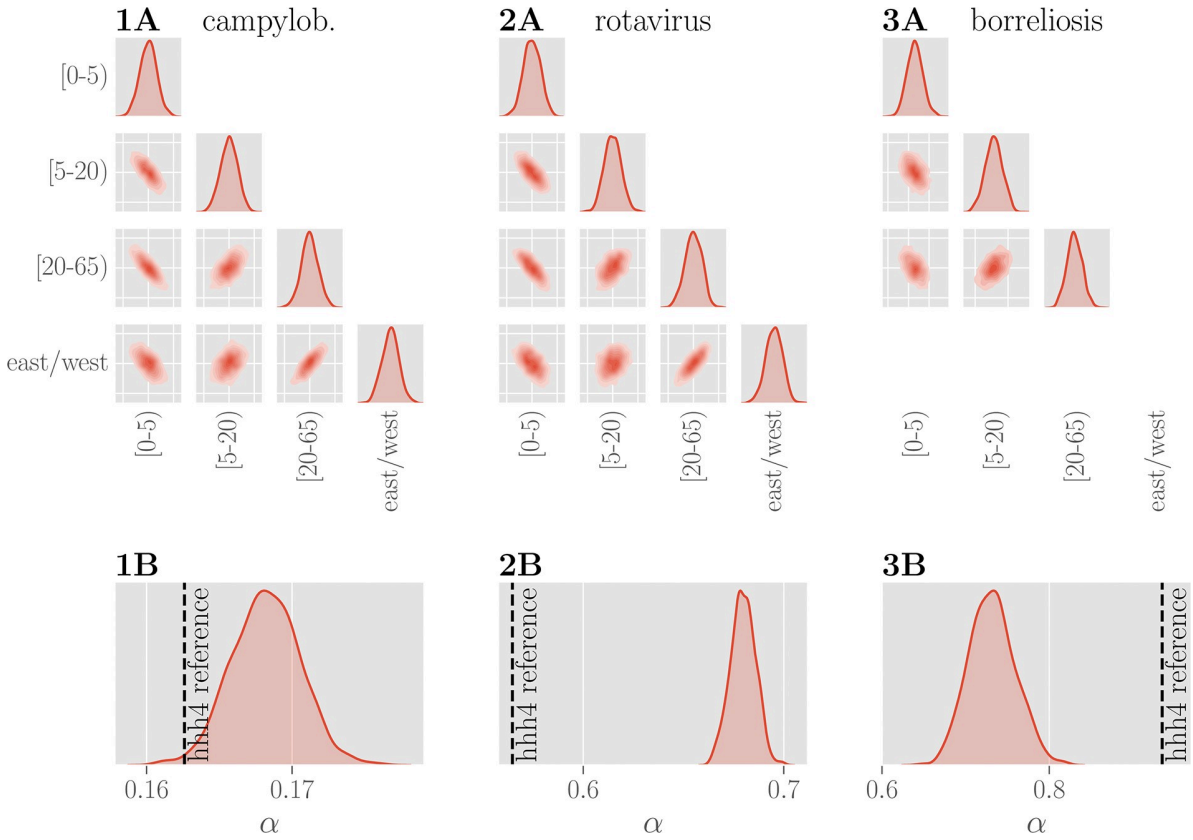


Fig 5. Learned weights for political and demographic components. Plots of the pairwise marginal distributions between inferred coefficients for three age groups and the east/west component for all three diseases are shown in **row A**. The marginal distribution of each coefficient shows a narrow unimodal peak, yet the pairwise distributions show that the individual features are clearly not independent. **Row B** shows the inferred posterior distributions of the overdispersion parameter α for three diseases. Values of α obtained using the *hhh4* reference model are indicated with a dashed black line. The inferred values for the dispersion parameter α are different, yet of similar magnitude, between the two models.

https://doi.org/10.1371/journal.pone.0225838.g005

but could instead be due to latent variables correlated with age stratification, such as economic or cultural differences. The pairwise joint distributions reveal strong (anti-)correlations of the coefficients associated with the demographic and political components. E.g. the coefficient associated with age group [20-65) is strongly correlated with the coefficient associated with the east/west component, which implies ambiguity in the optimal choice of parameters.

The posterior probability over the dispersion parameter α (see Fig 5) is tightly distributed around the respective disease specific means. With small values of α, the distribution of case counts for campylobacteriosis approaches a Poisson distribution, whereas the corresponding distributions for rotavirus and borreliosis are over-dispersed and deviate more from Poisson distributions.

Predictive performance

The one-week-ahead predictions are shown in Fig 6, for two selected cities (Dortmund and Leipzig for campylobacteriosis and rotavirus, Nürnberg (Nuremberg) and München (Munich) for borreliosis), together with the corresponding prediction from the reference *hhh4* model [7] fitted to the same data. A choropleth map of Germany (or the federal state of Bavaria in the case of borreliosis) shows the individual predictions for each county in one calendar week as an example. See also S8, S9 and S10 Figs for predictions for 25 additional counties.

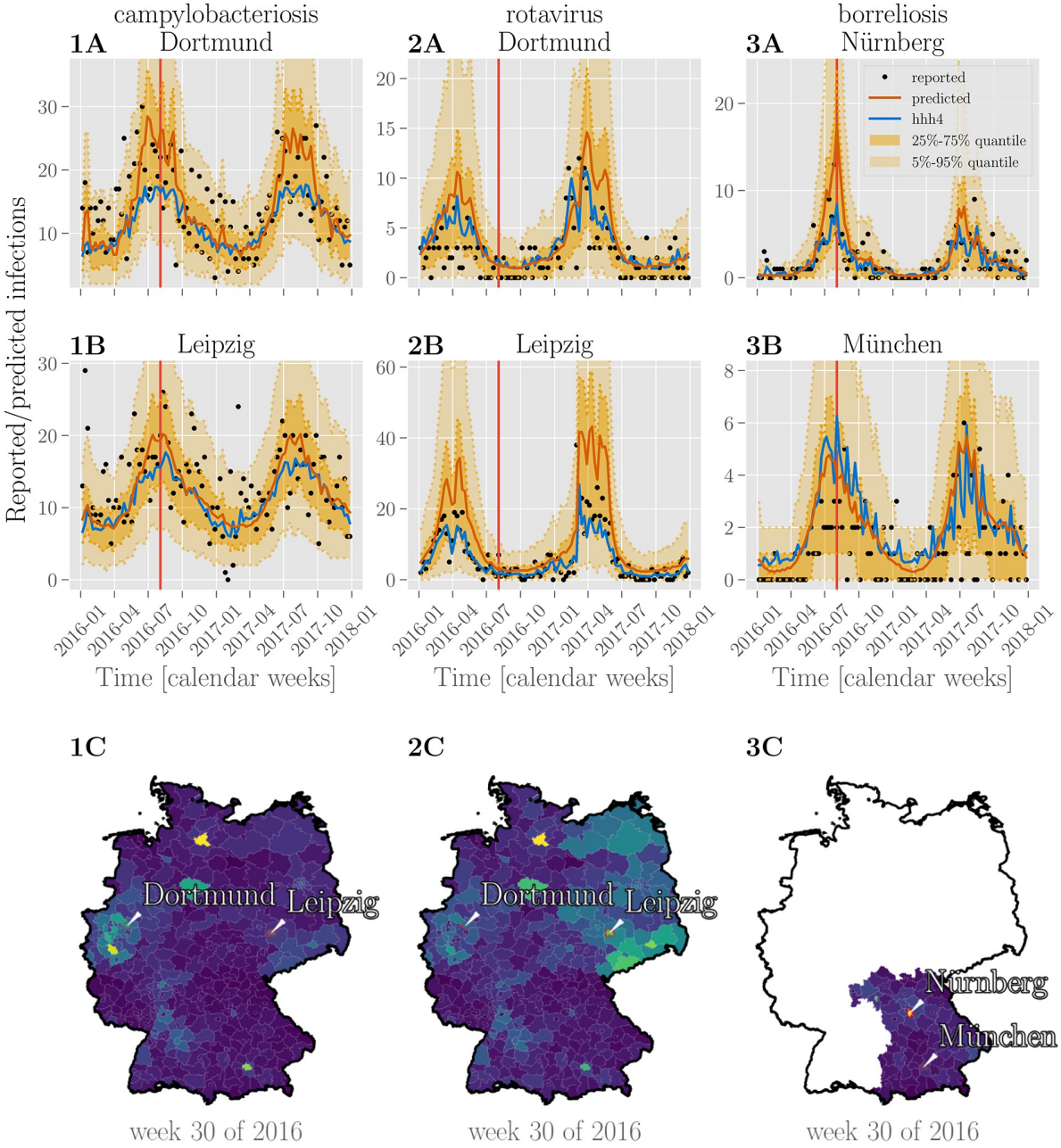


Fig 6. Predictions of case counts for various diseases by county. Reported infections (black dots), predictions of case counts by BSTIM (orange line) and the *hhh4* reference model (blue line) for campylobacteriosis (**column 1**), rotavirus (**column 2**) and borreliosis (**column 3**) for two counties in Germany (for campylobacteriosis and rotavirus) or Bavaria (borreliosis), are shown in **rows A and B**. The shaded areas show the inner 25%-75% and 5%-95% percentile. **Row C** shows predictions of the respective disease for each county in Germany or the federal state of Bavaria in week 30 of 2016 (indicated by a vertical red line in rows A and B). Information about the shape of counties within Germany is publicly provided by the German federal agency for cartography and geodesy (Bundesamt für Kartographie und Geodäsie) (GeoBasis-DE / BKG 2018) under the dl-de/by-2-0 license.

https://doi.org/10.1371/journal.pone.0225838.g006

The BSTIM fits the mean of the underlying distribution of the data well. For rotavirus and borreliosis, it appears to overestimate the dispersion for the cities shown in Fig 6 as indicated by most data points falling within the inner 25%-75% quantile. This may be due to a too high dispersion parameter α (cf. Fig 5) or uncertainty about model parameters. It should be noted,

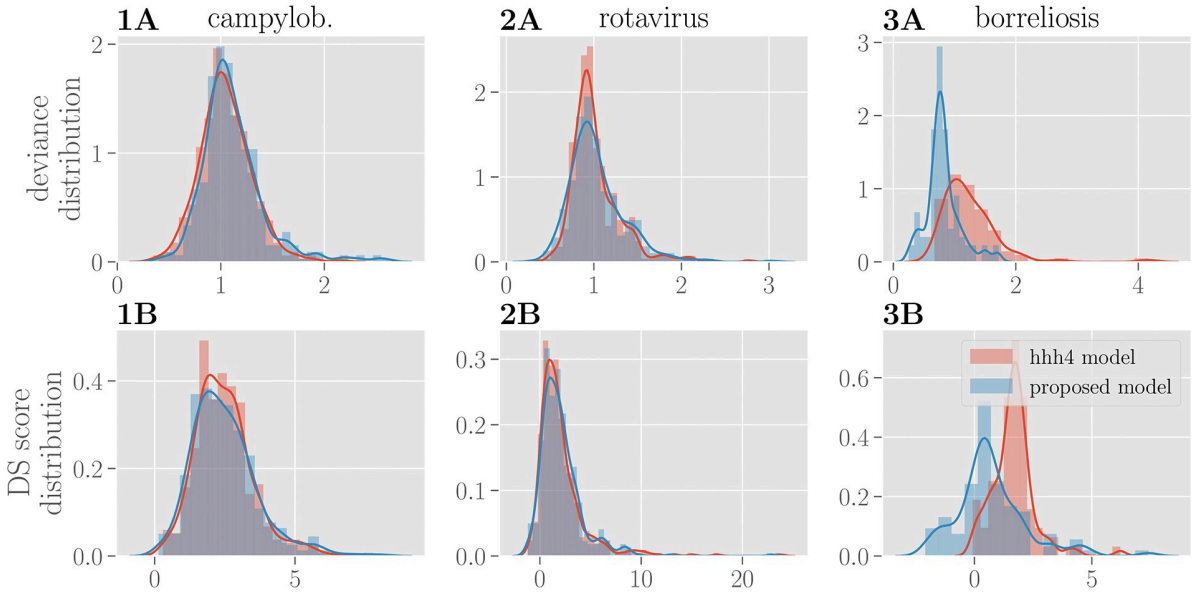


Fig 7. Evaluation of prediction performance. The distribution of deviance over counties is shown in **row A** for BSTIM (blue) and the reference *hhh4* model (red) for all three diseases. The corresponding distribution of Dawid-Sebastiani scores is shown in **row B**.

https://doi.org/10.1371/journal.pone.0225838.g007

however, that the optimal dispersion parameter itself may vary from county to county, whereas our model infers only one single value for all counties together. The resulting predictions for all three diseases are smoother in time and space (cf. the chloropleth maps in Fig 6) than the predictions by the reference *hhh4* model. We attribute this to the smooth temporal basis functions and spatio-temporal interaction kernel of our model.

To quantitatively compare the performance of both models, we calculate the distributions of deviance and Dawid-Sebastiani score over all counties for BSTIM and the *hhh4* reference model as shown in Fig 7. Both measures show a very similar distribution of errors between both models for all three diseases, as it can be seen in Table 2. Only for borreliosis, the *hhh4* model appears to be more sensitive to outliers.

Benefits of probabilistic modeling for epidemiology

Probabilistic modeling relies on the specification of prior probability distributions over parameters [17]. In the context of epidemiology, this makes it possible to incorporate domain knowledge (e.g. we know that case counts tend to be overdispersed relative to Poisson distributions, but not to which degree for a specific disease) as well as modeling assumptions. This is particularly relevant for diseases with limited available data (e.g. those not routinely recorded

Table 2. Deviance and Dawid-Sebastiani score (mean ± standard deviation) for all three diseases and both BSTIM and the *hhh4* model.

disease	score	BSTIM	*hhh4*
campylob.	deviance	1.11 ± 0.3	1.11 ± 0.26
	DS score	2.49 ± 1.17	2.47 ± 1.06
rotavirus	deviance	1.03 ± 0.32	1.04 ± 0.3
	DS score	2.08 ± 2.17	2.1 ± 2.54
borreliosis	deviance	0.81 ± 0.27	0.85 ± 0.27
	DS score	0.74 ± 1.54	1.63 ± 2.24

https://doi.org/10.1371/journal.pone.0225838.t002

through surveillance), where appropriately chosen priors are required to prevent overfitting. The framework can easily be extended to include additional features or latent variables. For example, we introduce precise locations and times of individual cases as latent variables, given only the counties and calendar weeks in which they occurred.

Probabilistic models as discussed here provide samples of the posterior distribution of parameters as well as model predictions. This allows for analysis that is not possible with point estimation techniques such as maximum likelihood estimation. In epidemiology, datasets can be small, noisy or collected with low spatial or temporal resolution. This can lead to ambiguity, where the observations could be equally well attributed to different features and thus different model parameterizations are plausible. While maximum likelihood estimation in such a situation selects only the single most likely model, Bayesian modeling captures the full distribution over possible parameters and predictions, and thus preserves information about the uncertainty associated with the parameters of the model itself. Analyzing the parameter distribution can thus help identify redundant or uninformative features. For example, an inspection of the posterior marginal distributions of the model parameters in S1 Fig shows, that e.g. the first parameter associated with the trend component, that constitutes an additive "bias" term, is subject to larger variance, which could indicate, that this coefficient is redundant given the other features and might inform further investigation.

Samples from the inferred parameter distributions are afterwards used to derive samples of predicted future cases. The resulting predictions thus incorporate both noise assumptions about the data as well as model uncertainty. This can be relevant for determining confidence intervals, in particular in situations where model uncertainty is large. The samples of the predictive distribution can in turn be used for additional processing, or if predictions in the form of point estimates are desired, they can be summarized by the posterior mean.

Possible extensions

To account for overdispersion in the data, we use a Negative Binomial distribution in this study. Other choices are possible, e.g. zero-inflated distributions [8, 12] or quasi-Poisson distributions [29], each of which has a different implication for the resulting model. Since the Negative Binomial distribution assigns more weight to smaller counts relative to quasi-Poisson [29], the latter may be a more adequate choice when accurately predicting higher counts, e.g. during outbreaks, is critical. If there are differences between individual counties, that are suspected to lead to varying degrees of overdispersion, the overdispersion parameter α of the Negative Binomial distribution could also be chosen to vary in time and space like the corresponding mean μ [30, 31].

Whereas spatio-temporal interaction effects are here modeled as a function of geographical proximity, the kernel's composite basis functions make it possible to use alternative spatial distance measures, e.g. derived from transportation networks for people or goods [32]. For diseases where the kernel clearly factorizes into a single temporal and spatial component, a simpler spatial kernel function with a parameter for the bandwidth could be chosen. This allows including further prior assumptions or constraints, e.g. strict non-negativity or power law characteristics of interactions [33].

Due to the flexibility of the probabilistic modeling and sampling approach, additional variables can be easily included and their influence analyzed (e.g. weather data, geographical features like forests, mountains and water bodies, the location and size of hospitals, vaccination rates, migration statistics, socioeconomic features, population densities, self-reported infections on social media [34], work, school and national holidays, weekends and large public events). For features where precise values are not known, probability distributions could be

specified and included in the probabilistic model, which could improve the model's estimate of uncertainty. For example, since the precise locations and times of individual infections are not publicly known, we simply assume a geographically and temporally uniform distribution of cases within the given county and calendar week. The conditional probability distributions could be refined by incorporating additional information (e.g. weekends and population density maps). However, precise information about place and time of infections are available to local health agencies. The model presented here could readily be implemented there to use this more accurate data.

In this study, we assume that the presented model, due to time-varying features as well as interaction effects is flexible enough to model the dynamics of the diseases in question throughout the year. There may, however, be influential latent variables that cannot be explicitly included as exogenous variables, in particular for diseases with very pronounced epidemic outbreaks. In such cases, the 'outbreak' stage of the disease could be modeled separately from the baseline stage, thereby increasing the degrees of freedom in the model. This has been demonstrated for dengue fever [8], where Markov switching is employed to detect sudden changes in the expected number of cases and provide early warnings when such a state transition occurs.

Conclusion

In this paper, a probabilistic model is proposed for predicting case counts of epidemic diseases. It takes into account a history of reported cases in a spatially extended region and employs MCMC sampling techniques to derive posterior parameter distributions, which in turn are incorporated in predicted probability distributions of future infection counts across time and space.

For all three tested diseases (campylobacteriosis, rotavirus and borreliosis) the same model, using interaction effects, temporal, political and demographic information, performs well and produces smooth predictions in time and space. For each disease, the inferred spatio-temporal kernels capture the specific interaction effects in a single function, that can be visualized and interpreted, and can be applied regardless of the topology of counties or their neighborhood relationships. A comparison with the standard *hhh4* model, which uses maximum likelihood estimation instead of Bayesian inference, shows comparable performance. At the expense of higher computational costs than the point estimate used in *hhh4*, the sampling approach employed here provides information about the full posterior distribution of parameters and predictions. The posterior parameter distribution provides information about the relevance of the corresponding features for the inferred model, and helps in identifying redundant features or violated model assumptions. The inferred features of our model are interpretable and their individual contribution to the model prediction can be analyzed: spatio-temporal interactions reveal information about the dynamic spread of the disease, temporal features capture seasonal fluctuations and long-term trends, and the assigned weights indicate relevance of additional features. The posterior predictive distribution also accounts for the uncertainty about parameters, e.g. due to simplifying model assumptions or a lack of data, rather than just the variability inherent in the data itself. This additional information is valuable for public-health policy-making, where accurate quantification of uncertainty is critical.

Supporting information

S1 Fig. Marginal posterior distributions of all parameters for campylobacteriosis. For each of four Markov chains, the mean (dot), the range from the 25% to 75% percentile (thick horizontal lines) as well as the 2.5% to 97.5% percentile (thin horizontal lines) are shown. For all

parameters, these summary statistics of the marginal distribution are similar across all four chains, indicating convergence of the MCMC sampling scheme (see also S7 Fig).
(TIFF)

S2 Fig. Marginal posterior distributions of all parameters for rotavirus. For each of four Markov chains, the mean (dot), the range from the 25% to 75% percentile (thick horizontal lines) as well as the 2.5% to 97.5% percentile (thin horizontal lines) are shown. For all parameters, these summary statistics of the marginal distribution are similar across all four chains, indicating convergence of the MCMC sampling scheme (see also S7 Fig).
(TIFF)

S3 Fig. Marginal posterior distributions of all parameters for Lyme borreliosis. For each of four Markov chains, the mean (dot), the range from the 25% to 75% percentile (thick horizontal lines) as well as the 2.5% to 97.5% percentile (thin horizontal lines) are shown. For all parameters, these summary statistics of the marginal distribution are similar across all four chains, indicating convergence of the MCMC sampling scheme (see also S7 Fig).
(TIFF)

S4 Fig. Sensitivity analysis for campylobacteriosis. Marginal posterior distributions of all parameters are shown for five different scales $\sigma_{W_{\mathrm{IA}}} = \{0.625, 2.5, 10.0, 40.0, 160.0\}$ (color coded), which includes the special case $\sigma_{W_{\mathrm{IA}}} = 10$ (see also S1 Fig) as used throughout this paper. For priors with standard deviation larger than 2.5, there is little qualitative change in the posterior distribution.
(TIFF)

S5 Fig. Sensitivity analysis for rotavirus. Marginal posterior distributions of all parameters are shown for five different scales $\sigma_{W_{\mathrm{IA}}} = \{0.625, 2.5, 10.0, 40.0, 160.0\}$ (color coded), which includes the special case $\sigma_{W_{\mathrm{IA}}} = 10$ (see also S2 Fig) as used throughout this paper. For priors with standard deviation larger than 2.5, there is little qualitative change in the posterior distribution.
(TIFF)

S6 Fig. Sensitivity analysis for Lyme borreliosis. Marginal posterior distributions of all parameters are shown for five different scales $\sigma_{W_{\mathrm{IA}}} = \{0.625, 2.5, 10.0, 40.0, 160.0\}$ (color coded), which includes the special case $\sigma_{W_{\mathrm{IA}}} = 10$ (see also S3 Fig) as used throughout this paper. Here, the choice of prior has considerably more impact on the posterior distribution than for campylobacteriosis (see S4 Fig) or rotavirus (see S5 Fig), for both of which more training data is available. For a narrow prior with standard deviation 0.625, the interaction effect coefficients appear to be strongly regularized towards zero.
(TIFF)

S7 Fig. Convergence diagnostics of MCMC chains. Gelman-Rubin diagnostics (red dots) for all parameters for campylobacteriosis (**1A**), rotavirus (**2B**) and borreliosis (**3C**). The values all lie close to 1.0 for all parameters, indicating convergence of the sampling procedure.
(TIFF)

S8 Fig. Predictions of case counts for campylobacteriosis for various counties across Germany. Reported infections (black dots), predictions of case counts by BSTIM (orange line) and the *hhh4* reference model (blue line) for campylobacteriosis for 25 counties in Germany. The shaded areas show the inner 25%-75% and 5%-95% percentile.
(TIFF)

S9 Fig. Predictions of case counts for rotavirus for various counties across Germany. Reported infections (black dots), predictions of case counts by BSTIM (orange line) and the *hhh4* reference model (blue line) for rotavirus for 25 counties in Germany. The shaded areas show the inner 25%-75% and 5%-95% percentile.
(TIFF)

S10 Fig. Predictions of case counts for borreliosis for various counties across Bavaria. Reported infections (black dots), predictions of case counts by BSTIM (orange line) and the *hhh4* reference model (blue line) for borreliosis for 25 counties in Bavaria. The shaded areas show the inner 25%-75% and 5%-95% percentile.
(TIFF)

Acknowledgments

We would like to thank our editor and anonymous reviewers for their comments which helped us to improve the paper.

Author Contributions

Conceptualization: Olivera Stojanović, Johannes Leugering, Gordon Pipa.

Data curation: Stéphane Ghozzi, Alexander Ullrich.

Methodology: Olivera Stojanović, Johannes Leugering.

Project administration: Olivera Stojanović.

Resources: Stéphane Ghozzi, Alexander Ullrich.

Software: Olivera Stojanović, Johannes Leugering.

Supervision: Gordon Pipa, Alexander Ullrich.

Validation: Stéphane Ghozzi, Alexander Ullrich.

Visualization: Olivera Stojanović, Johannes Leugering.

Writing – original draft: Olivera Stojanović, Johannes Leugering.

Writing – review & editing: Stéphane Ghozzi, Alexander Ullrich.

References

1. Faensen D, Claus H, Benzler J, Ammon A, Pfoch T, Breuer T, et al. SurvNet@RKI—a multistate electronic reporting system for communicable diseases. Euro surveillance: bulletin européen sur les maladies transmissibles = European communicable disease bulletin. 2006; 11(4):100–103.
2. Noufaily A, Enki DG, Farrington P, Garthwaite P, Andrews N, Charlett A. An improved algorithm for outbreak detection in multiple surveillance systems. Statistics in Medicine. 2013; 32(7):1206–1222. https://doi.org/10.1002/sim.5595 PMID: 22941770
3. Gertler M, Dürr M, Renner P, Poppert S, Askar M, Breidenbach J, et al. Outbreak of following river flooding in the city of Halle (Saale), Germany, August 2013. BMC Infectious Diseases. 2015; 15(1):1–10. https://doi.org/10.1186/s12879-015-0807-1
4. Salmon M, Schumacher D, Burmann H, Frank C, Claus H, Höhle M. A system for automated outbreak detection of communicable diseases in Germany. Euro Surveillance: Bulletin Europeen Sur Les Maladies Transmissibles = European Communicable Disease Bulletin. 2016; 21(13).
5. Kulldorff M. A spatial scan statistic. Communications in Statistics—Theory and Methods. 1997; 26 (6):1481–1496. https://doi.org/10.1080/03610929708831995
6. Kulldorff M, Heffernan R, Hartman J, Assunção R, Mostashari F. A space-time permutation scan statistic for disease outbreak detection. PLoS Medicine. 2005; 2(3):0216–0224. https://doi.org/10.1371/journal.pmed.0020059

7. Meyer S, Held L, Höhle M. Spatio-Temporal Analysis of Epidemic Phenomena Using the R Package surveillance. Journal of Statistical Software. 2017. https://doi.org/10.18637/jss.v077.i11
8. Chen CWS, Khamthong K, Lee S. Markov switching integer-valued generalized auto-regressive conditional heteroscedastic models for dengue counts. Journal of the Royal Statistical Society: Series C (Applied Statistics); 68(4):963–983. https://doi.org/10.1111/rssc.12344
9. Xia Y, Bjørnstad O, Grenfell B. Measles Metapopulation Dynamics: A Gravity Model for Epidemiological Coupling and Dynamics. The American Naturalist. 2004; 164(2):267–281. https://doi.org/10.1086/422341 PMID: 15278849
10. Held L, Meyer S. Forecasting Based on Surveillance Data. arXiv:180903735 [stat]. 2018;.
11. McCullagh P, Nelder JA. Generalized Linear Models. 2nd ed. Chapman & Hall/CRC Monographs on Statistics and Applied Probability. Chapman & Hall/CRC; 1989.
12. Lee JH, Han G, Fulp W, Giuliano A. Analysis of overdispersed count data: application to the Human Papillomavirus Infection in Men (HIM) Study. Epidemiology & Infection. 2012; 140(6):1087–1094. https://doi.org/10.1017/S095026881100166X
13. Gurland J. Some Applications of the Negative Binomial and Other Contagious Distributions. American Journal of Public Health and the Nations Health. 1959; 49(10):1388–1399. https://doi.org/10.2105/AJPH.49.10.1388
14. Coly S, Yao AF, Abrial D, Garrido M. Distributions to model overdispersed count data. Journal de la Societe Française de Statistique. 2016; 157(2):39–63.
15. Gelman A. Prior distributions for variance parameters in hierarchical models (comment on article by Browne and Draper). Bayesian Analysis. 2006; 1(3):515–534. https://doi.org/10.1214/06-BA117A
16. Hoffman MD, Gelman A. The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. arXiv e-prints. 2011; p. arXiv:1111.4246.
17. Salvatier J, Wiecki TV, Fonnesbeck C. Probabilistic programming in Python using PyMC3. PeerJ Computer Science. 2016; 2:e55. https://doi.org/10.7717/peerj-cs.55
18. Gelman A, Rubin DB. Inference from Iterative Simulation Using Multiple Sequences. Statistical Science. 1992; 7(4):457–472. https://doi.org/10.1214/ss/1177011136
19. De Boor C. On calculating with B-splines. Journal of Approximation theory. 1972; 6(1):50–62. https://doi.org/10.1016/0021-9045(72)90080-9
20. Food WHOa, of the United Nations and World Organisation for Animal Health AO. The global view of campylobacteriosis: report of an expert consultation, Utrecht, Netherlands, 9-11 July 2012. World Health Organization; 2013. Available from: https://apps.who.int/iris/handle/10665/80751.
21. Parashar UD, Nelson EAS, Kang G. Diagnosis, management, and prevention of rotavirus gastroenteritis in children. BMJ (Clinical research ed). 2013; 347:f7204.
22. Steere AC, Strle F, Wormser GP, Hu LT, Branda JA, Hovius JWR, et al. Lyme borreliosis. Nature reviews Disease primers. 2016; 2:16090. https://doi.org/10.1038/nrdp.2016.90 PMID: 27976670
23. Stutzer A, Frey BS. Commuting and Life Satisfaction in Germany. Informationen zur Raumentwicklung. 2007;.
24. Burda MC, Weder M. The Economics of German Unification after Twenty-five Years: Lessons for Korea. Sonderforschungsbereich 649, Humboldt University, Berlin, Germany; 2017. SFB649DP2017-009. Available from: https://ideas.repec.org/p/hum/wpaper/sfb649dp2017-009.html.
25. Zawilska-Florczuk M, Ciechanowicz A. One country, two societies? Germany twenty years after reunification. Centre for Eastern Studies; 2011. Available from: https://www.osw.waw.pl/en/publikacje/osw-studies/2011-02-15/one-country-two-societies-germany-twenty-years-after-reunification.
26. Watanabe S. A Widely Applicable Bayesian Information Criterion. J Mach Learn Res. 2013; 14(1):867–897.
27. Gelman A, Hwang J, Vehtari A. Understanding predictive information criteria for Bayesian models. Statistics and Computing. 2014; 24(6):997–1016. https://doi.org/10.1007/s11222-013-9416-2
28. Dawid AP, Sebastiani P. Coherent dispersion criteria for optimal experimental design. The Annals of Statistics. 1999; 27(1):65–81.
29. Ver Hoef JM, Boveng PL. Quasi-Poisson vs. negative binomial regression: how should we model overdispersed count data? Ecology. 2007; 88(11):2766–2772. https://doi.org/10.1890/07-0043.1 PMID: 18051645
30. Lawson AB. Bayesian Disease Mapping: Hierarchical Modeling in Spatial Epidemiology. 3rd ed. Chapman & Hall/CRC Interdisciplinary Statistics. New York: Chapman and Hall/CRC; 2018.
31. Banerjee S, Haining RP, Lawson AB, Ugarte MD. Handbook of Spatial Epidemiology. 1st ed. Chapman & Hall/CRC handbooks of modern statistical methods. New York: Chapman and Hall/CRC; 2016.

32. Manitz J, Kneib T, Schlather M, Helbing D, Brockmann D. Origin Detection During Food-borne Disease Outbreaks – A Case Study of the 2011 EHEC/HUS Outbreak in Germany. PLOS Currents Outbreaks. 2014 https://doi.org/10.1371/currents.outbreaks.f3fdeb08c5b9de7c09ed9cbcef5f01f2

33. Meyer S, Held L. Power-law models for infectious disease spread. Annals of Applied Statistics. 2014; 8(3):1612–1639. https://doi.org/10.1214/14-AOAS743

34. Pipa G. Analyzing tweets to predict flu epidemics; 2017. Available from: https://www.ibm.com/blogs/client-voices/analyzing-tweets-predict-flu-epidemics/.