

NEURAL COMPUTATION AND TIME

PASCAL NIETERS

Doktor der Naturwissenschaft (Dr. rer. nat.)
Neuroinformatics
Institute of Cognitive Science
University Osnabrück

December 2021

Pascal Nieters: *Neural computation and time* © December 2021

SUPERVISORS:

Prof. Dr. Tim Kietzmann

Prof. Dr. Michael Franke

Prof. Dr. Gordon Pipa

LOCATION:

Osnabrück

DISPUTATION:

11.05.2022

ABSTRACT

Time is not only the fundamental organizing principle of the universe, it is also the primary organizer of information about the world we perceive. Our brain encodes these perceptions in sequential patterns of spiking activity. But different stimuli lead to different information encoded on different timescales; sometimes the same stimulus carries information pertaining to different perceptions on different timescales. The orders of time are many and the computational circuits of the brain must disentangle these interwoven threads to decode the underlying structure.

This thesis deals with solutions to this disentanglement problem implemented not at the network level, but in smaller systems and single neurons that represent the past by clever use of internal mechanisms. Often, these solutions involve the intricate tools of the neural dendrite or other peculiar aspects of neural circuits that are well known to physiologists and biologists but disregarded in favor of more homogeneous models by many theoreticians. It is at the intersection of the diverse biological reality of the brain and the difficulty of the computational problem to disentangle the threads of temporal order that we find new and powerful computational principles:

Symbolic computation on the level of single neurons via dendritic plateau potentials, embedding history in delayed feedback dynamics or consecutive filter responses, or the idea that learning a generalized differential description of a systems can largely forgo the need to remember the past – instead, patterns can freely be generated.

Together, the different challenges that information ordered in different, asynchronous times present require a diverse palette of solutions. At the same time, computation and the structure imposed by time are deeply connected.

PUBLICATIONS

In [chapter 2](#) - [chapter 3](#), I follow ideas and figures first developed in close collaboration with Johannes Leugering and elaborate on and revise them in the context of this thesis. [chapter 4](#) follows up on some of these ideas and develops them deeper. The contents are largely new and have not yet been published. The following contributions are relevant:

[section B.1](#) *Preprint, under review, shared first author:*

Johannes Leugering, Pascal Nieters and Gordon Pipa. ‘A minimal model of neural computation with dendritic plateau potentials.’ In: *bioRxiv* (2021), p. 690792

[section B.2](#) *Conference abstract, first author:*

Pascal Nieters, Johannes Leugering and Gordon Pipa. ‘Active dendrites implement probabilistic temporal logic gates.’ In: *Proc. of the Computational Cognition Workshop Osnabrück*. 2019

[section B.3](#) *Patent, shared first author:*

Johannes Leugering, Pascal Nieters and Gordon Pipa. ‘Neuromorphic Pattern Detector and Neuromorphic Circuitry Here-with’. Pat. DE:102019134044:A1. June 2021

In [chapter 5](#) I summarize and put results first published in the following contributions into the wider context of the thesis:

[section B.4](#) *Peer reviewed published journal article, first author:*

Pascal Nieters, Johannes Leugering and Gordon Pipa. ‘Neuromorphic computation in multi-delay coupled models’. In: *IBM Journal of Research and Development* 61.2/3 (2017), pp. 8–7

[section B.5](#) *Conference abstract, shared first author:*

Pascal Nieters, Johannes Leugering and Gordon Pipa. ‘Neuromorphic Adaptive Filters for event detection, trained with a gradient free online learning rule’. In: *Cognitive Computing – Merging Concepts with Hardware*. 2018

[section B.6](#) *Peer reviewed published book chapter:*

Johannes Leugering, Pascal Nieters and Gordon Pipa. ‘Computational Elements of Circuits’. In: *The neocortex*. Ed. by Wolf Singer, Terrence J Sejnowski and Pasko Rakic. MIT Press, 2019, pp. 195–209

Lastly, [chapter 6](#) summarizes and puts into context results from work in collaboration with Rahel Vortmeyer-Kley:

[section B.7](#) *Peer reviewed, published journal article, shared first author:*

Rahel Vortmeyer-Kley, Pascal Nieters and Gordon Pipa. ‘A trajectory-based loss function to learn missing terms in bifurcating dynamical systems’. In: *Scientific reports* 11.1 (2021), pp. 1–13

[section B.8](#) *Conference abstract:*

Rahel Vortmeyer-Kley, Pascal Nieters and Gordon Pipa. ‘A trajectories’ guide to the state space-learning missing terms in bifurcating ecological systems’. In: *EGU General Assembly Conference Abstracts*. 2021, EGU21–16159. DOI: [10.5194/egusphere-egu21-16159](https://doi.org/10.5194/egusphere-egu21-16159). URL: <https://doi.org/10.5194/egusphere-egu21-16159>

ACKNOWLEDGEMENTS

Support on my journey to this completed dissertation was plentiful, otherwise I am sure i would not have arrived at the end. Still, a public display of feelings in prose makes me deeply uncomfortable. Therefore, an abridged summary of support I am deeply grateful for:

My parents and my family, for keeping my feet mostly on the ground and allowing me to become who I am.

My friends of old, whom I've been with since school, for accepting who I am and who I want to be. Also for keeping in touch when I continue to be bad at it.

My colleagues many of whom I now consider friends – both in the smaller Neuroinformatics lab of old and the newer, significantly larger version – for many stimulating discussion and fantastically supportive work environment. A particular nod to Johannes, for endless sessions of complaining about theoretical neuroscience while trying to do theoretical neuroscience.

The entire Institute of Cognitive Science for the advice and support I was given – both in administrative matters and in science.

A special thank you to my supervisor, Gordon. If not for you, I would have never considered a career in science or that it may be possible for me to write a doctoral thesis at all in the first place. Along the way, you have offered support and advice when needed but gave me and others, the freedom to find out who we are as scientists. More importantly, however, I do not think I would have taken any of it if you weren't the person you are. So, thank you, deeply.

An ode to my wonderful partner and wife will remain missing from this list. No words can do justice to the support and love you have given and continue to give, and the support and love you allow me to return. You make me a better person.

CONTENTS

1	INTRODUCTION: COMPUTATION <i>in time</i>	1
2	SEQUENTIAL ORGANIZATION OF INFORMATION IN THE BRAIN.	9
2.1	Sequences in neural representations.	10
2.2	The neural basis of sequence processing in single neurons.	15
3	SEGMENTED DENDRITIC TREES	23
3.1	The SDT Model	23
3.2	Example: Detecting paths from place cell activity . . .	29
4	COMPUTATION IN NEURONS WITH ACTIVE DENDRITES.	35
4.1	Structured computation and Events in SDT neurons. .	39
4.2	SDT Neurons and Networks.	52
4.3	A neuromorphic hardware implementation of the SDT neuron.	57
5	PREDICTING TIME-SERIES WITH DYNAMIC COMPUTING SYSTEMS	61
6	LEARNING GENERALIZED DYNAMICS FROM TRAJECTORIES.	71
7	DISCUSSION	77
I	APPENDIX	83
A	APPENDIX: ADDITIONAL METHODS	85
A.1	Implementation of the navigation experiment	85
A.2	Simulation framework for dendritic plateau computation	86
A.3	A method for experimental verification.	86
B	PUBLICATIONS AND CONTRIBUTIONS	89
B.1	Paper: A minimal model of neural computation with dendritic plateau potentials	89
B.2	Conference Abstract: Active dendrites implement temporal logic gates	107
B.3	Patent: Neuromorphic Pattern Detector and Neuromorphic Circuitry	109
B.4	Paper: Neuromorphic computation in multi-delay coupled models	132
B.5	Adaptive Filters	133
B.6	Book Chapter: Computational Elements of Circuits . .	135
B.7	Paper: A trajectory-based loss function to learn missing terms in bifurcation dynamical systems	136
B.8	Conference Abstract: A trajectories' guide to the state space - learning missing terms in bifurcating ecological systems	150

LIST OF FIGURES

Figure 1	COMPUTERS OPERATE ON SYNCHRONIZED CLOCKS	2
Figure 2	NEURONS OPERATE ON INTERNAL CLOCKS .	3
Figure 3	NATURAL STIMULI CAN CONTAIN INFORMATION ON MANY DIFFERENT TIME-SCALES.	4
Figure 4	COMPUTATION AND MEMORY STRATEGIES.	6
Figure 5	EXAMPLES: SEQUENCES IN THE BRAIN . . .	11
Figure 6	SKETCH: AN SDT SIMPLIFIES THE STRUCTURE OF A NEURON'S DENDRITIC TREE. . .	24
Figure 7	GENERATION OF PLATEAUS IN BIOLOGY AND IN THE MODEL.	28
Figure 8	PATH-DETECTION FROM PLACE CELL ACTIVITY.	30
Figure 9	SPATIOTEMPORAL RECEPTIVE FIELDS OF SDT NEURONS.	32
Figure 10	SIMPLE AND COMPLETE REPRESENTATIONS OF SDT NEURONS	39
Figure 11	COMPUTATIONAL EVENTS FROM A SPIKE RASTER.	41
Figure 12	ENCODING AND DECODING STIMULUS UNCERTAINTY.	44
Figure 13	RANK-ORDERING OF SEQUENTIAL REAL-TIME EVENTS.	46
Figure 14	RANK-ORDERING OF PARALLEL EVENTS AND INHIBITION.	47
Figure 15	RANK ORDERING OF MULTIPLE EVENTS. . .	48
Figure 16	EVALUATING AN SDT EXPRESSION WITH DETERMINISTIC SYNAPSES.	50
Figure 17	CONVERTING MCCULLOCH AND PITTS NETS.	54
Figure 18	REDUCING SDT NEURONS TO SINGLE SEGMENT MODELS.	56
Figure 19	A SIMPLE SDT SEGMENT HARDWARE IMPLEMENTATION.	58
Figure 20	RESERVOIR COMPUTERS REPRESENT PAST INPUTS IN THEIR ACTIVATION VECTOR.	63
Figure 21	COMPUTATION IN DELAY-COUPLED SYSTEMS WITH MULTIPLE DELAYS	66
Figure 22	FILTER BANKS CAN REPRESENT A SIGNAL ON MULTIPLE TIMESCALES.	67
Figure 23	PROOF OF CONCEPT: ADAPTIVE FILTER NEURON	69
Figure 24	LEARNING MISSING TERMS IN THE SELKOV MODEL.	73
Figure 25	MSE AND LDA BASED LOSS FUNCTION	74

Figure 26	TD DISTRIBUTION SELKOV MODEL	75
Figure 27	DENDRITIC MORPHOLOGY IMPOSES TIMING CONSTRAINTS.	87

INTRODUCTION: COMPUTATION IN TIME

For living organisms, including us, the world changes in time from the past to the future. We remember what we ate for breakfast *yesterday* and know who you are discussing *tomorrow's* dinner plans with right *now*. Time orders our experience of the world, and we are naturally skilled at dealing with it [100].

Explaining our experience of time, or even capturing what time is scientifically has proven much more difficult. In his book “The order of time”, the physicist Carlo Rovelli calls it ‘[...] perhaps the greatest remaining mystery.’ [206, p. 2]. In the fundamental description of the universe, something peculiar happens and time loses its meaning that we are so accustomed to in every day life. First, time loses its directionality – in a world described by quantum fields, there is no distinction between past and future. Then, time loses universality becomes deeply personal – how fast your own clock ticks depends on your position in relation to centres of gravity, like the earth, and even changes depending on how fast you are moving. In one attempt to find a quantum theory of gravity, called loop quantum gravity, the variable “time” vanishes from the equation all together [46].

What remains of the description of the world is a network of events, ever changing processes, partially-ordered by what physicists call the arrow of time. The direction of the arrow of time emerges in accordance with the second law of thermodynamics, that entropy never decreases and disorder grows. Rovelli notes that ‘[...] entropy is the quantity that counts how many are the different configurations that our blurred vision does not distinguish between.’ [206, p. 30]. In our particular, blurred perception of the world entropy increases and the familiar concept of time materializes and separates past from future.

But, even then, time is experienced differently by different people in different situations [36]. We operate on our own, personal clocks that order our perception of the world.

Herein lies a fundamental question: Time orders our perception of the world, but it does so according to a different, personal clock for everything in it – *so how do we understand it?*

Understanding the world, the cognitive process of integrating perception with a mental model, is a computational process of the brain. Any computation is an operation on a set of inputs, which trivially requires that the operation is carried out when all inputs are available. Fulfilling this simple requirement in a physically realized computing system, however, is not at all trivial. Modern computers run on chips

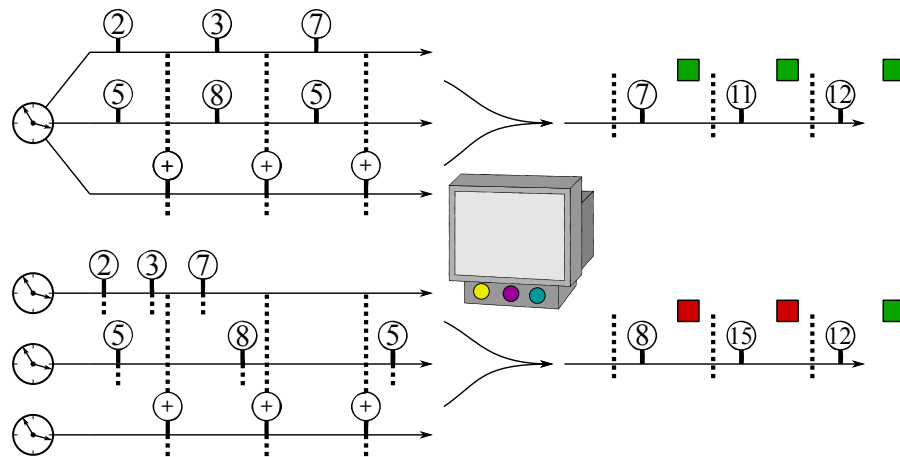


Figure 1: COMPUTERS OPERATE ON SYNCHRONIZED CLOCKS. A fictional computer that receives two data and one instruction signal. Its job is to add the first two numbers, the second two numbers, etc. It does so by adding the last two numbers seen since the last instruction signal on the current instruction signal. a. When both data signals are aligned with the same global clock as the instruction signal, the computer computes correctly. b. But when the clocks of the data signals are missaligned and operate on varying clock speeds, the computer has no mechanism to associate the correct two data signals with the current instruction signal. The computations are wrong.

architected as von-Neumann digital machines [258] that are specifically engineered to synchronize data and operations with global clock signals (Fig. 1a). If data and instruction are desynchronized because each signal has its own personal clock, the computer still computes numbers, it just doesn't give the right answers anymore (Fig. 1b).

The parallel and distributed neural network in the brain works differently. A neuron's computation is responsive to its input, if there is no input the neuron remains quiescent¹. To be able to compute whether for example the sum of active neurons in two populations is larger than a particular threshold, a very simple point neuron integrates the evoked post-synaptic potential (EPSP) in response to each spike of each population as it arrives. The duration of the post-synaptic response to the spike and the timescale of the neuron's leaky membrane potential can be understood as a very short memory cache that maintains intermediate results of the computation until it completes with a spike. Implicitly, the dynamic internal timescales of the neurons together with spike-triggered responsive computation determine the irregular internal clock of the neuron. This process is able to easily deal with minor delays in spike signals (Fig. 2a), but when different input populations desynchronize and input signals do not arrive in the correct, required short time window afforded by the dynamic

¹ This is not true for all biological neurons but describes all computational models of neurons commonly used.

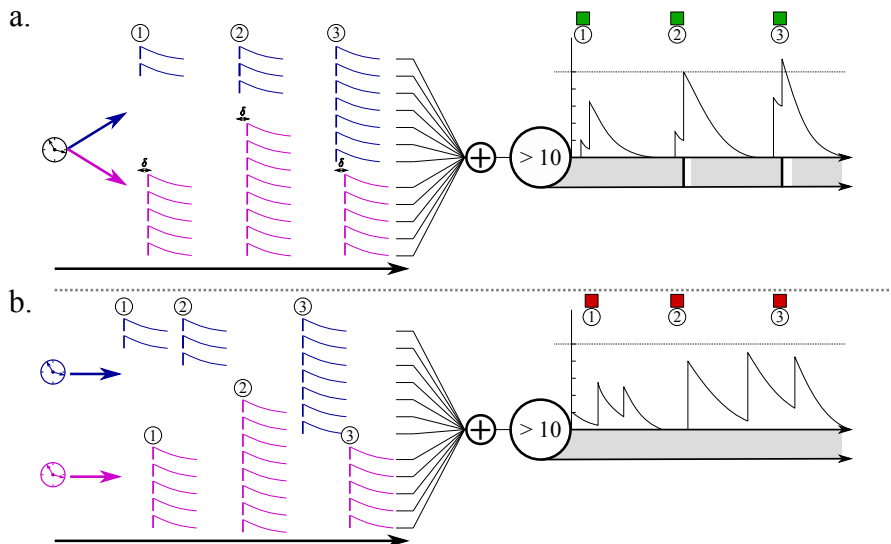


Figure 2: NEURONS OPERATE ON INTERNAL CLOCKS. This simplified cartoon neuron receives spiking input from two populations (blue and magenta) and has to emit a spike when a total of ten afferent neurons spiked in the first, second, or third volley. Each spike leaves an exponential trace in the membrane potential that is summed over and compared against this threshold. **a.** This endows the neuron with a short, dynamic and responsive memory cache in case the two input populations are slightly delayed but are aligned to a common clock. **b.** If each input population fires on different timescales, i. e. has its own clock, the misaligned inputs 1, 2, and 3 cannot be summed correctly anymore.

membrane response we have the same problem a traditional computer has without a global clock (Fig. 2b). We may still get answers, but the questions they are answers to may have shifted in unexpected ways and are no longer what we intended to ask.

In cognitive science, neuroscience, and neural network theory this problem is sometimes referred to as the temporal binding problem [60]. Temporal correlation was famously suggested as a strategy to coherently transmit information about one stimulus or event in time [225, 256] although the theory has come under critique more recently [28, 186].

We presupposed that it is not only simultaneity that is required by the encoding of stimuli. Instead, it is the temporal order of stimuli that is decisively important in our perception of the world and our understanding of it [130]. How the brain's neural circuitry can compute responses to ordered sequential stimuli with independent clocks on multiple different timescales remains mystery at the heart of our understanding of the brain.

Potential solutions to the problem of storing past inputs for a particular computation are heavily dependent on the question the computation must answer. Imagine the wooden bars of a xylophone arranged as a musical staircase. If you push a marble down it, you can observe

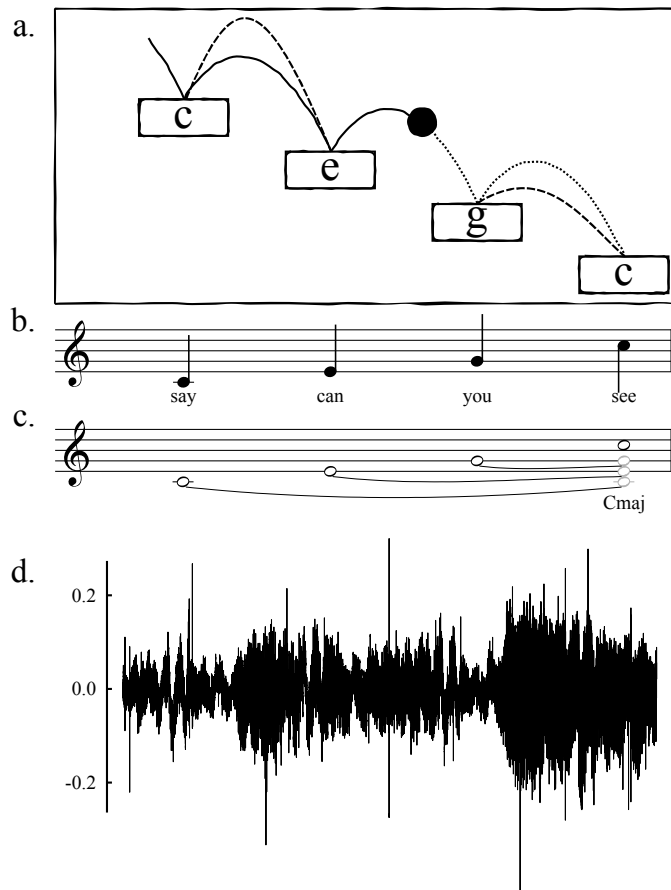


Figure 3: NATURAL STIMULI CAN CONTAIN INFORMATION ON MANY DIFFERENT TIME-SCALES. A marble hopping down a set of musical stairs generates many temporal stimuli that the brain can understand. a. It's movement trajectory can be extrapolated to predict the time of the next impact. b. the sequence of sounds resulting from each impact may be recognized as the beginning melody of the American national anthem, or c. as a single C-major chord, for example out of John Lennon's "Imagine", if the sounds overlap. d. If the song "The Star-Spangled Banner" is instead sung, the waveform of the audio our ears receives contains the tone of voice of the singer and the actual words sung.

it hopping from note to note and listen to the sound (Fig. 3a). Your brain will be able to extrapolate the trajectory of the marble based on its previous path and velocity. You are able to predict when the marble will hit the next note. This problem can be solved by storing a short past of the trajectory as a number of sampled points, as well as the positions of each bar.

You also hear the notes in sequence as the melody and recognize it from the beginning of the American national anthem "A Star-Spangled Banner" (Fig. 3b). In this case, the individual notes should be stored as separate events in the correct order and for long enough to identify the next element in the sequence. If the marble takes the dashed line, resulting in an alternative timing and rhythm but the same order of

notes, you are likely still able to identify the melody.

Alternatively, especially if the notes ring out, you can hear the C-major chord, extensively featured throughout music. John Lennon's "Imagine" is a particular example that comes to mind. Now, from the perspective of recognizing the chord C-major, each note adds to the evidence for the chord. Their particular order may modify our musical perception, but it is not fundamentally important to the identification of the chord. You do have to store all notes in memory, at least until the major triad c-e-g rings out, to confidently solve the problem.

Lastly, a vocal recording of the phrase "say can you see" from the beginning of the anthem contains an even richer tapestry of information. The periodicity, the envelope, and the fine-structure of the signal can be decoded to identify the singer, to register the tone of voice and articulation, and connect emotionally to the musical performance [204].

Which strategy to keep past information in memory during a computation is therefore inherently dependent on the question the computation is tasked to answer.

Examples from language illustrate how time can be the organizing principle for discrete and structured information. "The cat sits on the table" means something entirely different than "The table sits on the cat". Common to examples where time organizes the structure of information is that the specific timing of each element in a sequence does not change the semantic meaning and should often be ignored. Here, a good strategy is a real-time version of a finite-state automaton that requires the correct order in its input sequence and maintains each state for some time before it resets and can be reused for a new computation (Fig. 4a).

Alternatively, the sequence in question may be best thought of as a continuous trajectory through a space. Examples include the hopping marble or a baseball hurtling towards you, but also the more abstract situation such as the time-series of a stock price one may want to forecast and predict. We can conceptualize these examples as continuously changing dynamic systems, and of the sequence as a series of sampled measurement of it that have no inherent structure beyond belonging to the same system. In this case, it is often a good strategy to keep a sufficient number of these sampled measurements in accessible memory and use all of them in the prediction and forecasting of the trajectories future (Fig. 4b).

If we study the systems we observe, we are often able to generalize and predict their future and past behavior from a single measurement and a set of parameters of the system. This can be done if we have access to the systems governing differential equation that describes

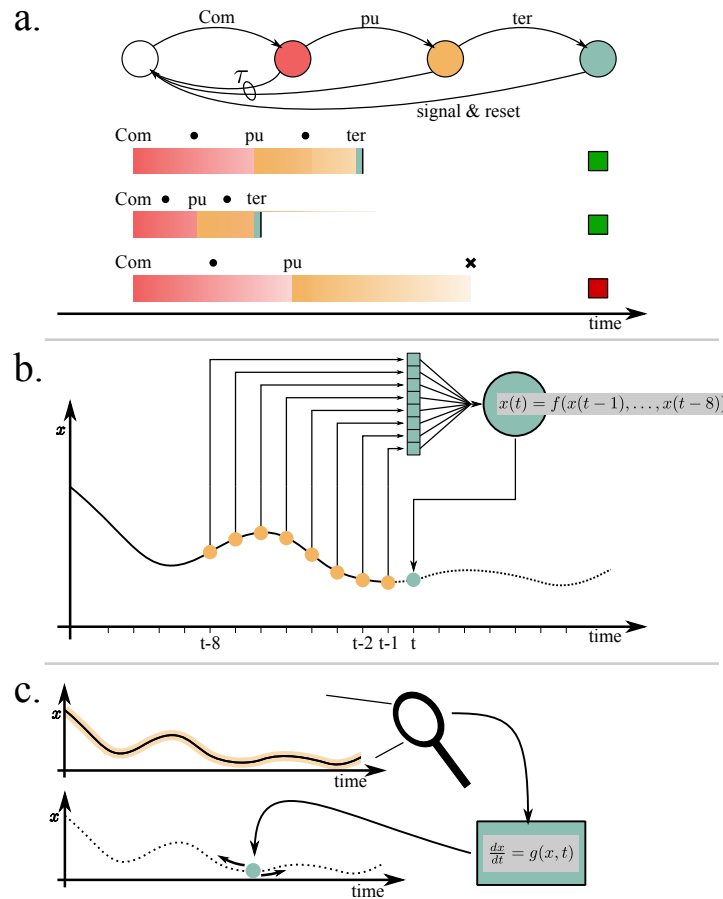


Figure 4: COMPUTATION AND MEMORY STRATEGIES.

a. Structured information ordered in time can be memorized by the current state of a finite-state automaton. Each correct, new syllable corresponds to a transition until the word is complete. The final state (green) marks successful recognition, which can be marked by a signal and a reset to initial position to restart the automaton. If the correct syllable is not heard for a time τ , failure is assumed and the automaton resets.

b. To predict and forecast the behavior of a particular time-series, a collection of sampled points along it must be kept in memory. A black box model f , for example a generalized linear model or an artificial neural network, can then be used to learn the prediction of a new point along the trajectory.

c. A differential equation $\frac{dx}{dt} = g(x, t)$ describes the continuous change of system with respect to time. If we can find the function g in a data-driven way, we can predict both the future and past of any point on a trajectory without the need to explicitly store more information in memory.

the change of the variables in relation to each other and time². Discovering differential equations is one of the core pursuits of scientific inquiry, but we can also use modern machine learning and artificial neural networks to approximate the differential equation in a data-

² Usually in the universal, Newtonian sense.

driven way. In this approach, the approximation of the differential equation encodes a generalized version of change in the system that we can use to predict, forecast and analyze without further explicit need for memory (Fig. 4c).

At the center of this thesis is the question how neural circuits can represent information about past inputs relevant to a particular computation. In large parts, we will focus on seemingly simple solutions that can be implemented even in computing systems as small as single neurons and do not require the external labeling of order, the external discretization and synchronization of different clocks, or the externalization of memory. This internalized processing of information irregularly ordered by time is what we know brains are able to do. We suggest the diversity of biological processes may lead to diverse solutions adapted to different problems and make the following contributions:

1. Segmented dendritic trees respond to spiking input by generating a long plateau potential that matches the timescale of sequentially organized information in the brain ([chapter 2](#)). The interaction of plateau potentials convincingly decodes the rank-order of sequential events in single neurons independent of precisely clocked timing ([chapter 3](#)). The model can also form the basis for a symbolic computational architecture of the brain and offers a new perspective on neural computation ([chapter 4](#)).
RELATED PUBLICATIONS AND CONFERENCE CONTRIBUTIONS:
[B.1](#), [B.2](#), [B.3](#)
2. Single-node systems with delayed feedback and filtering are able to represent trajectories of dynamical systems and time-series in their own dynamic state. Prediction and forecasting questions can therefore be answered by systems that have a simple physical implementation but complex dynamics ([chapter 5](#)).
RELATED PUBLICATIONS AND CONFERENCE CONTRIBUTIONS:
[B.4](#), [B.5](#), [B.6](#)
3. Differential equations are the generalized and symbolic representation of changing, dynamic systems and are one of the most important tools to scientifically understand our ever changing universe. Neural networks, as a data science and artificial intelligence tool, can approximate this equation. By changing the loss function in this data-driven approach to discovering a generalized representation of system change, we are able to find missing terms in systems with bifurcations from single trajectories ([chapter 6](#)).
RELATED PUBLICATIONS AND CONFERENCE CONTRIBUTIONS:
[B.7](#)

SEQUENTIAL ORGANIZATION OF INFORMATION IN THE BRAIN.

The brain constantly processes information encoded in temporal sequences of spiking activity. Much of it emerges from sensory input, our perception of the world as it changes in time across multiple dynamically varying timescales. But the brain is also perfectly capable of generating sequences internally. Karl Lashley noted as early as 1951 that many behaviors simply cannot be explained by a succession of external stimuli mapping onto an action, as each stimulus must be integrated into a system that is already excited and organized [130]. He raised the problem of serial order. He argued that a theory is missing which explains "[...] how people store and retrieve a sequence of item in the correct order." as Henson and Burgess summarized [95, page 1]. Henson and Burgess [95] further discuss how the problem can potentially be solved by different codes and representations in cognitive architectures, but they leave out the question of neural implementation. Yet, if the brain encodes the world in temporal sequences, and generates temporal sequences on its own, and if decoding the serial order of these sequences is a key component of behavior and cognition, then there must be some neural circuitry or mechanism that can robustly decode this information. Currently, this circuitry or mechanism is unknown.

First of all, we must know how exactly the brain organizes information sequentially. In recent years, a number of investigations into often completely different regions of the brain, engaged a variety of tasks and involving different sensory modalities, have started to collect evidence. Putting them side-by-side not only reveals that there are significant similarities in how external sequences are represented, but that the brain seems at least in part specifically organized to support the organization and generation of sequences (see Fig. 5 for an overview in the next section). This allows us to paint a clear picture of what the scope of the problem is: Sequence elements are encoded on a fast timescale, typically less than 30 ms. Sequences are presented on a much slower timescale of around 200 ms or more. The core challenge is to decode the ranked order of sequence elements and compute a response that is sensitive to this order. The global timescale of sequences can vary drastically and is simply too long for typically assumed timescales of single neurons. The next section summarizes the evidence for this analysis from different brain regions and exper-

iments.

In the following section, we reexamine the computational toolkit of single neurons. We suggested in the introduction that a stable, maintained memory of the state of a computation can be a robust solution to this problem. A relatively recently discovered neural process, the dendritic plateau process [4], fits this description perfectly. Even though the vast majority of neural tissue is occupied by neural dendrites [15] and the structure and active processes of the dendrite may be vitally important to solve problems on the single neuron level [144] including working memory [84], they are frequently ignored in computation models of the brain. Therefore, we reevaluate the core neurobiological assumptions that must be made for a model based on plateau potentials compared to more typical point neuron approaches in the second section of this thesis.

2.1 SEQUENCES IN NEURAL REPRESENTATIONS.

Sequences of spikes and brain activity are ubiquitous. This is not surprising at all, as we record activity over time. However, the brain is unlikely to care very much about the time pieces we use to record its activity [31]. Instead we have to understand how these sequences are internally organized. In the following, we have summarized a number of investigations ranging from the processing of speech in auditory cortex, to the representation of movement through space and potentially time more generally in hippocampus, to the identification of consecutive odors and visual stimuli (see Figure 5).

AUDITORY PROCESSING The first example is the most apparent and follows from the example in the introduction: Auditory processing of speech in the human brain is sequentially organized. Somewhat unsurprisingly, speech contains dynamics on multiple timescales naturally analysed in the signal processing framework of frequencies and envelopes [204]. What recent investigations began to shed light on is how the brain understands this intricate signal.

Firstly, multiple behavioral studies found that speech processing can

¹ All image credits (CCA/wiki licenses) for Fig. 5:
 Nose/Ear adapted by user "hunotik"
<https://thenounproject.com/icon/laryngology-42647/>
 Pepper adapted by user "the mother of japan"
<https://commons.wikimedia.org/wiki/File:Paprika.svg>
 Chocolate cake by user "Candyman777"
https://commons.wikimedia.org/wiki/File:Choc_cake_ill_01.svg
 Speaker by user "Mobius"
https://pt.wikipedia.org/wiki/Ficheiro:Speaker_Icon.svg
 Brain adapted from Patrick J. Lynch
https://en.wikipedia.org/wiki/File:Skull_and_brain_normal_human.svg

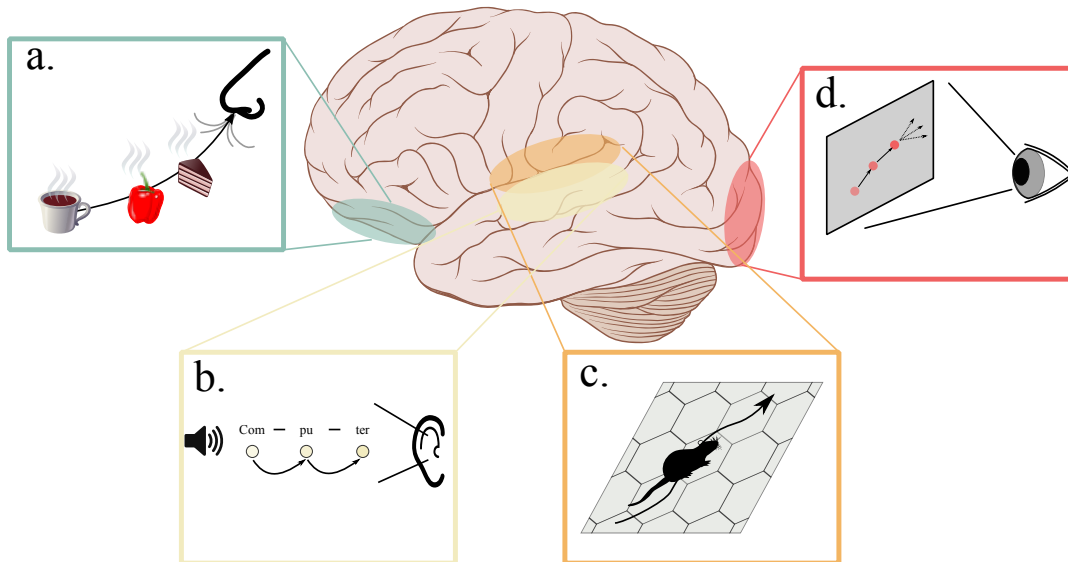


Figure 5: EXAMPLES: SEQUENCES IN THE BRAIN encode information in olfaction (a.), in auditory processing (b.), in the consecutive representation of visited locations (c.) and in moving visual stimuli (d.)¹

be linked to two specific timescales. On a local level, sounds for individual phonemes are processed on a short timescale of up to 50 ms whereas the global temporal order of individual sounds, for example in syllables, is encoded on a longer timescale of around 200 ms. Intelligibility tests showed that both local information and global, structural information about a stimulus is important for identification [33, 247] and behavioral performance suggests that the global structure is represented as prototypes in the brain [162]. Judging the order of stimuli is also a classical task in auditory processing. This task is interesting because tokenization of the individual sequence elements is important and their order is much harder to distinguish when less than 15 ms to 20 ms separate sounds [98]. The same local-global auditory process is associated with the same timescales. The longer timescale can be varied in many cases without effecting the task significantly [246].

Importantly, MEG analysis supports information processing in the brain on these timescales, too. Luo and Poeppel [150] find two distinct integration windows of 20 ms and 200 ms. The slower integration window resets and thus represents a “privileged timescale”, which encodes speech syllables [149].

The multiple timescale encoding that can be found in the analysis of speech recordings is thus reflected in both behavioral and brain recording data, supporting the claim that it is an encoding used by the brain internally. The slower timescale does not seem to be tuned to the timing of sequence elements. Global structure, such as a syllable,

are defined by their order.

REPRESENTATION OF LOCATION The second example of sequential organization of information highlights the independence of the sequential code from the specific timing of sequence elements. Place cells in the hippocampus [181] have receptive fields centered at specific locations in a physical environment and therefore encode the location of an agent in this environment. During navigation the agent will inevitably traverse multiple locations in sequence, and the timing between these visits is dependent on the agent's movement speed. Thus, place cell populations activate in sequence, and their relative timing reflects movement speed [2]. It is, however, not the only factor that influences the timing of sequential place cell activation.

Firstly, in a 200 ms window (corresponding to a theta cycle) a number of place cell populations activate for 25 ms sequentially (corresponding to a gamma cycle) [182] in a phenomenon that can be explained by the dynamics of recurrent networks [250]. These 200 ms sequence episodes combine past, present and predicted future location of the agent [49].

Secondly, sequential activation of place-cells can be replayed during slow-wave sleep, but at 20-fold faster speed [132]. A sequence of neural activation that unfolded over 5 s during navigation was recorded again during sleep compressed to 100 ms. Later experiments discovered, that sequences of place cells can also be "preplayed" during rest or sleep before exploration [50]. Preplay of place-cells can also encode a path to a goal in the environment at compressed timescales of 100 ms [189].

In sum, place cell population activity is organized in sequences whose timing can reflect movement speed on behavioral timescales, an internal "encoding-rhythm" or compressed sequence replay during sleep or preplay in anticipation of a navigation task.

A GENERAL PRINCIPLE The sequential coding principles for place cells seem in fact to be a strategy generally employed by the hippocampus [266], as it integrates time, space, and memory [58]. Eichenbaum [57] finds that so called time cells in the hippocampus are responsible for sequential organization of events and related variables as well as episodic memories, thereby encoding time. Buzsáki and Tingley [31] agree that the hippocampus primary function is sequence generation but change the perspective and argue it's change that is encoded. Resorting to external measures of time is not necessary, the sequences and their precise ordering [52] are really the point. In a recent study Schuck and Niv [211] were able to show that effects such as replay are also not limited to spatial navigation in rats. They

studied human fMRI in a sequential decision making task and found that complex task-related states were encoded in sequences in the hippocampus and also replayed during the rest phase. Another important observation they made emphasizes that it is the order of sequences and not their timing that must be decoded. The quality of the measured replay in hippocampus was related to the quality of representation in orbitofrontal cortex, which in turn was related to task performance. If hippocampal replay is indeed a training mechanism as this observation suggests, then the trained area must be able to decode both the replay sequence as well as the sequence induced by the active task.

The involvement of the hippocampus in many important tasks is always accompanied by sequential organization of information, suggesting that ordered sequences, often presented with different specific timings, are indeed a fundamental principle of information processing in the brain.

OLFACTORY AND VISUAL STIMULI More evidence can be collected from different sensory areas. Odor representations for example, evolve over 200 ms in sequential, history-dependent activity in locusts [22]. In zebrafish [71], the evolving representation of odors can become more and more dissimilar over time for similar odors, highlighting a potential benefit of temporal coding. Sequences of odors are also represented in the hippocampus, including replay, in humans [270] and hippocampal lesions can lead to failure in tasks where the order of odor events must be recalled in rats [69].

In visual cortex, sequences of neural activation naturally occur in almost the same manner described in the place cell example, except now it's the stimulus that is moving instead of the agent. As a dot moves through the visual field of a rat for example, neurons in primary visual cortex with receptive fields at the stimulus locations activate in sequence [269]. What is particularly interesting is that these V1 neurons can replay the previously presented sequence when a cue is presented. The timescale of replay was compressed and seemed reflective of internal properties of the neural system and not of the speed of movement of the stimulus. Lu et al. [148] were able to reproduce both findings in humans. Repeated exposure of stimulus sequences can lead to adaptations in V1 that enhance cue triggered reactivation [78]. Eagleman and Dragoi [56] found that sequences of random images induce sequence reactivate in the correct order in macaque V4 when the image sequence is expected to occur but does not. This reactivation occurred on the timescale of the original stimulus.

Even the mechanism of time-compressed preplay coding for anticipated events can be found in human primary visual cortex [59], al-

though it seems to be dependent on the involvement of the hippocampus directly [65].

Sequences of visual stimuli in visual cortex are not only dependent on the external movement. Visual information is sampled by saccades that occur on average every 250 ms, creating sequences of visual stimuli on the timescale we previously found important for the representation of sequences [109, 157].

In summary, the brain reflects the sequential organization of our perceived world in its encoding. But it also explicitly produces sequential codes throughout, often with some involvement of the hippocampus. In most cases, the structure of the sequence is remarkably similar. Sequence elements are encoded on a shorter timescale of about 25 ms^2 , and the global sequence are presented on a longer timescale of about 25 ms^3 . However, this second timescale can vary widely, either because of the external stimulus or because the sequence was internally compressed, often to about 100 ms. These timescales seem to be remarkably consistent across different species [30].

The key information carrying feature of the sequence is the order of its elements and not their relative timing. This is illustrated clearly by the independence of linguistic content from the speaker's words-per-minute, the independence of a path taken through a maze from an agent's movement speed and generally by compressed replay of sequences. The information that must be communicated is the scale-free rank-order of the sequence elements.

From the perspective of a computing device that must process this information, for example a neural circuit, the input signal comes with its own private and changing clock, its own time. But it still must be able to compute a response. Computer engineers spend a significant amount of time synchronizing operations and inputs with the help of shared global clocks, neurons on the other hand do not share a clock with their inputs. Further, their own internal clock, the membrane time constant, is simply too fast to keep distinctly in memory more than a single sequence element. Therefore, performing any operation on an ordered sequence of unaligned inputs is a deeply difficult problem. The abundance of sequential codes with their own clocks in the brain gives an indication of how important a solution of this problem is.

We suggest that an ideal candidate solution should maintain a stable memory of past inputs for a just long enough amount of time that the system is largely independent of the various and varying clocks of its input. The extended plateau potential in the dendrites of single neurons could implement exactly this mechanism on a neural level. In the next section, we examine the neurobiological underpinnings

² Which would correspond to a gamma phase.

³ which would correspond to a theta phase.

of dendritic plateaus and formulate the assumptions that allow us to formulate a qualitative model of plateau-based computation in the next chapter.

2.2 THE NEURAL BASIS OF SEQUENCE PROCESSING IN SINGLE NEURONS.

In computational and theoretical neuroscience, models of single neurons must be sufficiently simple to learn about and understand the computation the neuron performs. They must make assumptions about what is considered a critical component of neuronal dynamics to be considered, and what can be ignored. For example, when McCulloch and Pitts [161] examined what nets of neurons can principally compute, they started by clearly stating a set of assumptions based on the then-known neurophysiology:

The activity of neurons is an “all-or-none” response to some fixed number of inputs within a “period of latent addition” independent of previous excitations of the neuron. Delays are only significant at the synapse, and any inhibitory activation prevents the excitation of a a neuron. Unlike in many theories today, learning networks was not yet core to their approach, so they assumed that the structures of their nets would not change with time and focused on what could principally expressed in nets of neurons.

For many, the inclusion of learning in a theory of neural nets represents the origin of all modern neural network theory. Hebb [94] introduced the learning principle *What wires together, fires together*, now named after its inventor (“Hebb’s Rule”), in his important first attempt to formulate a theory of neural computation that explains how behavior results from processing of cell assemblies in the brain. A few years later, Rosenblatt [205] introduced the *Perceptron*, a first implementation of Hebb’s theory that computed the adaptation of synapses in response to input. The Perceptron was able to learn to recognize patterns and kickstarted neural networks as a model for artificial intelligence. Neurobiologically speaking it was essentially still based on the assumption McCulloch and Pitts made some years earlier, except networks now changed over time and learning became central to this line of research.

COINCIDENCE DETECTION AND PLATEAU GENERATION In parallel, many more details about the physiology of neurons were discovered and continue to be discovered to this day. Firstly, following the investigation into the biophysical implementation of spike initiation by Hodgkin and Huxley [99], significant effort has gone into constructing simplified dynamical models that can capture the range of spiking behavior in cortex [110]. Accurate models often do depend-

ent on previous activity of the neuron [21].

Secondly, on the input side of the neuron, details of the mechanism of how one spiking neuron can elicit an excitation in another neuron's membrane potential were discovered. AMPA receptor (AMPAr) channels [101, 102] that are activated by pre-synaptic glutamate release become conductive to a mixture of positively charged ions. This leads to the excitatory post-synaptic potential (EPSP) that can sum and cross a neuron's response threshold. Different stereotypical forms of the EPSP elicited by AMPAr channels are often included in theoretical neuroscientific models of neural computation as kernel functions [117]. Each synapse is a collection out of multiple AMPAr channels, and the weight one neuron assigns to an input roughly corresponds to the number of AMPAr per synapse which changes the height of the EPSP.

Inhibitory synapses are activated by the GABA neurotransmitter and are permeable to Cl^- ions. Their effect, the inhibitory post-synaptic potential (IPSP) counteracts the excitatory response [118, Chapter 10]. The shunting inhibition effect can lead to both a subtractive and divisive modification of the post-synaptic potential [89].

Of particular interest for the generation of plateau potentials is the NMDA receptor (NMDAr) channel that is co-located with AMPAr channels on dendritic spines – small, mushroom-like protrusions along the dendritic tree that form excitatory (Type II) synapses by connection to a presynaptic terminal [9]. It has a comparatively slower gating dynamic [134] and only opens after both pre-synaptic glutamate attached to the receptor and a blocking Mg^{2+} ion has been dislodged from the channel [85, 169, 180]. This happens when the local membrane potential is large enough, for example because a number of neighboring synapses generated coincident EPSPs in response to spiking input [145]. A high membrane potential without local EPSPs is not sufficient to trigger a plateau, as this would indicate that no presynaptic glutamate was released – no AMPAr channels are open – and one of the conditions for NMDAr activation has not been met. Experimental as well as simulation studies report that the required number of spikes in a volley of coincident EPSPs is 4-20 or even up to 50 coincident pre-synaptic spikes within 1 ms to 1 ms, depending on the location along the dendritic tree [14, 76, 77, 145]. NMDAr channels are particularly permeable to Ca^{2+} ions, which means that the dendritic response to such local, coincident spike input can be seen in in-vivo experiments with two photon calcium imaging [236]. The channel is also permeable to other ions and its opening therefore triggers a massive influx of different ionic currents that lead to a full depolarization of the local dendritic membrane potential. The global effect of this processes on membrane potentials at the soma cannot be captured with a model that simply sums the contribution of individual incoming spikes [164], especially so when spikes are temporally syn-

chronized [12]. The isolated NMDAr response is reported to last for around 25 ms [201], but in-vivo recordings show that voltage-gated calcium channels inside the dendritic membrane can be subsequently activated [230] which prolongs the active dendritic response. The depolarization can last from tens to hundreds of milliseconds [156] and is what we and others refer to as a dendritic plateau potential. Plateau potentials are ubiquitous in the nervous system [4, 183] and are our candidate solution for memory traces that are stable for a long time. Nevertheless, they are often not included in computational neuron models (see [chapter 4](#) for models that do).

Strategically placed GABAergic inhibitory synapses also interact with active dendritic trees as they can control dendritic excitability [82] or gate specific dendritic signals from reaching the soma altogether [172]. Their direct effect on plateaus can be dramatic and interrupt or prevent the generation of plateau potentials altogether [47, 53, 121].

This gives us the first cardinal ingredient for computation based on plateaus. Coincident spikes lead to coincident EPSPs that can generate a plateau response on a significantly longer time scale. Inhibitory synapses interact with this process by preventing or disabling plateaus.

UNRELIABLE SYNAPSES Neurons signal their spiking activity to other neurons by releasing neurotransmitters such as glutamate or GABA at pre-synaptic terminals – at least in the case of chemical synapses. These lead to the activation of post-synaptic channels and electrical activity in the receiving neuron which may be used for computation. However, the presynaptic neurotransmitter release is in fact stochastic. The process is described by the “quantal” theory of neurotransmitter release [32, 234] which treats each transmission of a spike as a probabilistic event with probability p_r . In hippocampal synapses, for example, p_r has a distribution with median 0.22 across all synapses [18]. The variance of the distribution is largely explained by the position of the synapse in the neurons dendritic tree. Nearby synapses that can collectively generate a plateau potential have much more homogeneous release probabilities.

In popular firing-rate models of neural computation [43, Chapter 7] this fact has little significance. From the perspective of each neuron, it merely changes the rate of an incoming connection. Whether a rate model which assumes that inputs aren’t very correlated is appropriate is intimately connected to the question of whether the neuron integrates synaptic inputs or detects coincidences between them. The latter of course assumes that inputs are in fact correlated significantly. For pyramidal neurons in particular, this assumption is traditionally hotly debated [125, 228]. However, there is little doubt that plateau potentials are responses to coincident spike volleys. In fact, there

is convincing evidence that spines along the dendrite form clusters whose distinguishing feature is that they regularly receive coincident spike input [241]. These are the ideal conditions for the generation of an active dendritic response. Therefore, we cannot simply absorb the stochastic transmission of spikes into another variable of the model. Instead, the individually stochastic transmission of spikes at unreliable synapses is the second fundamental ingredient for computation based on plateaus.

FUNCTIONAL COMPARTMENTALIZATION IN DENDRITIC TREES

Aside from the properties of spike transmission from pre- to post-synaptic neurons via chemical synapses, we've also learned much more about how information is passed around in the dendritic tree. The post-synaptic potential at a synaptic site in response to a pre-synaptic spike must contribute to the generation of a spike at the axon hillock, proverbial miles away at the soma, by travelling through the extensively branched dendritic tree towards the soma of the neuron. Rall [199] was able to correct membrane time constants by taking into account the properties of this dendritic cable and proceeded to develop a full theoretical model of dendritic transmission [200]. One important finding that turned into an assumption for point-neuron models in theoretical neuroscience is that branching dendrites can be reduced to an "equivalent cylinder" under certain conditions⁴. All synapses can now be thought of as connecting to a single, unbranched cable and the only difference between them is that the EPSPs at distant synapses are attenuated so heavily that barely any signal arrives at the soma [238]. This can be accounted for by a principle called synaptic democracy [91] in which distal EPSPs are simply appropriately larger to compensate for the attenuation. The effect has been observed in experiments [154], although it may be a neuron specific phenomenon [220] and more distant synapses may rely on different effects such as dendritic spikes to contribute to computation at the soma [176]. Therefore, it is assumed in many point neuron models that all synapses can be treated equally and as part of one, single neuron compartment no matter where they might be positioned on the dendritic tree [27].

However, many neurons in the brain see little point in adhering to the terms and conditions of the "equivalent cylinder" model and the propagation of membrane potentials instead is intimately dependent on specific properties of the branching points in the dendritic

⁴ The rule is that the diameter of the parent branch (0) and the two parent branches (1) and (2) follow $d_0^{\frac{3}{2}} = d_1^{\frac{3}{2}} + d_2^{\frac{3}{2}}$. This leads to impedance matching at the branch point and allows for a complete collapse of the dendritic branching structure onto a single cable, if the input and membrane resistances are also the same in all branches, terminal points end in the same boundary condition, and all terminal branches have the same electric distance from the origin in the main branch [124]

tree [231]. For example, an impedance mismatch at these branching points in the dendritic trees of various retinal ganglion cells can lead to distinct, electrically isolated functional subunits [123]. Each subunit has roughly equal local membrane potentials throughout but its membrane potential is only weakly coupled to that of their neighboring units. A similar independent computation subunit is found in thin dendrites of neocortical pyramidal neurons of rats and can add an additional layer of non-linear integrators to the neuron [192]. Branco and Häusser [17] identify functional subunits in individual dendritic branches and suggest that these may constitute the “atomic unit” of computation in neural systems. Which section of the dendritic tree is electrically compartmentalized and can therefore act as a functional subunit does not have to correspond single branchlets. Wybo et al. [267] present a method that can identify a range of different compartmentalizations for different neurons and dendritic trees. They observed that the specific topology of subunits may even be dynamically modified depending on the input statistics and shunting inhibition.

Losonczy, Makara and Magee [146] distinguish weakly and strongly propagating branchlets that can generate NMDA spikes but differ in how strongly they affect neighboring branches. Plateau potentials can be locally generated in functional subunits and then elevate voltages in neighboring subunits which effectively lowers the threshold for plateau initiation in that subunit [155]. This seems particularly important because the increase number of coincident spikes required to initiate plateaus closer to the soma may render them too unlikely to occur in most input conditions. Even in the absence of functional subunits, plateau potentials and active dendritic processes can attenuate towards the soma along the dendritic tree [129]. This may be sufficient to allow distal plateaus to sufficiently depolarize more proximal sections of the dendrite that can then easily generate a plateau as well [4]. It was shown that single dendrites support direction selective activation of NMDAR channels which implements sequential recognition of patterns [16].

How dendritic plateaus influence the generation of other dendritic plateaus is the last important ingredient in a model computation for plateau-based computation. Plateaus are generated locally in and do not actively propagate out of what we simply call dendrite segments. Dendrite segments are weakly coupled to neighboring segments, either by branching that supports compartmentalization or simply by distance and attenuation. Plateaus in one dendrite segment therefore lead to sub-threshold depolarization of neighbors which can enable them to initiate a plateau without requiring an exorbitant number of synchronous input spikes.

As a side effect of this interaction of plateau potentials and the asymmetry of attenuation along the dendritic tree, we assume that plat-

eaus interact this way in a directed fashion towards the soma. In the reverse direction towards the distal regions of the dendrite, plateau potentials would passively depolarize segments. Because plateaus are all-or-none responses, as soon as one segment initiates a plateau the feedforward signal does not reflect whether any more distal segments initiated a plateau at some other point in time.

In addition to investigations into the underlying biophysical mechanism, active dendrites have now been firmly established to play an important role in a range of behavioral tasks. They may even contribute significantly to surface recordings such as EEG, as the associated calcium spikes in dendrites are detectable even through the scalp [239].

Briefly, Xu et al. [268] show that global Ca^{2+} produced by dendritic plateau events can be observed in an object-localisation task when mice received both whisker sensory input and primary motor cortex activity. Several subthreshold calcium events corresponding to orientation specific synaptic input can also be found widely distributed throughout the dendritic tree, while events for different orientations are interspersed. This indicates that the imaged neuron was able to code for multiple orientations of a stimulus via computation in its dendrite [116]. Active dendrites in cortex are further directly involved in perceptual detection of tactile and whisker deflection sensory events in mice [240, 242]. Kerlin et al. [119] were able to image the soma and 300 μm of contiguous dendrite in mice during a tactile decision-making task. They found that the dendritic branching structure compartmentalized calcium signals in a task-associated way, including the activation of entire branches and only segments of a branch. This emphasizes that understanding the computational role of dendritic plateaus is not only important as an elegant candidate solution to the range of multiple timescale sequence detection problems reviewed above, but because they are fundamentally involved in neural processing.

In summary, we model the computation enabled with dendritic plateau potentials based on the following assumptions:

- (i) Coincident excitatory input in a dendrite segment triggers plateau potentials in a dendrite segment, if the dendrite segment is sufficiently depolarized by other dendritic plateau activity.
- (ii) Plateau potentials are all-or-none responses that last significantly longer than post-synaptic changes in response to spiking input.
- (iii) Inhibitory spikes prevent or interrupt the plateau process.
- (iv) Spikes are transmitted stochastically.

Additionally, we assume the computational properties of SDT computation are not fundamentally changed by considering additional detail:

- (v) We can ignore passively backpropagating plateaus and only consider computation in the direction towards the soma.
- (vi) We can ignore sub-plateau-threshold interactions between dendrite segments.

SEGMENTED DENDRITIC TREES

Sequential patterns of spiking activity are ubiquitous in the brain. In many tasks, it is important to distinguish the order of each of the sequence elements whereas the specific relative timing of each sequence element is not. We hypothesize that plateau potentials in the active neural dendrite are uniquely suited to this task and present a much simpler solution than precisely configured network dynamics. Based on the assumptions derived from in-vivo and in-vitro measurements presented in the previous chapter, we define a qualitative model to test this hypothesis. In it, the dendritic tree is functionally compartmentalized into dendrite segments that generate a plateau potential response to coincident, excitatory spiking input.

We are particularly interested in capturing the interaction between events on two timescales: short, spike-triggered excitatory post-synaptic potentials and long dendritic plateau potentials as a local response to coinciding spiking input. More detailed subthreshold dynamics in each dendrite segment are not considered in this thesis and their interaction with the plateau computation process is left to future work. How do interactions of plateau potentials lead to computation that solves the timing invariant detection of ordered sequences? Our model enables us to test the presumption that plateau potentials solve sequence detection in an example experiment with sequentially activated place cells. In the following chapter, it will also enable us to understand the contribution of dendritic plateau potentials to neural computation.

3.1 THE SDT MODEL

Firstly, we model the structural relationships of dendrite segments along the biological dendritic tree in a tree data structure (Fig. 1). In it, each segment's parent is the one neighboring segment connected in the somatic direction, and all child segments are connected in the distal direction. We define a segmented dendritic tree (SDT) recursively as the current segment s and a list of subtrees, each another SDT, we call its children.

$$\text{SDT} : s [s[1], \dots, s[n]] \quad (1)$$

Each subtree $s[i]$ again contains a segment and a new list of subtrees of this node's children. The current node, segment s , is called the parent of its children and the recursion ends when the list of subtrees is empty. These segments are called leafs of the tree structure and cor-

respond to distal regions of a neuron's dendritic tree. The recursion starts at the root node of the tree, the segment in the tree that has no parents. In our model, the root of the SDT contains the soma and proximal regions of the neuron.

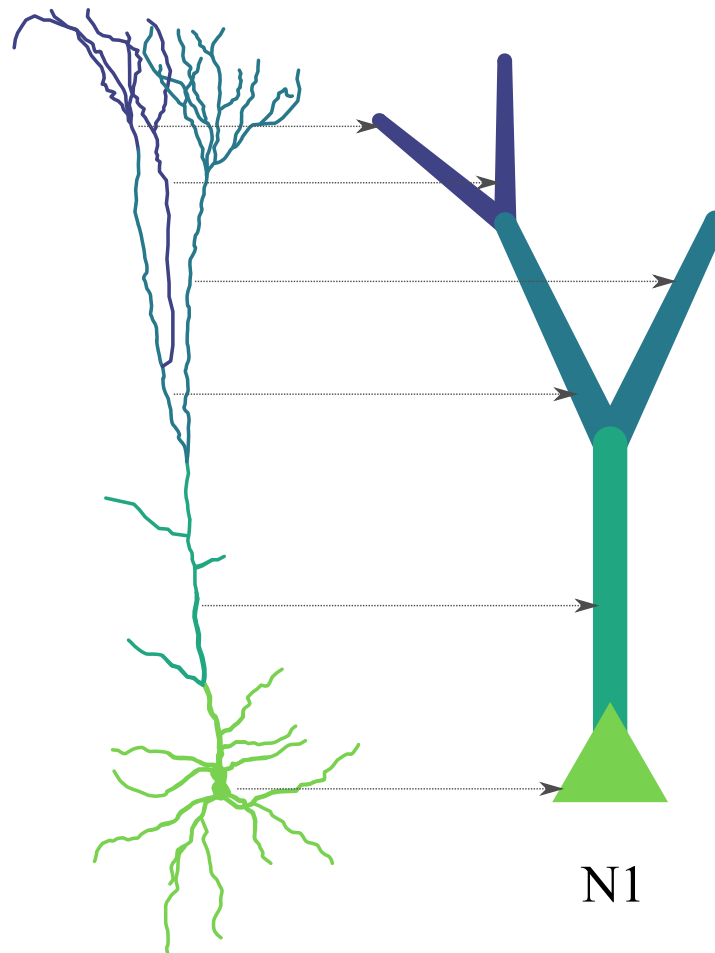


Figure 6: SKETCH: AN SDT SIMPLIFIES THE STRUCTURE OF A NEURON'S DENDRITIC TREE. Connected regions of the dendritic tree that can initiate plateau potentials map onto segments in the SDT model. Each of these regions is weakly coupled to connected segments. This is represented by the parent - child relation in the SDT.¹

How does a segmented dendritic tree correspond to the biological structure of a neural dendritic tree? Segments are connected regions of the dendritic tree that can cooperatively initiate a plateau and are electrically compartmentalized from other segments in the dendritic tree. As we have reviewed, plateau initiation zones do not necessarily correspond to individual branches. In Figure 6 we have sketched one example pyramidal neuron and colored in how connected regions of

¹ Image Credit: the modified pyramidal neuron was originally created by the wikipedia user Fabuio under a Creative Commons license. Original work at: https://en.wikipedia.org/wiki/Pyramidal_cell.

the dendrite may map onto the simplified segmented dendritic tree structure.

Next, we model how an isolated SDT segment s responds to spiking input. A segment is itself similar to a spiking point neuron in that it generates a non-linear response when the membrane potential crosses a threshold. But instead of generating a short spike, a dendrite segment generates and maintains a long plateau response. Here, the local post-synaptic potential V_s due to local synaptic currents is the difference between the excitatory post-synaptic potential (EPSP) and the inhibitory post-synaptic potential (IPSP):

$$V_s(t) = \underbrace{\sum_{q \in Q} \sum_{i=1}^N w_q \xi_q^i \kappa_E(t - t_q^i)}_{\text{EPSP}} - \underbrace{\sum_{r \in R} \sum_{j=1}^M w_r \kappa_I(t - t_r^j)}_{\text{IPSP}} \quad (2)$$

Q is set of excitatory synapses at the segment, and each synapse $q \in Q_s$ has a synaptic weight w_q . Spikes arrive at these synapses at spike arrival times $t_q^i \in T_q$ and are transmitted stochastically with probability p_q . The binary random variable $\xi_q^i \sim \text{Bernoulli}(p_q)$ indicates whether transmission was successful for the i -th spike arriving at synapse q at time t_q^i . The effect of one successfully transmitted spike on the local membrane potential is described by a response kernel κ_E . Different kernel responses such as an exponential or double-exponential ("alpha") response have been proposed in literature [117]. We will emphasize the different duration of synaptic responses and the active plateau response on dendrites and therefore choose without loss of generality the very simple rectangular response kernel with duration τ_E :

$$\kappa_E(t) = \begin{cases} 1 & \text{if } 0 \leq t \leq \tau_E \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

R is the set of inhibitory synapses at the segment, and each synapse $r \in R$ has a synaptic weight w_r . Spikes arrive at inhibitory synapses at spike arrival times $t_r^j \in T_r$. The synaptic response is again captured with a rectangular response kernel with duration τ_I specific to inhibitory synapses:

$$\kappa_I(t) = \begin{cases} 1 & \text{if } 0 \leq t \leq \tau_I \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

Note that the interaction of the EPSP and IPSP in equation 2 is linear, even though the effect of active inhibitory channels on both the generation and continued maintenance of plateau potentials is strikingly nonlinear. We use the linear form of the membrane potential

and choose w_r appropriately large such that inhibition can gate plateau activity by preventing a threshold crossing. This can capture the non-linear effect of inhibition on the threshold process of plateau generation in an uncomplicated way.

Note also that all excitatory and inhibitory spike arrival times must individually be at least the duration of one response kernel apart, because our model doesn't accurately capture repeated high frequency stimulation of single synapses. This means that: $|t_q^i - t_q^j| > \tau_E \forall i \neq j$ and $|t_r^i - t_r^j| > \tau_I \forall i \neq j$.

To generate a plateau response, the first condition is that $V_s(t)$ crosses a segment specific synaptic threshold θ_s : $V_s(t) \geq \theta_s$. This indicates a high post-synaptic potential but also sufficient glutamate at the NMDA receptors of excitatory synapses, which are the necessary conditions for NMDA receptors to open.

In leaf nodes of the SDT, corresponding to the most distal regions of the dendrite, this first condition would be enough to elicit a plateau response in the dendrite segment. But in the general case we must also consider attenuated plateau input, $V_d(t)$ from a segments children.

$$V_d(t) = \sum_{k=1}^N \omega_k s[k](t) \quad (5)$$

The second condition we required to trigger a plateau potential is sufficient dendritic input which we model with a second threshold: $V_d(t) \geq \theta_d$. For leaf segments, θ_d is simply 0. We can then characterize the typical plateau response by its onset and shutoff times according to a chosen plateau duration τ_P :

$$\begin{aligned} T_i^{\text{on}} &\equiv \min_t t > T_{i-1}^{\text{off}} : V_s(t) \geq \theta_s \wedge V_d(t) \geq \theta_D \\ T_i^{\text{off}} &\equiv T_i^{\text{on}} + \tau_P \end{aligned}$$

This initially is highly similar to the stereotypical rectangular synaptic response kernels k_E and k_I . But, because inhibition can interact with plateau potentials after they have already activated, the plateau response can be much more heterogeneous. We therefore have to consider inhibitory spike arrival times a second time:

$$\begin{aligned} T_i^{\text{on}} &\equiv \min_t t > T_{i-1}^{\text{off}} : V_s(t) \geq \theta_s \wedge V_d(t) \geq \theta_D \\ t^{\text{off}} &\equiv \min_t t > T_i^{\text{on}} : t \in \bigcup_{r \in R} T_r \\ T_i^{\text{off}} &\equiv \min \{T_i^{\text{on}} + \tau_P, t^{\text{off}}\} \end{aligned} \quad (6)$$

where t^{off} is the earliest possible inhibitory spike after the plateau was initiated. We can now define the output of the segmented dendritic tree.

$$s(t) = \begin{cases} 1 & \text{if } \exists i : T_i^{\text{on}} \leq t \leq T_i^{\text{off}} \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

Because each segment can have multiple children, we might find ourselves in the situation that a parent segment enters the plateau state because one of its children generated a plateau before, but all other children are not yet in a plateau state. Because the dendrite leads to asymmetric attenuation, we can generally say that the effect of parents on children is much stronger. Therefore, children inherit the plateau state of their parent segments.

Moreover, equation 6 does not trigger a plateau if the segment is already in a plateau state. From biology we know that plateaus may be prolonged by additional input but not indefinitely. Hence, we can consider τ_P a prototypical plateau length and keep it simple.

The model is summarized in Figure 7. All model components relate to properties of a biological pyramidal neuron and qualitatively capture the behavior as it relates to plateau generation. AMPAR channels are responsible for excitatory EPSPs that, together with dendritic input, can displace the Mg^{2+} blocker at NMDA receptors, opening the NMDAR channel and generating a plateau response. Glutamate, the neurotransmitter required for AMPAR and NMDAR activation, is released stochastically at the presynapse. Inhibitory input, represented here as a GABA receptor channel that is permeable to Cl^- , inhibits plateau generation by counteracting the local synaptic contribution to the membrane potential as well as shunting any ongoing plateau potential.

Lastly, we have to adjust the model for the root of a neuron's segmented dendritic tree, the soma. Instead of a plateau response, a somatic spike is generated in response to spiking input. We can model this within our framework by adjusting the condition for a response initiation and replacing equations 6 with:

$$T_i^{\text{spike}} \equiv \min_t t > T_{i-1}^{\text{spike}} + \tau_H : V_s(t) \geq \theta_S \wedge V_d(t) \geq \theta_D \quad (8)$$

The timescale τ_H gives the hysteresis time preventing the neuron to emit another spike. By setting the specific synaptic threshold $\theta_S = 0$ at the soma, this model can produce regular somatic spiking in response to high dendritic activity with frequency $\frac{1}{\tau_H}$, otherwise the dendritic input signals a neural UP-State in an extremely simple spike-response model (the membrane response kernel η is a simple inverted

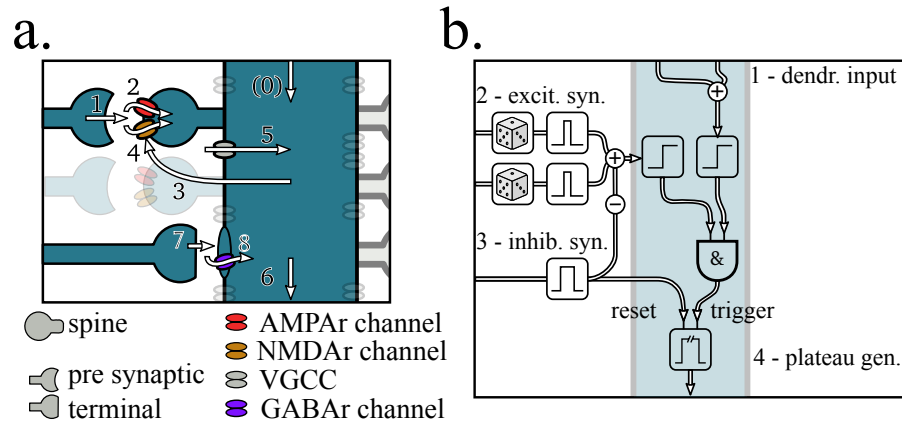


Figure 7: GENERATION OF PLATEAUS IN BIOLOGY AND IN THE MODEL.

a. The sketch shows a sequence of events that can lead to plateau generation and shunting in a region of the dendritic tree. After glutamate is released stochastically at the presynapse (1), AMPAR channels open leading to an influx of ions. If sufficient dendritic input is present (0), the local membrane potential will be large enough (3) to remove the Mg^{2+} block at the NMDAR channel. Because the NMDAR channel has also bound glutamate from the transmitter release that activated the AMPAR channel, it opens leading to a large influx of ions and depolarization of the dendrite in an NMDAR spike. Local voltage-gated calcium channels (VGCC) are also recruited (5), leading to a prolonged plateau potential that is passively transmitted along the dendrite (6). When GABA is released at the terminal of an inhibitory synapse (7), the GABA channel opens. The influx of Cl^{-} ions interrupts the previously stable plateau potential.

b. In our model, these steps are mirrored. If both dendritic input (1) and sufficient excitatory input (2) are available and no inhibitory input is present (3), the conditions for a plateau are met and it is generated (3) and transmitted to parent segments. If an inhibitory spike arrives during a plateau potential, the process is reset.

rectangular function that simply prevents a spike for τ_H seconds). This concludes the definition of the segmented dendritic tree (SDT) model of neural computation. Assumptions based on biological observations and spotlighting processes that directly interact with the generation of plateau potentials lead to a conceptual model that forgoes some amount of accuracy with respect to the moment to moment local membrane potential in order to concisely capture that idea that plateaus interacting in segmented dendrites can implement computation (see also the next chapter). But first, we can verify the qualitative effects of plateau potentials by computing the SDT response to spiking input in experiments that specifically require timing invariance and the detection of sequential order. In the next section we show this in an example of path detection from sequential place cell activity. The model and all experiments are available in a Julia package and published open source (see [section A.2](#)).

3.2 EXAMPLE: DETECTING PATHS FROM PLACE CELL ACTIVITY

A good proof-of-concept example to illustrate how dendritic plateau computation can function in a close-to-real-world example is the detection of sequential patterns in place cells. As we have reviewed earlier, place cells naturally activate sequentially as an animal traverses different locations encoded by different place cell populations during navigation. The traversed path is dependent only on the order of activation, but the timing depends on the animal's movement speed. The path decoding task therefore naturally requires timing invariance and computation on multiple timescales to decode the path from the sequence of place cell activations.

In order to verify that an SDT neuron can solve this task, we set up a computational experiment (see [Figure 8](#)). The environment was a hypothetical rectangular space tiled by place cell populations (20 Neurons each) with 2D-Gaussian receptive fields aligned to a hexagonal grid. Each population emitted spike volleys – a number of synchronized spikes in a small τ_s time window such that the EPSPs of the spikes would overlap – at a rate of 50Hz. Each neuron in a population participated in a spike volley with a probability proportional to the distance of the animal's position to the center of the population's receptive field. Additionally, each neuron would randomly fire with a background firing rate of 5Hz. To simulate a wide range of possible movement through the environment, we drew random paths through the environment by varying the direction of the path and the movement speed along it according to a stochastic differential equation (see [section A.1](#) for additional detail). In sum, the problem has the two distinct time scales typical for sequential information in the brain. Here, they correspond to the fast estimation of the current

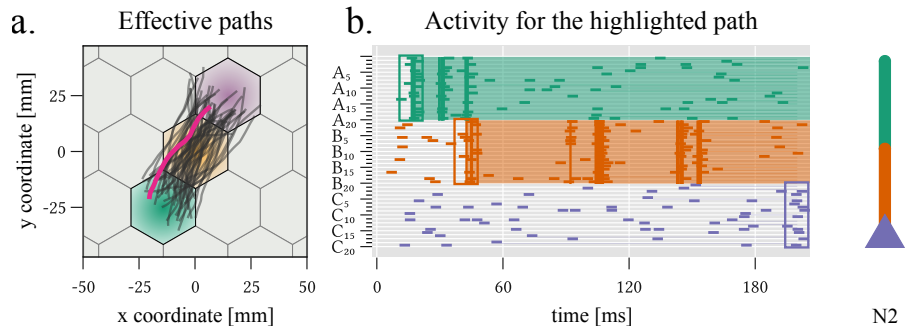


Figure 8: PATH-DETECTION FROM PLACE CELL ACTIVITY. The sequence detection neuron N2 was connected to three place cell populations green, orange, and purple. a. The randomly drawn paths through these three populations that were recognized by neuron N2 are shown. b. A spike train raster for the pink path in panel a. indicates plateau initiation times (boxes) and plateau activation in consecutive segments (shaded region indicates plateau potential). The long plateau memory allowed for significant timing-invariance between initiation of consecutive plateaus.

location via the detection of noisy spike volleys and slow integration of the traversed path represented by sequential activation of different populations. The slow time scale would additionally vary based on the varying movement speeds along the trajectory.

To solve this task, the SDT neuron was set up with three consecutive segments corresponding to three consecutive place cell receptive fields. All weights were set to $w_q = 1$, all transmission probabilities were set to $p_q = 0.5$, and the synaptic threshold was $\theta_s = 6$ in all segments. The configuration of the three segments N2 is illustrated on the right in figure 8. The root segment (purple), or soma, was connected to one child segment (orange) with a dendritic threshold of $\theta_d = 1$ which means that it could only fire if its child segment was already emitting a plateau. The orange segment was set up in the same way but its child segment (green) had no further children and did not require any dendritic input ($\theta_d = 0$). Each segment was connected to a place cell population encoded in the same color and corresponding to a short, straight path through the environment.

Figure 8a shows this environment and the receptive fields of the three place cell populations connected to the SDT neuron. We also show all random paths that resulted in at least one spike emitted by this neuron: All are aligned with the bottom-left to top-right path we expected, giving an indication of the neurons spatiotemporal receptive field. Panel b shows the spiking activity the neuron received when the pink path indicated in panel a was simulated. The shaded background shows the plateaus initiated by coincident spikes, the colored boxes indicate the spike volley that triggered the plateau. Clearly, the timing between the spike volleys that initiated the plateau didn't matter because the plateau potentials remained active for a long time.

Anytime during the plateau potential, the parent segment can initiate a new plateau that indicates that a previous detection, namely that of its child segment, was successful and that enough evidence in form of successfully transmitted spikes out of a spike volley indicated that the animal was close to the center of the receptive field of the connected place cell population. This is exactly the timing-invariant detection of sequences required to solve this path detection problem.

How good is the SDT neuron N2 as a detector for the path "green — orange — purple"? To answer this question, we measured the probability that the neuron emitted a spike in response to systematically varied paths through the environment (Figure 9). Given a prototypical path through the center of all 3 receptive fields along it, we independently varied the speed at which the animal traveled and the center-rotation and parallel translation of the path in space (Fig. 9 top panels). For each sampled path, we ran the experiment and recorded whether the neuron emitted a spike or not. The results are plotted in the bottom panels of figure 9.

Firstly, the response probability to the optimal path peaks at 90% when the animal moved at 0.5 m s^{-1} (Fig. 9a solid line). If the animal moved at a slower speed and the time difference between spike volleys exceeded the plateau duration, the response probability dropped rapidly. If instead the movement speed was three times faster, the neuron would still respond in 30% of the cases. For very fast movement speeds, the encoding place cell populations may fail to emit a spike volley before the animal has already moved towards a new location, resulting in low response probabilities. The longer the animal spend close to the receptive field center of one place cell population, the higher the number of spike volleys that can be detected by each segment. This explains the shape of the response curve over varying movement speeds. It peaks when the speed is just fast enough for plateaus to reliably overlap but the animal also has enough time to detect the current position by sampling multiple spike volleys. Nevertheless, the response probability remains high for a wide range of movement speeds and shows that the receptive field is not specific to precise timing.

An even stronger invariance to timing can be observed if the segment specific threshold to detect volleys is lowered to $\theta_s = 3$ (Fig. 9a dashed line). Despite running at six times the optimal movement speed at 3 m s^{-1} , the response probability remains above 30%.

Instead of varying the movement speed, we can also vary spatial properties of the path and keep the speed constant at the optimal 0.5 m s^{-1} to identify how specific the SDT neuron's receptive field is (Fig. 9b and c). Firstly, parallel translation of the path towards the edges of the receptive field shows expected behavior. Because spike

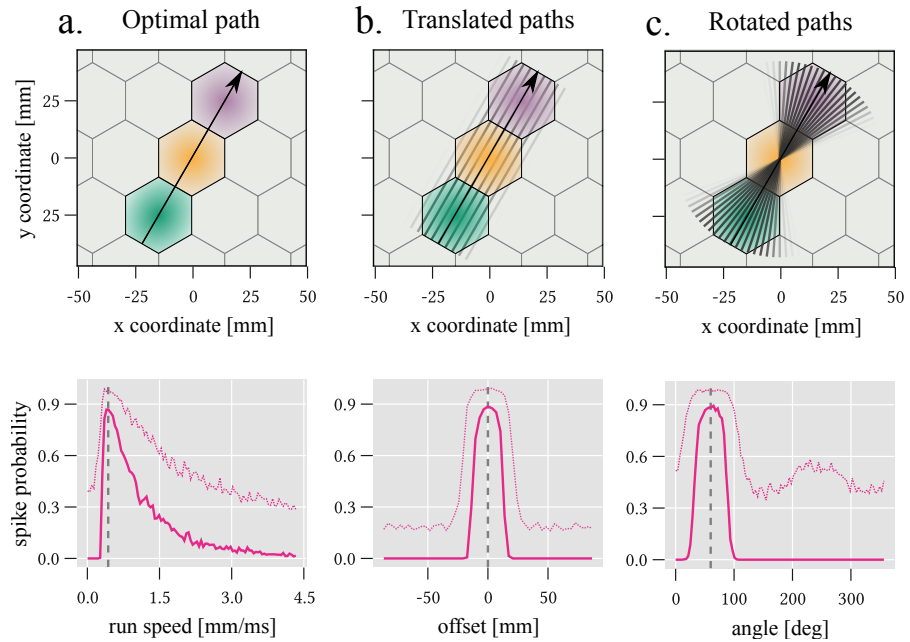


Figure 9: SPATIOTEMPORAL RECEPTIVE FIELDS OF SDT NEURONS. a. The SDT neuron N2 responds optimally to a center path run at a slow speed through the three receptive fields. The response probability falls off slowly for higher movement speeds, indicating a high degree of timing invariance. b. Translating the optimal path orthogonal to the optimal version lowers the response probability. Closer to the center, the neuron exhibits a graded response that falls to zero if the path is completely outside the spatial receptive field. c. Rotation about the center of the path instead of translation shows a similar graded response that falls to zero when just two of the three paths are not part of the running trajectory. In all three experiments, the base response probability can be increased by decreasing the synaptic specific threshold to detect a spike volley. The width of the receptive field also widens.

volleys of large magnitude are much less likely at the edge of each receptive field, the response probability falls off sharply and the receptive field is highly specific. If the threshold is lowered to $\theta_s = 3$, the response probability generally increases and plateaus around 0.27% instead of dropping to 0 for paths with a translational offset larger than 25 mm symmetrically in either direction.

Rotating the path around its central point tells a similar story for the higher threshold of $\theta_s = 6$: A sharp-drop off for deviations larger than 40° around the 60° orientation of the optimal path. Here, however, decreasing the synaptic threshold has a stronger effect and the receptive field is less specific to the exact orientation.

The experiment shows, that the receptive field of SDT neurons can be quite specific and indicate deviations from the optimal stimulus in graded response. It's primary feature is the detection of a completed sequence in the correct order. Reliably detecting just one out of three sequence components (see figure 9 c) was not enough to increase the

neuron's response probability unless the threshold was low enough to also sometimes detect false positive spike volleys. At the same time, the specific timing between plateau initiations does not matter nearly as much and the neuron could still respond to the correct path at vastly different running speeds.

However, repeated measurement of the SDT neuron's binary response to recover the presented graded response is not a realistic scenario for computation in the brain. The long duration of the entire sequential pattern and the fact that plateau potentials are long all-or-none responses would require repeated presentation of the entire sequence on a time scale of $\sim 3\tau_P$. Instead, this suggests that the graded response is encoded in a population or ensemble code instead of a single neuron rate code. Across an ensemble of SDT neurons that have the same spatiotemporal receptive-field, stochastic synapses transmit independently in the same way they transmitted independently over multiple presentations of the same pattern in the experiment above. Therefore, all SDT neurons in an ensemble will respond to the same pattern with the same probability. If ensembles of SDT neurons can coordinate such that co-activated neurons emit spikes synchronously, the graded response is encoded in the magnitude of a spike volleys in the same way we encoded the distance to place cell receptive field centers.

Thus, the SDT neuron's mechanism for sequence detection processes sequences of events encoded as coherent spike volleys and in turn can signal the detection of a sequence as a graded ensemble response which is again encoded in a spike volley. This suggests that computation based on dendritic plateau potentials may in fact be a general computational principle in which case they ought to be seriously considered in models of neural computation. In the next chapter, we therefore investigate if plateau potentials in segmented dendritic trees do in fact introduce new computational capabilities.

COMPUTATION IN NEURONS WITH ACTIVE DENDRITES.

A central idea in neuroscience and cognitive science is to understand the mind by investigating the physical, chemical, and biological processes in the brain in terms of the computational function they implement [68, 214]. To understand how the brain implements cognition as computation [196], one therefore has to understand the brain's computational architecture [127].

The deep connection between computers and the brain is also evident in the work of computer science pioneers. Alan Turing did not only propose the "Imitation Game" to test whether a computing machine can think [252], he also investigated neural networks – "B-Type Unorganized Machines" – built out of initially randomly connected NOT-AND gates that would organize based on experience [252, Chapter 11]. John von Neumann also discussed parallels between the early computers he helped design and the brain [257]. He noted that the brain can be regarded as a digital machine and the neuron as a "typical, digital, active organ". But he also points out that it will be difficult to understand how memory is implemented in the brain, and that synchronicity of inputs is by no means certain and effects the computation.

What exactly can be computed by neural networks? McCulloch and Pitts [161] construed a model of neural computation in which the threshold process that leads to spiking activity in neurons and inhibition that prevents it is regarded as the logical operation the neuron implements. Based on this formalism, nets of neurons, even those that contain cycles (recurrent nets) can be constructed. The operations are synchronous and each neuron can compute its output at $t + 1$ by considering its inputs at discrete time t . Under the assumptions made, the authors argue, neural nets can compute some but not all numbers that can be computed by a Turing machine. Again, the issue lies with the system's memory. While a net without cycles in combination with an external tape memory can compute all Turing functions, memory in a net with cycles is not quite sufficient¹. In fact, Carl Petri showed in his dissertation that an implementation of memory in a physical computer would need to be extensible and operate *asynchronously* to

¹ Franklin and Garzon [70, Chapter 3] recall that Turing-equivalence of neural networks was for a long time assumed in the community of neural network researchers with few recalling a proof. McCulloch and Pitts work seems to be the origin. Proofs that one can construct Turing machines as networks of simple neurons were given by Franklin and Garzon themselves, as well as for example Minsky and Papert [166] or Siegelmann and Sontag [221].

meet this standard [188].

Since then, neural network theory has evolved from Hebb Assemblies [94] to the Perceptron [205], the Multi-Layer Perceptron [207], and Deep Learning [131]. Compared to the earlier work by McCulloch & Pitts, neurons are no longer threshold units and are instead linear-non-linear point neurons that map input vectors of real numbers onto scalars by taking the dot product with a weight vector and applying a non-linear transformation. At the same time, larger and larger networks of these simple units derive their impressive problem solving abilities from the way in which they are connected, the weights between the units, and the way they can learn.

Trained by copious amounts of data and supported by extensive computational resources, deep learning models are able to beat every human player in the games of Go and Chess [222] and produce suspiciously convincing texts on many topics [23]. Connectionism [226] establishes the relation of this type of neural network to a cognitive theory of distributed representation and computation. Consequently, this family of models are considered by many to be the best computational description of brain function we currently have. Apart from their success in artificial intelligence applications, the fact that they seem to at least mimick some of the principles of brain computation supports this connectionist hypothesis [235].

Critics point to the weaknesses that still exist in these models [160], such as distinctly non-human mistakes [7], and some have argued for decades that any connectionist account of the mind is ill equipped to explain the structure of thought we experience [67]. Advocates continue to develop new ways to train and setup networks to achieve artificially intelligent systems that operate more like humans [10, 208]. Nevertheless, the question whether connectionist neural networks can ultimately explain computation in the brain and the emergence of the mind is unanswered.

In light of an increasing number of discoveries illuminating the intricacies of neuronal processes we examined earlier in this thesis, the reliance of our best theories of neural computation on the simple point neuron model may seem foolishly simple. However, the successes of these theories in artificial intelligence speak for themselves and are convincing to many. Further, determining at what level of detail the biological processes must be captured in a computational model of a neuron to accurately represent the computational function of said neuron is in general an open and difficult question [96]. Biological discoveries have thus inspired multiple theoretical studies on the topic, but no consensus has emerged. Ujfalussy et al. [253] concluded that the somatic membrane potential in layer II/III pyramidal neurons can be explained to a large degree by a completely linear statistical model. Adding an additional non-linearity and modelling

the neuron hierarchically improved model accuracy significantly, but further compartmentalization and additional hierarchical layers only lead to minor improvements in model accuracy. Similarly, Li et al. [139] argue that the properties of dendritic integration can be approximately captured in a point-neuron model with a more intricate and interdependent model for synaptic currents. But seminal work by Poirazi, Brannon and Mel [191] suggests that in fact a 2-layer artificial neural network is required to capture the input-output mapping of a single neuron. This implies that the computational, expressive power may be on a similar level, too. These earlier results analyzed the neuron in a static framework. Beniaguev, Segev and London [11] also incorporated the temporal evolution of membrane dynamics and found that capturing the I/O mapping of a cortical pyramidal neuron accurately required a temporally convolutional deep neural network with 5 to 8 layers. This complexity was on the one hand necessary to capture the non-linear, time varying response exhibited by pyramidal neuron, largely attributable to NMDA-mediated dendritic processes. On the other hand, it suggests a significant increase the computational power of single neurons compared to point neurons.

If active dendritic processes indeed have a large influence on the input - output mapping of neurons, then we should find that they add something fundamentally new to the computational function of single neurons and neural networks. Poirazi, Brennen, and Mel's work [191] is still a classic in this department and ascribes a flexible and more expressive non-linear function approximation capability to single neurons. From the computational perspective on neural networks, this doesn't introduce any new capabilities to the model because each node in the more complex neuron can also be replaced by a more traditional linear-non-linear point-neuron leading to admittedly large and specially constructed but otherwise classical neural network.

More recent theoretical models address dendritic processes in three distinct categories: Backpropagation, local learning, and sequence processing.

Backpropagation [207] is the key algorithmic ingredient in the success of deep learning [131], the modern incarnation of connectionism and cybernetics. Because a direct biological implementation of backpropagation in biological neural networks is implausible, one hypothesis is that the credit assignment problem solved by backpropagation in artificial neural networks may be solved by feedback connections that terminate in segregated sections of the dendrite [202]. Plateau potentials achieve the overlap between input and feedback signal [87]. This influences the local plasticity at each neuron via the credit assigned in the feedback pass. The mechanism can elegantly be used for other learning paradigms such as reinforcement

learning [141], but is at odds with the finding that dendritic plateau potentials are the primary driver for synaptic plasticity at the plateau initiation site in the absence of feedback from other parts of the neuron [90, 143]. Fundamentally, these models aim to approximate deep learning in models that are closer to biology and therefore cannot introduce any new functionality.

Urbanczik and Senn [254] introduce a model where plasticity in one dendritic compartment is dependent on the difference of dendritic and somatic voltage. Because the soma can also be driven by fixed synapses (nudged), different learning schemes can be implemented with this rule. They extend the concept and enable prospective learning by increasing the learning window when the soma is nudged to find earlier and earlier predictions for these events [20].

In a larger neural network scheme, Illing et al. [105] use a similar idea and are able to successfully train neural networks without back-propagation. These models, too, fundamentally rely on the function approximation analogy in their feedforward computation. However, their novel approach to credit assignment certainly alters the perspective on how biological neural networks may find good configuration without relying on a global, task specific loss that locally changes synapses based on their contributions to this global measure.

Lastly, Hawkins and Ahmad [92] show that dendritic plateaus can in fact introduce fundamentally new computational functions into neural networks. In their work, pyramidal neurons enter a predictive UP-states based on input to a basal compartment of their dendrite by neurons in the same layer. A plateau in this department depolarizes the soma and permits the neuron to generate a spike response. The response of an entire layer to input is thus dependent on which neurons are currently in the predictive UP-state. At each discrete time step, the set of neurons currently in this UP-state changes. This endows the entire layer with sequence memory and allows it to be used as a neural implementation of the hierarchical temporal memory (HTM) architecture of information processing based on sequences by the same group [79].

The SDT model presented in the previous chapter can also capture this effect of prolonged depolarization at the soma due to a dendritic plateau potential. But it also permits dendritic UP-states: dendrite segments can depolarize neighboring dendrite segments and enable the generation of new plateaus. It concisely encodes the cardinal processes involved in computation with interacting plateau potentials: stochastic coincidence detection on short time-scales to trigger plateaus, much longer plateau processes that remember coincidence events, and functionally compartmentalized dendritic trees into dendrite segments which enables the neuron to rank-order these events and compute functions on them.

Hence, it is uniquely suited to investigate what the novel contribu-

tions of plateau potentials to computation are and whether they should therefore be regarded as fundamental in computational theories of the brain.

We find that the event-based computation enabled by dendritic plateaus can be regarded as fundamentally symbolic. The structure of the dendritic tree determines the constituent structure of expressions evaluated by the SDT neuron over input sequences rank-ordered in real-time. The stochastic nature of synaptic transmission leads to a probabilistic evaluation of these expressions that is proportional to the evidence encoded in spike volleys by input populations. This suggests that dendritic plateau potentials should indeed be regarded as a fundamental mechanism in neural computation.

4.1 STRUCTURED COMPUTATION AND EVENTS IN SDT NEURONS.

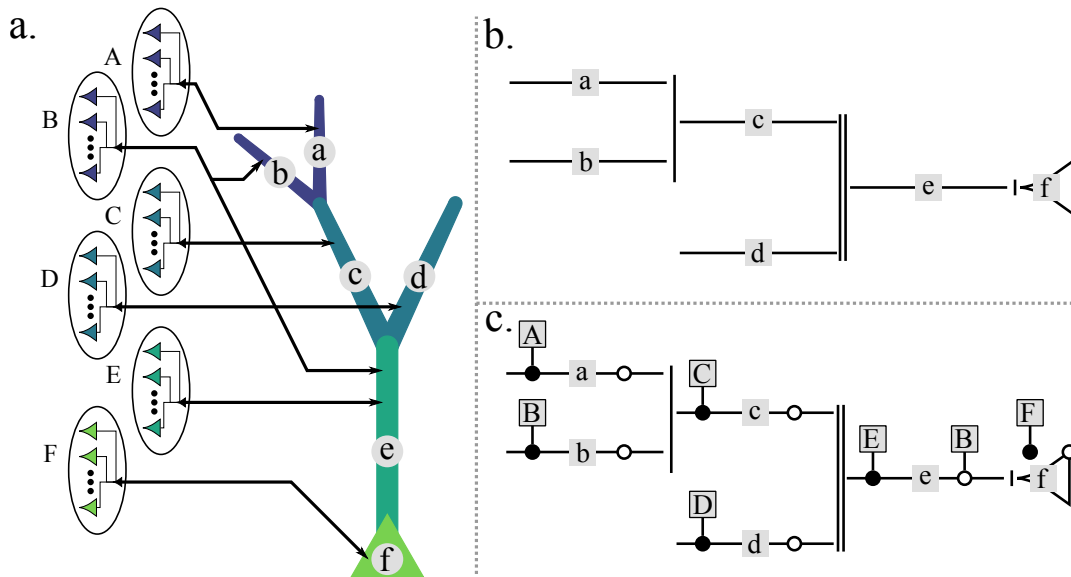


Figure 10: SIMPLE AND COMPLETE REPRESENTATIONS OF SDT NEURONS
 a. Neuron N_1 (from prev. chapter) with named segments a, \dots, f connected to input populations A, \dots, F . b. The structure of N_1 can be captured in a simple diagram where children connect to parents via vertical lines. The number of vertical lines gives the dendritic threshold of the parent segment. Each segment is a named horizontal line. c. Adding filled circles for excitatory connections and empty circles for inhibitory connections to segments allows connections to input populations represented by named squares.

In order to talk about the computational properties of SDT neurons, we must describe their computational function firstly in relation to their internal structure and secondly, how this internal structure relates to external inputs. We will name neurons with different in-

ternal structure N_1, N_2, \dots and so forth.

The recursive definition of SDT neurons in the last chapter (Equation 1) centered on individual segments. However, the global structure of a neuron is difficult to understand from the hierarchically nested lists of children with associated dendritic thresholds. For example, neuron N_1 that is redrawn in figure 10a is technically specified by:

$$N_1 \equiv [f : [e : [d : [], c : [b : [], a : []]]]] \\ \&[\theta_d(f) = 1, \theta_d(e) = 2, \theta_d(d) = 0, \theta_d(c) = 1, \theta_d(b) = 0, \theta_d(a) = 0]$$

where individual segments are lowercase letters and $\theta_d(s)$ specifies the dendritic threshold of some segment s . This encodes more information about N_1 because the dendritic thresholds were not captured by the sketch, but it is hardly useful.

Instead, we will specify the structure of SDT neurons by simple graphical representations. Figure 10b shows how this is done for neuron N_1 : Each segment is a horizontal line, named by a lowercase letter. Children connect to parents from left to right via vertical line segments. The number of vertical line segments indicates the dendritic threshold of the parent segment. In this simple Neuron, all dendritic weights ω_k are 1, we will deal with more complicated branching later. Additionally, we may want to specify which input populations connect to excitatory or inhibitory synapses at each segment. Figure 10a indicates a number of input populations which we have labeled with uppercase letters A, \dots, F that connect to the different segments. The sketch doesn't show which connections are to inhibitory or excitatory synapses respectively. In Figure 10c, we indicate excitatory input connections by linking the named input population to a filled in circle, whereas inhibitory synapses are indicated by an empty circle.

Two types of synaptic events significantly interact with dendritic plateau generation. Firstly, a spike volley event refers to a set of spikes in the afferent population of a segment s such that the sum of their weighted EPSPs exceeds the local, segment specific synaptic threshold θ_s under the assumption that all spikes are transmitted successfully. All spike volley events can be defined iteratively:

$$t_s^{i+1} \equiv \min_t t > t_s^i : \sum_{q \in Q} w_q \kappa_E(t - t_q^i) \geq \theta_s \quad (9)$$

The equation captures threshold crossing times of the local post-synaptic potential (Equation 2) due to the EPSP only and under the assumption that all spikes were successfully transmitted. All t_s^i in Equation 9 therefore fulfill the minimum condition for a plateau potential: If no inhibitory synapse is active and all spikes are transmitted, a spike volley will always lead to a plateau potential. Figure 11 shows how this works in our specific case with the rectangular EPSP kernel κ_E

and time scale τ_s .

The second set of events relevant for computation are inhibitory events. In the SDT model, we have simplified the effect of inhibition to be a plateau potential veto that either prevents a plateau if inhibitory spikes arrive up to τ_i seconds before a spike volley or shuts down the plateau process if inhibitory spikes arrive during an active plateau potential. Therefore, we can simply identify inhibition events with all spikes sent by afferent populations to inhibitory synapses at a segment (see Fig. 11).

$$t_{-s}^{i+1} \equiv \min_t t > t_{-s}^i : t \in \bigcup_{r \in R} T_r \quad (10)$$

We will later see that spike volley events can sometimes turn into inhibitory events if the same population is connected to different synapse types at different segments. In this sense, inhibitory events are spike volleys with just at least one spike, which is always true for spike volley events.

In this simpler world of spike volley and inhibition events on seg-

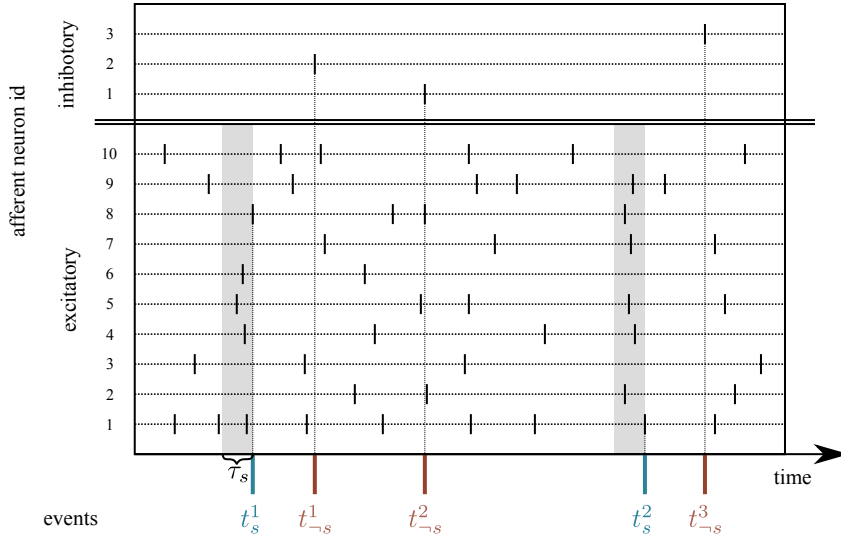


Figure 11: COMPUTATIONAL EVENTS FROM A SPIKE RASTER. Spike volley events (blue) occur when a sufficient number of excitatory afferent neurons have emitted a spike. Here, the threshold is $\theta_s = 5$, all weights are 1 and τ_s is the time scale of the excitatory post-synaptic kernel κ_E . Inhibitory events (red) are identified with spikes emitted by inhibitory afferent neurons.

mented dendritic trees, we can discuss the three main aspects of computation in SDT neurons:

1. Probabilistic synapses enable a graded plateau response probability proportional that reflects confidence in the input.
2. The overlap of plateau responses required for SDT computation rank-orders real-time events such that they can be represented

in an internal ranked time invariant to small perturbations and delays in timing.

3. The structure of the SDT neuron, reflecting the structure of the compartmentalization of the neural dendritic tree, determines structured expressions. Computation in an SDT neuron means evaluation of these expressions.

PROBABILISTIC SYNAPSES. Whether spike volley events can lead to a plateau potential is dependent on the successful transmission of sufficiently many spikes by the stochastic synapses. Whether this stochasticity, or unreliability, of presynaptic neurotransmitter release has a computational function or is instead a consequence of signaling spikes with a limited number of vesicles at chemical synapses [147] is a hotly debated and open question. Proponents of probabilistic codes and Bayesian computation in the brain [122, 193] argue that the brain encodes uncertainty about stimuli or parameters in probability distributions in order to make Bayesian optimal choices given the data available. In circuit models, probabilistic synapses can generate the required variability [170] and sample from distributions encoded in network connections [174]. In SDT neurons, can stochastic responses due to unreliable synapses encode uncertainty about a stimulus or must they be considered additional noise?

Consider a stimulus that is encoded by a population A with N neurons. Segment a is connected to A and evaluates whether the stimulus was present or not by generating a plateau if the post-synaptic potential was high enough. In any spike volley t_a , the uncertainty that the stimulus is present is encoded in the independent probability q of each neuron to fire a spike at this point or not. The spike volley magnitude for the spike volley event $|t_a| = n_a$ is then distributed according to a binomial distribution:

$$n_a \sim f(n_a; N, q) = \binom{N}{n_a} q^{n_a} (1 - q)^{N - n_a}$$

At segment a , each spike in the spike volley is independently transmitted with homogeneous probability p and added with the same weight w ². The number of successfully transmitted spikes m_a is also distributed as a binomial distribution with n_a synapses flipping coins with bias p :

$$m_a \sim f(m_a; n_a, p) = \binom{n_a}{m_a} p^{m_a} (1 - p)^{n_a - m_a}$$

² The assumption that all synapses transmitting the same spike volley have the same transmission probability and weight. This is well supported by the fact that this set of synapses would be subject to the same learning due to local plateaus [90] and observations [18, 241].

The distribution of successfully transmitted spikes given a stimulus uncertainty q , an encoding population size N and transmission probability p then is:

$$p(m_a|q) = \sum_{n_a=0}^N p(m_a|n_a)p(n_a|q) = \sum_{n_a=0}^N f(m_a; n_a, p)f(n_a; N, q)$$

We can rewrite the condition for plateau generation given that we know which synapses have overlapping rectangular EPSPs, i.e. all m_a synapses that receive inputs in a volley, as $wm \geq \theta_s$ which is equivalent to:

$$m \geq \lfloor \frac{\theta_s}{w} \rfloor \quad (11)$$

Because of this, we can assume without loss of generality that $w = 1$ for all synapses and the threshold θ_s at segment a is some integer. The probability that segment a responds by generating a plateau given q is the probability that $m_a \geq \theta_s$, which is given by:

$$p(r|q) = 1 - \sum_{m_a=0}^{\theta-1} p(m_a|q) = 1 - \sum_{m_a=0}^{\theta-1} \sum_{n_a=m_a}^N f(m_a; n_a, p)f(n_a; N, q)$$

where we have dropped terms from the sum in which $m > n_a$, which have probability 0.

We can now plot the stimulus uncertainty or strength q against the probability that a plateau would be generated by coincidence detection (Fig. 12). In sum, the response of the segment is extremely sharp when the transmission probability is 1 and synapses are not stochastic. Lowering the transmission probability significantly leads to a graded response that can be shifted to along the x-axis by adapting the threshold (Fig. 12c). This enables the tuning of the response curve such that the plateau response probability is proportional to the certainty about the input encoded in q and therefore encodes the confidence that a plateau should be initiated [194]. Since the plateau is a long all-or-none response, the stochastic synapse is on the one hand the only way to get create a graded response proportional to the encoded certainty or uncertainty q about a stimulus. As many authors have pointed out (e.g. in Doya et al. [48]), this information is required to make decision and reason under uncertainty [113] – the default for computation in the brain.

However, the long plateau response prevents repeated sampling of the it. After we have discussed the two other aspects of computation, rank-ordering and evaluating SDT structured expressions, we will return to this question and show that probabilistic synapses enable probabilistic computations in ensembles of SDT neurons.

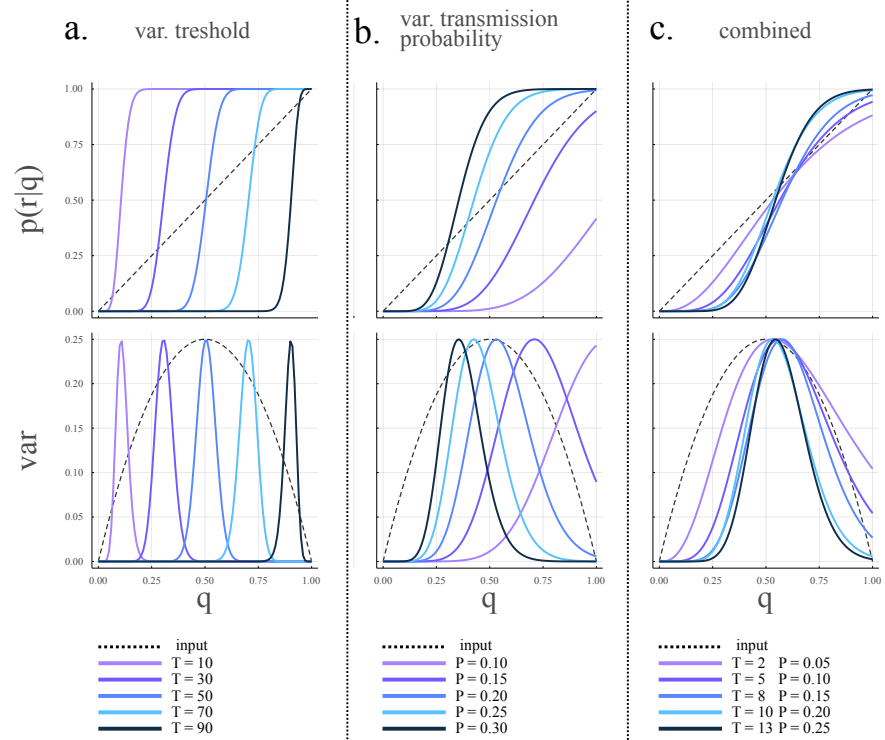


Figure 12: ENCODING AND DECODING STIMULUS UNCERTAINTY. The top row shows the uncertainty q of encoded stimulus Q against the probability of generating a plateau by coincidence detection. Dashed line corresponds to a linear mapping. Bottom row shows the Bernoulli variance $p(1 - p)$ for both the plateau response and the stimulus (dashed line). a. If the transmission probability is 1, the sigmoidal transfer function approaches a step function and the uncertainty of the stimulus is not reflected in the plateau response. Different thresholds are shown, shifting position of the step. The variance is narrowly distributed around the steep section of the transfer function. b. If the threshold is fixed at 10, varying the transmission changes the slope of the transfer function. For low transmission probabilities, even a high confidence in the stimulus results in unreliable responses. The variance widens accordingly with lowering transmission probabilities. c. If both the threshold and probability are varied, the plateau response probability can approach a linear encoding of the stimulus certainty and remains proportional to q in the entire range.

RANK-ORDERING INPUT EVENTS. Which spike volley events ultimately to plateau potentials and therefore contribute to computation in the SDT neuron depends not only on successful transmission of a sufficient number of spikes but also previous activations of dendrite segments. We cannot a priori know which spike volley events lead to plateaus, since each plateau response is probabilistic and different numbers of previous plateaus in child segments may be required to initiate a plateau at any particular segment. Here, we are interested in finding out which spike volley events could potentially contribute

to a computation before we give the expression that is evaluated over these spike volley events.

What decides whether two spike volley events t_a at segment a and spike volley event t_b at segment b are considered in the same computation? Depending on the morphology of the dendritic tree, their plateaus have to overlap in either an ordered manor if one segment is a child of the other, or in an unordered manor if they are the children of the same parent node. Additionally, there must be a continuous path of overlapping plateaus from the soma at a time t at which we are interested to find out which events contribute to the two events by the same rules. Here, t is the query time at which we consider the computation of the SDT neuron and can in principle be arbitrary. Often, we will take t to be the time of a query spike or spike volley at the soma, which would lead to an output spike immediately if the computation in the dendritic tree is successful.

Let's look at an example (Fig. 13). Neuron N_1 , which we've previously used to recognized paths from place cell activations invariant to movement speed, has three consecutive segments, one of which is the soma. If we drive the neuron with spiking input, we can think of spike-volley events and plateau responses sequentially: First, A activates a , then B activates b , then C activates c and a spike is transmitted. Instead, we can ask which spike volleys would be considered for a computation at time t represented by a spike volley at the soma. Recursively, each spike volley can be affected by a spike volley in a previous plateau length. We can draw the connected events in a simple rank-ordered time tree (ROTT) and give each depth of the tree a rank-order. Numbering from left to right, N_1 transforms three events in real, continuous time $a(t_a), b(t_a), c(t)$ into a discretely rank-ordered sequence a_1, b_2, c_3 at t_1 and b_1, c_2 at t_2 .

If segments are instead parallel to each other, events must occur in the same plateau window defined by the spike volley event of a parent segment. These events should get the same rank-order. Inhibition, however, must be treated differently because it interacts only with the local spike volley event and can either precede the spike volley by τ_1 to prevent a plateau potential or locally interrupt any plateau potential during it. It always has the same rank-order as the local spike volley event it is associated with. Neuron N_3 illustrates and explains this (Fig. 14). $a(t_a), b(t_b), c(t)$ is ordered as $a_1, -a_1, b_1, -b_1, c_2$ at t_1 and as a_1, c_2 at t_2 . The example also shows that spike volley events can always double as inhibitory events if populations are connected to different synapses at different segments.

This covers all the core rank-ordering procedure that is inherently performed by dendritic plateau potentials in simple cases. In general, if we are given a rank-ordering of spike volley events due to some

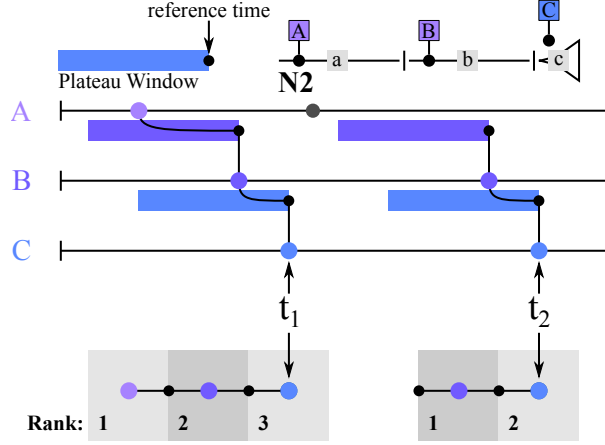


Figure 13: RANK-ORDERING OF SEQUENTIAL REAL-TIME EVENTS. Neuron N1 rank orders spike-volley events in its input populations A and B according to a reference spike at the soma at time t_1 and t_2 . Each spike volley on a segment can recursively consider all spike volleys at its child segment for one plateau duration.

The black path gives the rank-ordering of spike volley events starting at t_1 and t_2 respectively. The final rank-order can be represented by a simple rank-ordered time tree (ROTT) with one rank-order per level according to the depth.

ordering operation O implemented by a neuron, we can say the following is true for any two events a_1 and b_1 :

- (i) For any two events $a(t_a)$ and $b(t_b)$ that we order as a_1 and b_2 , we know $a(t_a)$ precedes $b(t_b)$ by at most one plateau length τ_p :

$$O(a(t_a), b(t_b)) = (a_1(t_a), b_2(t_b)) \rightarrow t_a \in [t_b - \tau_p, t_b]$$

- (ii) For any two events $a(t_a)$ and $b(t_b)$ that we order as $a_1(t_a)$ and $b_1(t_b)$, we know that there must be some interval of length τ_p during which both events occurred.

$$O(a(t_a), b(t_b)) = (a_1(t_a), b_1(t_b)) \rightarrow \exists t : t_a, t_b \in [t - \tau_p, t]$$

- (iii) For any pair of one spike volley event $a(t_a)$ and one inhibitory event $\neg a(t_{-a})$ that we order as a_1 and $\neg a_1$, we know that there must be some interval of length τ_p during which $a(t_a)$ occurred, and $\neg a(t_{-a})$ either occurred before t_a within an inhibitory interval of length τ_I , or after t_a but before t .

$$O(a(t_a), \neg a(t_{-a})) = (a_1, \neg a_1) \\ \rightarrow \exists t : t_a \in [t - \tau_p, t] \wedge t_{-a} \in [t_a - \tau_I, t]$$

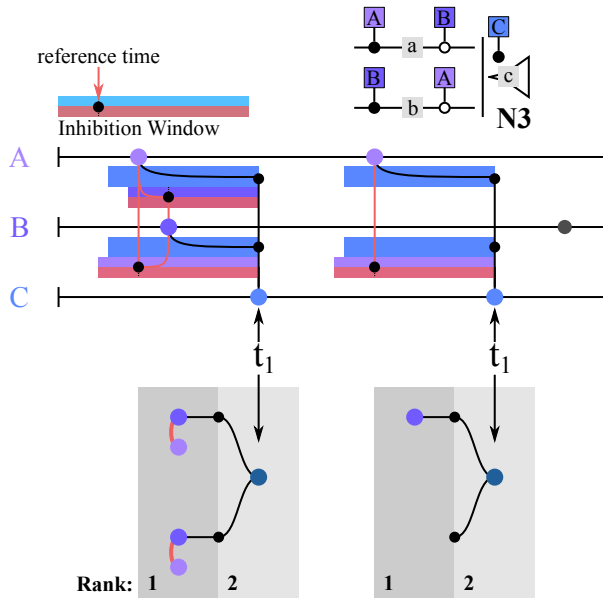


Figure 14: RANK-ORDERING OF PARALLEL EVENTS AND INHIBITION. Parallel events occur during the same plateau window referenced to a spike volley in a parent segment. The rank-ordering path and resulting graph branch at the parent volley event. The window in which an inhibitory event can affect a spike volley event extends into the future of said event, corresponding to a plateau disable. The length of this window stretches from the reference event to the time of the parent segments reference events. If the spike was already send at segment s , inhibition at b no longer has an effect. The same is true for plateaus. It also extend into the past of any reference events for the duration of the inhibitory kernel. The path connecting inhibitory events to spike volley events is painted red. In the ROTT, inhibition is always a parallel event indicated with a red connection to the reference event.

Colloquially, we may summarize that maximal memory length of an SDT neuron is $d(N)\tau_P$ where $d(N)$ gives the depth of neuron N 's SDT, but all events considered in one computation can also occur within a much shorter interval, say $\frac{\tau_P}{10}$. The representation in internal rank-ordered time and the ROTT is the same in both cases. Each level of depth in the SDT can correspond to a layer in the rank-ordered representation. From child to parent, the relation of events is a strict successor relation within one plateau length, whereas siblings must have events in the same plateau interval given by the parent. The effect of inhibition dynamically depends on the context, but only affects the local segment and rank-order directly.

This is not the entire story. When multiple events arrive within the same plateau interval, which ones are we to consider? We give the generalized form of the ROTT by example and return to neuron N_1 together with a possibly more realistic timeline of events by popula-

tions A through E (Fig. 15).

At time t , multiple spike volley events at segments c from population C and segment a from population A must be considered. In the ROTT, we can deal with this by adding an additional branching node, possibility branches, at the rank-ordering boundary that represents multiple possible realizations of the rank-ordering. In practice, only one will actually initiate a plateau, but all must be considered.

Further, we can annotate the rank-ordered spike volley events with their magnitude and original time, so that we can distinguish multiple events assigned to the same rank. For example, the spike volley event t_a due to an excitatory connection from population A and with $n_A(t_a)$ may be written as $a_1[|(t_a)|]$. If we simply number all multiple events as t_{a1}, t_{a2}, t_{a3} , we can mark the set $a_1[|t_{a1}|, |t_{a2}|, |t_{a3}|]$ for the a branch of the rank-ordering. The ROTT and this set of associated spike volley and inhibitory events with a given rank order captures all events considered in the SDT computation at time t .

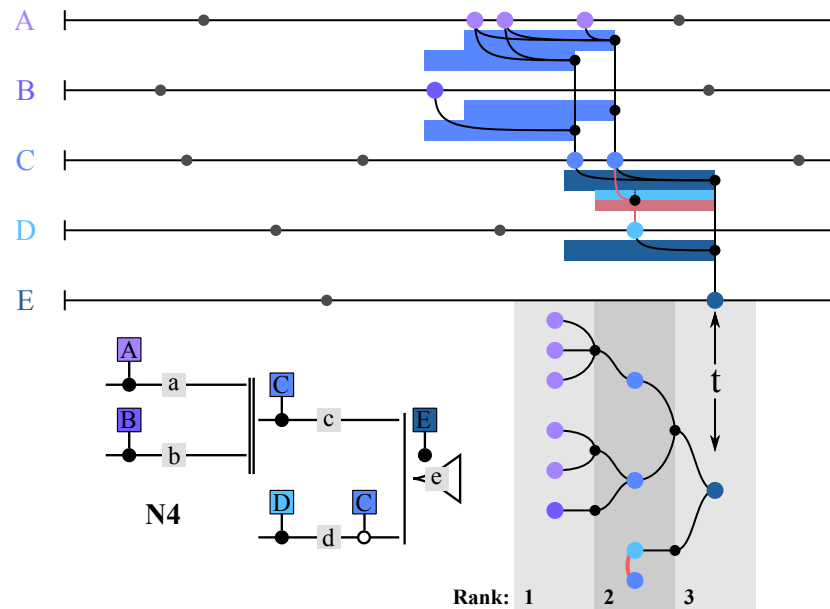


Figure 15: RANK ORDERING OF MULTIPLE EVENTS. If multiple events overlap with in the same plateau window, all must be considered to evaluate the response of the SDT neuron. This introduces a second type of branching point, possibility branches, in the ROTT (black dots at rank boundary) that represents not branching in the SDT neuron, but branching between different possible pasts.

The ROTT gives a good representation of events ordered in time and different possible pasts, but it is easy to imagine that even this representation will quickly run into the limits of feasibility. This is indicative of the fact that describing multiple different asynchronous streams of events in a general fashion is a fundamentally difficult exercise. Event logics [217] and timed automata [3] are two theoretical

frameworks that illustrate the point.

In relatively simple cases, the ROTT does the trick and also preserves the fundamental intuition about computation in trees the SDT model relies on. Each event node can be evaluated independently as *true* or *false* depending on whether a plateau was generated in the deterministic case where all synapses transmit reliably. If they do not, the probability of a synapse driven threshold crossing can be calculated independently. This is the basis for computation across the different segments presented in the next paragraph.

EVALUATING SDT EXPRESSIONS. In SDT neurons, synaptic weights do not have as decisive a role as they do in most neural network models. At each segment, they largely decide the threshold of m successfully transmitted spikes (see 11) and are homogeneous among synapses that frequently contribute collectively to plateau generation via spike volleys. Instead, the coupling of different segments to each other determines the computational function of the neuron in conjunction with coincidence detection at each segment. In the following, we assume that inputs are already available in a rank-ordering and that synapses transmit reliably for now.

For a segment s with child segments $s[1], s[2], \dots, s[n]$ the condition for a dendritic threshold crossing was given by Equation 5 and dendritic threshold θ_d :

$$\begin{aligned} V^d(t) &= \sum_{k=1}^N \omega_{ks}[k](t-1) \\ V^d(t) &\geq \theta_d \end{aligned} \quad (12)$$

The second and third condition for a plateau was the existence of a sufficiently large spike volley and the absence of inhibitory input. As spike volleys and inhibitory events are available in discrete, rank-ordered time we can transform the conditions for plateau generation into a predicate logic expression following a similar argument to McCulloch and Pitts [161].

Let κ_i be the set of subsets of ω_κ such that the sum over it exceeds θ_d . Then, the condition for plateau generation is given by the following predicate expression over spike volley events, inhibitory events and a recursive dependence on expressions of child segments:

$$\mathbf{S}_t = \underbrace{s_t}_{\text{excitation}} \wedge \underbrace{\neg s_t}_{\text{inhibition}} \wedge \underbrace{\sum_{\pi \in \kappa} \prod_{j \in \pi} \mathbf{S}[j]_{t-1}}_{\text{dendritic gating}} \quad (13)$$

where \prod is the logical multiplication (\wedge) and \sum is the logical sum (\vee). Small letters are events that may be part of any particular ROTT, fat capital letters are predicate sentences. If a segment has no inhib-

itory inputs, we simply leave out the symbol and the absence of a particular event evaluates as false.

We can then recursively ascribe expressions to previously seen neurons for which ω_k was implicitly 1 for all k and get very simple formulas:

$$\mathbf{S}(\mathbf{N}_1) \equiv f_t \wedge e_{t-1} \wedge d_{t-2} \wedge c_{t-2} \wedge (a_{t-3} \vee b_{t-3})$$

$$\mathbf{S}(\mathbf{N}_2) \equiv c_t \wedge b_{t-1} \wedge a_{t-2}$$

$$\mathbf{S}(\mathbf{N}_3) \equiv c_t \wedge ((a_{t-1} \wedge \neg b_{t-1}) \vee (b_{t-1} \wedge \neg a_{t-1}))$$

$$\mathbf{S}(\mathbf{N}_4) \equiv e_t \wedge ((d_{t-1} \wedge \neg c_{t-1}) \vee (c_{t-1} \wedge a_{t-2} \wedge b_{t-2}))$$

\mathbf{N}_3 for example implements an exclusive or operation (XOR) on the input populations of a and b at any point in time t in accordance with the rank-ordering constraints given in the previous section. Whenever a neuron \mathbf{N} sends an output, we know that $\mathbf{S}(\mathbf{N})$ was true at that particular point in time.

We can plug in a ROTT into such a formula similarly to how one might plug in a row of truth values from a truth table into a standard predicate logic formula and evaluate whether the neuron would respond at time t given $p = 1$ for all synapses. For each possibility branch in the ROTT, we must evaluate the corresponding subformula for the branch in question and logically sum over all possibilities, starting at the lowest rank. Figure 15 shows how this procedure may work in the deterministic case.

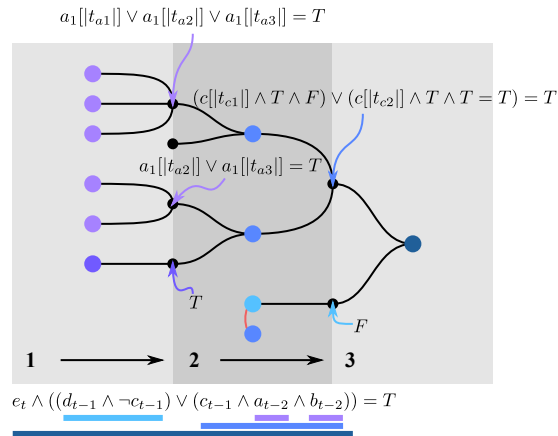


Figure 16: EVALUATING AN SDT EXPRESSION WITH DETERMINISTIC SYNAPSES. Together with a ROTT of time-ordered events and assumed deterministic synapses, the truth value of an SDT expression ($\mathbf{S}(\mathbf{N}_4)$) can be evaluated directly. By evaluating from the first rank and innermost segment expression previous possibility branches can be merged by evaluating the logical sum over all possibilities of the given subformula.

In general, the rank-ordered spike volley events a_{ti}, b_{tj}, \dots aren't evaluated deterministically but are probabilistic symbols associated with plateau response probabilities $p(r||a_{ti}|)$. As we have seen earlier, this probability encodes the confidence that the symbol encoded by population A that send out the spike volley is actually present. Internally, the SDT neuron evaluates deterministic expressions over probabilistic symbols, constrained to the rank-ordering enforced by plateau potentials. The rules by which we can derive the probabilistic expression are simply:

$$\begin{aligned} P(a_{ti} \wedge b_{tj}) &= P(a_{ti})P(b_{tj}) \\ P(a_{ti} \vee b_{tj}) &= P(a_{ti}) + P(b_{tj}) - P(a_{ti})P(b_{tj}) \end{aligned}$$

This is possible because we can consider spike volley events a_{ti}, b_{tj}, \dots independent events that are observed as a side effect of the rank-ordering process. The probability model for N2 for example then is:

$$\begin{aligned} P(N_2) &\equiv P(c_t \wedge b_{t-1} \wedge a_{t-2}) \\ &= P(c_t)P(b_{t-1})P(a_{t-2}) \end{aligned}$$

Inhibition events aren't stochastic, so they continue to evaluate as true or false with probability 1 depending on whether they are existent or not³. The probability model for the XOR neuron N3 then is:

$$\begin{aligned} P(N_3) &\equiv P(c_t \wedge ((a_{t-1} \wedge \neg b_{t-1}) \vee (b_{t-1} \wedge \neg a_{t-1}))) \\ &= P(c_t)(\neg b_{t-1}P(a_{t-1}) + \neg a_{t-1}P(b_{t-1}) \\ &\quad - \neg(b_{t-1} \wedge a_{t-1})P(a_{t-1})P(b_{t-1})) \end{aligned}$$

To merge multiple possibility branches in the ROTT given a probabilistic model, we replace the probability of individual events at possibility branch with the probability of the union over all probabilities. For example, for $a_1[|t_{a,1}|, |t_{a,2}|, |t_{a,3}|]$ the probability of a to generate a plateau is given by $P(\bigcup_i r||t_{a,i}|)$, the union over all possible plateau generating events. Because $P(\bigcup_i r||t_{a,i}|) \geq P(r||t_{a,j}|)$ for any individual j, repeated volleys increase the confidence in the presence of the stimulus associated with population A at the segment a.

Applying these algebraic rules to possibility branches and SDT expression to find algebraic forms of the probability model quickly becomes cumbersome. The insight they provide is that SDT computation is consistent. We were able to draw a direct line from multiple rank-ordered input spike volley and inhibition events to structured

³ Stochastic inhibition causes implementation headaches in the mechanistic model because it leads to stochastically sampled events affecting the model at multiple points. It is however easy to see what the effect of stochastic inhibitory synapse would ultimately be in the probabilistic model

expressions that define the computational function. The entire expression represented by the SDT is probabilistically evaluated with each symbol encoding the confidence that a particular term in the expression is true at the time of computation t .

To read the encoded confidence in the computation, we can make use of a simple network motif and create a neural ensemble out of multiple neurons that represent the same expression. At the same query time t , each of these neurons will have the same independent probability to fire because they received the same spike volley input transmitted by independent stochastic synapses. At time t , the ensemble will therefore emit a spike volley that encodes in its magnitude the confidence the ensemble has in its expression. This exactly the encoding our input populations used to encode certainty in a stimulus. The output of an SDT ensemble can therefore be used as input to another SDT neuron or ensemble.

Each component of the biological mechanism illustrated in [chapter 2](#) and encoded in the model assumptions plays an important, distinguished role in this process. The long duration and sequential gating of plateau processes rank-orders inputs and renders the computation invariant to individual clocks of different input signals. Different degrees and structure of functional compartmentalization between dendrite segments in the dendritic tree and strategically placed inhibitory synapses determine the structure of computation implemented by the SDT neuron. Due to the interaction of coincidence detection and probabilistic synapses, symbols encode confidence in the presence of associated stimuli. As a result, the evaluation of the entire computational expression implemented by the SDT neuron turns probabilistic such that the response probability encodes the confidence in the expression being true over all input symbols.

4.2 SDT NEURONS AND NETWORKS.

The connection to the formalism McCulloch and Pitts (McP) introduced runs deeper than the fact that threshold functions can be represented as predicate expressions. SDT neurons with only the somatic segment are for all intents and purposes equivalent to McP neurons, but this analogy misses the point. So far, we have argued that the inputs at a single segment should be regarded as inputs from a population that emits spike volleys. Therefore, formulating the threshold function at a single segment as a predicate expression misses the point of the SDT model: Long, interacting memory due to plateaus and a more biologically realistic, probabilistic model of coincidence detection that encodes stimulus confidence. Instead, we can represent any McP order 0 net by SDT neurons constructed by the following procedure:

1. For each neuron in the McP net, introduce an SDT neuron.

2. For each afferent neuron M_i with an excitatory connection to McP neuron M_j , introduce a child segment c^i with an excitatory connection to N_i at SDT neuron N_j .
3. For each afferent neuron M_k with an inhibitory connection to a McP neuron M_j , connect N_k to every child segment c_i of SDT neuron N_j .
4. Set the dendritic threshold of the soma at N_j to the threshold of M_j , set the dendritic weight ω_i to m_i for each segment and respective afferent neuron. The soma of N_j has a synaptic threshold of 1, and all child segments c^i have a synaptic threshold of 1.

The expression of each SDT neuron is then given by

$$\mathbf{S}(N_i)(t) \equiv \prod_k -c_{t-1}^k \sum_{\pi \in \kappa} \prod_{i \in \pi} c_{t-1}^i \quad (14)$$

because the common inhibitory terms can be written outside the sum over child segments and the recursion can be replaced with spike volley events directly. This is equivalent to a McP net of order 0 without cycles if we make the same assumptions about timing⁴. In our case, this would require a plateau reset on every dendrite segment via inhibition and a spike trigger signal on the soma all supplied by the same clock. This is of course also what happens in an McP net, but because their model isn't inherently asynchronous like the SDT model it doesn't have to be dealt with explicitly. Figure 17a-d show how this procedure can be used to transform McP nets to simple SDT nets. Can we find asynchronous solutions that implement the same computation? The example in figure 17c encodes the perception of heat felt when a cold object briefly touches the skin (node 3) that turns cold when the object is held to the skin for a longer time (node 4). Node 1 is meant to represent a heat receptor, and node 2 a receptor for cold. Hence, heat is either felt if node 1 is active or if node 2 is active, but is inactive a short while after. If node 2 is active for a longer time, cold is felt. The diagram in figure 17e shows how a simpler and asynchronous solution is available as an SDT model. Given that the receptors might be noisy, the encoding of heat and cold signals as spike volleys may also be sensible.

That we can implement McP nets with SDT neurons means that the *temporal propositional logic* introduced for events in discrete time is also implementable in SDT neurons (Theorem 2) and that propositional sentences in disjunctive normal form can be represented by such neurons, but only when there is no term in any of the individual

⁴ See their assumption about synaptic delay advancing the clock by 1 per operation.

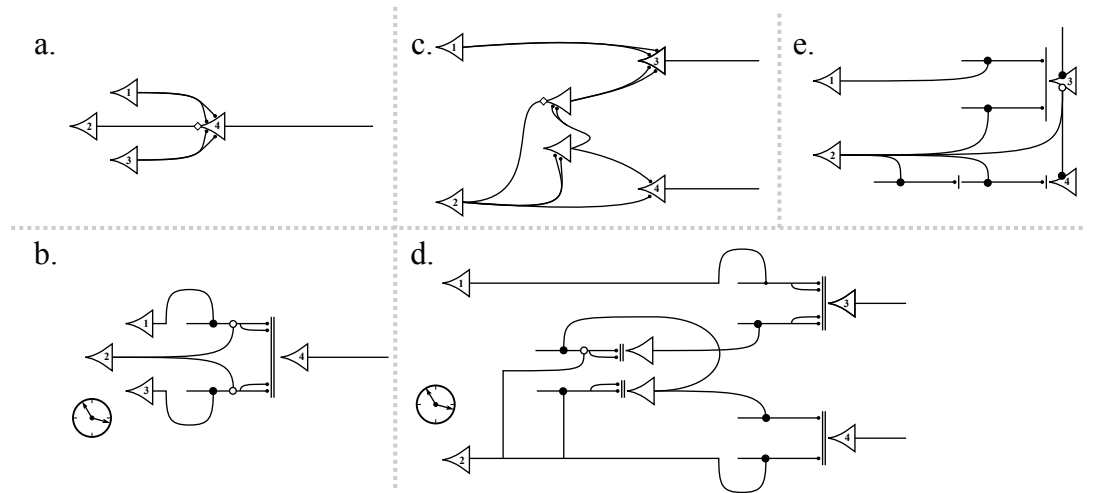


Figure 17: CONVERTING MCCULLOCH AND PITTS NETS. Examples from McCulloch and Pitts:

a. $N_4(t) \equiv N_1(t-1) \vee N_2(t-1) \wedge \neg N_3(t-1)$ as well as

c. $N_3(t) \equiv N_1(t-1) \vee N_2(t-3) \wedge \neg N_2(t-2)$

and $N_4(t) \equiv N_2(t-2) \wedge N_2(t-1)$.

b. implements a. directly with SDT neurons, d. implements c. directly with SDT neurons. Both solutions require the assumed clock in the McCulloch and Pitts nets to coordinate inputs and cancel plateaus early (see Text). e. implements the logic of c. asynchronously on rank-ordered inputs. The soma may be queried by external spikes here.

conjunctions that contains purely negated terms (Theorem 3). This reflects the fact that inhibitory synapses can prevent but not generate a signal.

The read/write head of a Turing machine, a finite-state machine, is implementable by McP nets without cycles, and therefore by SDT nets as well. This is no surprise since the internal operations of an SDT neuron may be modeled by simple finite state machine (see [chapter 1](#)).

In sum, the SDT neuron can also implement point neuron models, but with less strict requirements on the timing of signals carrying data for the computation. It also computes a probabilistic graded response to a graded stimulus in form of a spike-volley. This means that the computation is robust to additional noise in the signal and deteriorates gracefully.

However, thinking of dendritic plateau computation as a more robust re-implementation of threshold neurons doesn't paint the complete picture. We started out with the question whether dendritic plateaus and their interaction on the dendritic tree add a fundamentally new quality to neural computation. Let's therefore consider whether the SDT neuron could reasonably be reduced into smaller units. A good candidate is a somatic segment and a dendritic segment together. This preserves the long memory that was our initial motivation to invest-

igate dendritic plateaus if the soma persistently fires spikes, like for example delay cells do [84]. When the soma responds to a pattern of activation, a spike volley, this corresponds to the HTM neuron model [92].

Figure 18 shows two solutions we may come up with to implement the functionality of N4. In the first solution with 5 distinct smaller neurons, we need an inconsistent model that sometimes has deterministic synapses when connections come from neurons previously part of the dendritic tree or probabilistic synapses when external populations are connected. If we instead consider an ensemble for each of the previous segments, this inconsistency can be eliminated. But we now have the issue that the populations with segments d and c terminate on the same segment e, which means that spike volleys of the two populations can add up and low confidence in either signal can erroneously add up to high confidence that either signal true. We might then argue that we can prevent this by preventing the populations, d for example, from permanently firing, similar to c. But if we are then interested in populations for c and d⁵ active at the same time, we are back at the asynchrony problem we started at. If a and b aren't permanently firing during plateau potentials, the problem would occur at segment c, too. The signal isn't bound by a plateau potential overlap and must be somehow externally synced even though the populations may code for stimuli on different clocks. Neurons must now change their firing behavior depending the function their signal is to be used in later.

In parallel, we have now turned the previously deterministic connection from children to parents within the dendritic tree into probabilistic connections which may introduce a second source of noise and requires near perfect linear encoding and decoding of uncertainty at each state.

This circle that contradictions which seem solvable at first lead to solution with a different problem instead is no accident. The SDT model strictly separates computation relating to external and internal representations.

Externally, the SDT model follows a typically connectionist model: Representations are distributed among many different input neurons, each individual spike can be considered subsymbolic [226, 227] in the truest sense of the word, no meaning can be ascribed to it. With the initiation of a plateau due to a spike volley, the subsymbolic, distributed external representation turns symbolic. Internally, the SDT model follows a classical, symbolic structure: Representations are local⁶ and symbolic, and part of an expression with obvious constituent structure. The meaning of the whole is made up out of

⁵ Ignoring the inhibitory connection from C to d for the sake of the argument.

⁶ As many authors have pointed out, symbolic architectures are not limited to local representations, but local representations are typical of them. The same is true for distributed representations and connectionist architectures.

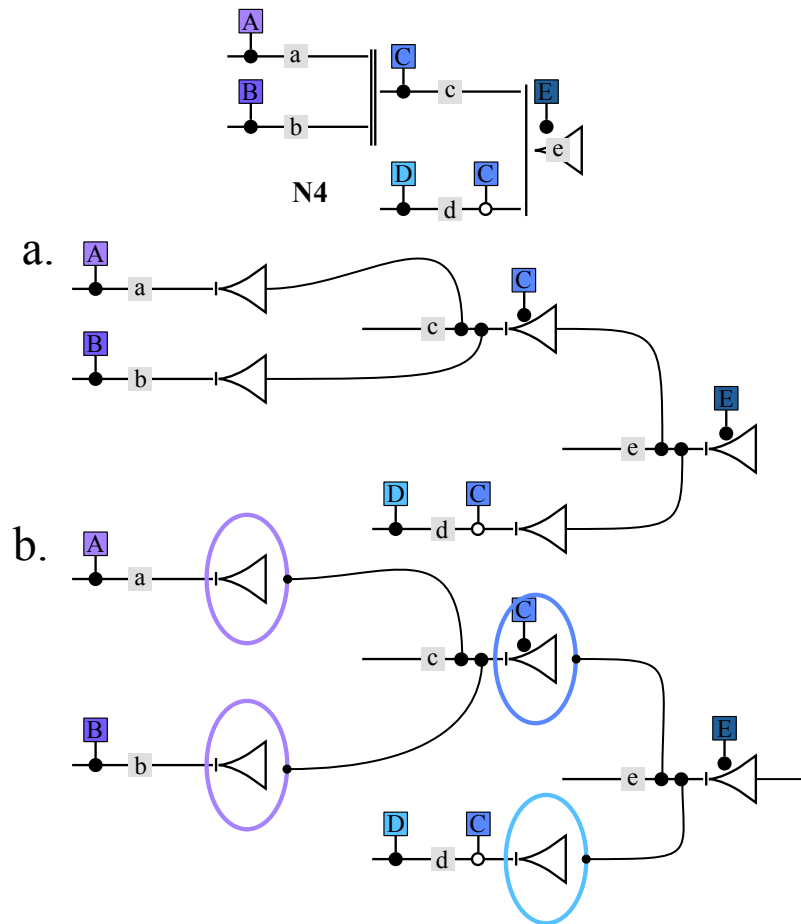


Figure 18: REDUCING SDT NEURONS TO SINGLE SEGMENT MODELS. Two options to reduce a single SDT neuron N_4 into networks of neurons with only a single segment. a. Replacing each segment with a single segment neuron that forms a deterministic connection to the parent segment or b. replace each segment with a single segment neuron population that forms probabilistic synapses. Circles indicate ensembles of neurons with the same structure. See the text for a discussion.

the meaning of its parts, and changing one part changes the meaning of the expression. The SDT model follows the idea of mapping symbolic expressions to structures of physical states discussed by Pylyshyn [196]. In his book “The algebraic mind”, Gary Marcus proposes *treelets* [159] as a possible foundation for a computational cognitive architecture that follow the same branching structure that SDT neurons follow.

In many of the discussions between proponents of the connectionist point of view and representatives of good old fashioned A.I. and a symbolic view of the mind, questions revolve around whether our symbolic language of thought can or cannot arise from an implementation in connectionist hardware [34, 66]. And if it can, should we think

of the mind as connectionist?

The brain might have been one step ahead. Dendritic plateau computation analyzed through the SDT model suggests that pyramidal neurons may be tiny, symbolic computers that are then connected through a connectionist network of synapses. The hardware is both connectionist and symbolic, different problems seem to require different solutions. Both the routing of information via synaptic connections and the extraordinary diversity of dendritic structure across different neurons have a role to play. This presents an exciting new perspective for neural computation and answers the question whether dendritic plateaus and their interaction contribute novel computational capabilities to single neurons: They do.

4.3 A NEUROMORPHIC HARDWARE IMPLEMENTATION OF THE SDT NEURON.

This exciting new perspective also extends to the field of neuromorphic hardware. It's central goal is to develop computer hardware that directly implements ideas derived from the computational principles of the brain for technological benefit or to help understand computation in the brain [106]. Historically, this has often meant to implement computation with spiking neurons in super low voltage ranges of transistor leading to particularly energy efficient circuits [163]. For example, modern analog neuromorphic computing systems such as BrainScaleS [210] can be used to simulate neural circuits that implement a deep neural network at 10e5x biological time.

The second defining characteristic of brain computation, information transmission via spikes, can lead to sparse communication patterns and reduce energy consumption that way. It can also make extremely parallel computation viable. The SpiNNaker project uses this to enable the simulation of neural networks for research over 2,500 processors in a super computer [74].

Since the successes of deep learning, the technological prospects of neuromorphic hardware have also come into focus. Both IBM and Intel have developed neuromorphic chips that leverage modern, digital semiconductor technology and merge it with spike-based communication and extremely parallel processing [42, 165].

Ultimately, the application of these technologies is to be the brain of autonomously behaving, cognitive artificial systems [35]. Here, the challenges for the computing system are the same we have discussed in [chapter 1](#): All inputs are presented in time, but all inputs follow their own, internal clock.

This is where a neuromorphic implementation of the SDT model can contribute. The focus isn't so much an analog implementation or spike-based communication. Instead, the goal is to develop technology for an asynchronous neural computer that can reliably compute

on asynchronous inputs.

In Leugering, Nieters and Pipa [137], we have presented an implementation of an SDT ensemble. The core computational component, the dendrite segment with coincidence detection and dendritic plateau computation, is built out of very simple units. The reactive data path of spikes and plateau potentials is not linked to any clock and drives the computation. Clocks are only used to discount each spike we have counted after one kernel length τ_E , and to disable the plateau after τ_P . The exact length is not precise because the plateau and spike onsets are not synced to either clock. This implementation maintains the long memory of the SDT neuron as well as the extremely fast reaction to inputs. Figure 19 gives an indication of the design that is presented in detail in section B.3 and also includes configuration, inhibition and stochastic synapses in binary branching SDT neurons embedded in an ensemble.

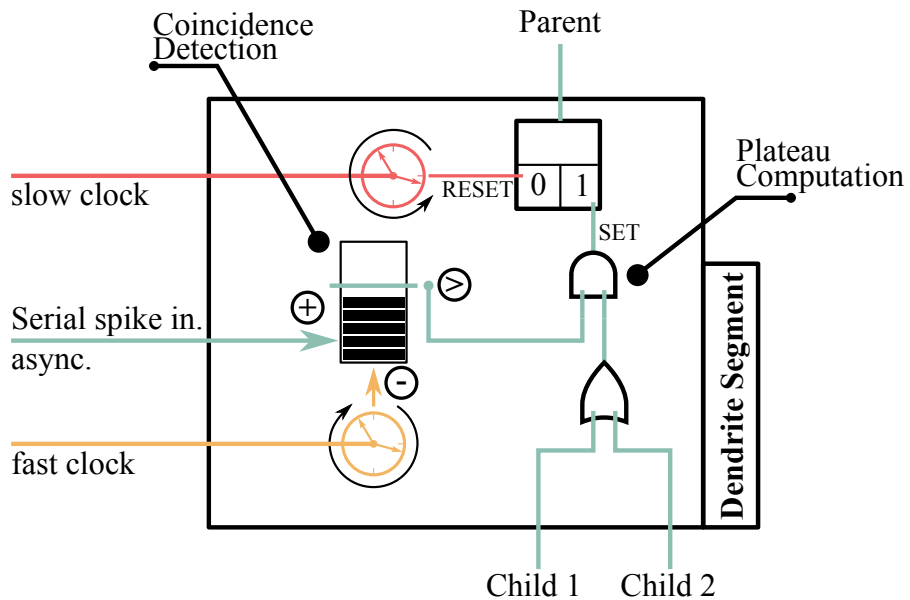


Figure 19: A SIMPLE SDT SEGMENT HARDWARE IMPLEMENTATION. Each segment is connected to two child segments and one parent segment. The signal from the serialized spike input line and the parallel child segments are not clocked and trigger gates and computational elements on the rising edge of their signal. A Plateau is set, when more than a threshold spikes are currently in the spike counter and at least one child segment is on. A fast external clock subtracts a spike from the spike counter roughly τ_E after it was added. A slow external clock resets the plateau signal roughly τ_P after it was set.

The implementation does not require any arithmetic logic units and is therefore quite simple. A considerable amount of gates and logic still has to be used to generate long enough timescales to reset the contribution of a spike to the post-synaptic potential or to disable

the clock. Future work should further reduce the reliability on external clocks that cannot be synced to fundamentally asynchronous inputs and increase the configurability of dendritic trees. The focus on timescales, particularly slow ones, also makes the model ideal for new computing technologies such as volatile memristors [263], the purposefully forgetting cousin to more stable memristors that have previously been used to implement synaptic weights [8].

In the broader picture of implementing a technical brain for autonomous agents SDT neurons are unlikely to be the only computational building block required. As we have discussed in the introduction, different problems require different types of memory. A good basis for a neuromorphic system with a diverse toolbox of solutions for different problems may be mixed-signal architectures that include analog computation alongside digital systems [173]. In the next chapter, we will discuss how problems for which data is best viewed as continuous time series can be handled by small dynamical computing systems that can complement SDT computation.

PREDICTING TIME-SERIES WITH DYNAMIC COMPUTING SYSTEMS

The time-series view of temporally ordered information differs from the discrete and structured sequences of events we have considered so far. A discrete time-series is often thought of as series of temporally equidistant samples of a system that is continually changing in time. The discreteness, which is fundamental in sequences of linguistic utterances, is not a feature of the signal itself but just a relic of the fact that we cannot easily store continuous information. This puts different constraints and requirements on the representation of the time-series in memory.

A loose motivation for what a representation of the past has to accomplish to enable the prediction and forecasting of dynamical systems comes from the mathematical study of dynamical systems. When studying chaotic systems such as turbulence in fluids, forecasting and understanding any particular system is notoriously difficult. Takens [243] and Packard et al. [185] showed that, surprisingly, the delay embedding of the time-series of just a single, derived variable is sufficient to accurately reconstruct the strange attractor that governs the system's chaotic behavior. The variable observed in the *delay embedding* must link to all variables in the system of interest, and we need to measure a system with d variables at at least $2d + 1$ different delayed points to be able to reconstruct the dynamics in the observed and potentially derived variable. The application of the approach to real world measurements of dynamical systems proves to be more difficult than the simplicity of the statement suggests. Particular formal demands are placed on the delay embedding in practice – for example in the estimation of causality in delay-coupled systems such as activity across different brain regions [213]. But the intuition that a compact representation of a system's history is a powerful basis to predict its future is powerful, especially in time-series prediction tasks [248].

In reservoir computing [111, 152], recurrent and randomly connected neural networks are driven by dynamically varying stimuli. At any one point in time, a readout, i.e. a memory-free mapping of the current reservoir states onto an output variable, can predict the dynamics of the system or compute other features of the system dynamics (Fig. 20a). This mapping can be as comparatively simple as a generalized linear model [175] or as complex as a feedforward neural network and is trained to map the dynamics of the input driven random recurrent system onto the target output. Often, a simple readout

model suffices as the neural network dynamics are already non-linear. In order to do so successfully, the dynamics of the reservoir should represent a non-linear expansion of its current and past inputs. A sufficient condition for such dynamics is the “echo-state” property which means that networks cannot be chaotic and have to be independent of inputs at some point in the past. Reservoir computing near the “edge-of-chaos” [133], i.e. systems that have a long dependence of the past, has been shown to work well in many applications. The randomness of the connections in a reservoir computer can be interpreted as being non-specific. On the one hand, this means that the weights that recurrently connect neurons in the network need not be specific to a particular task. This generic approach to circuit connections has led to reservoir computing as an explanatory framework for the intricate dynamics in cortical networks [26, 224]. On the other hand, it means that the dynamics of the reservoir must fulfill the requirements for reservoir computing but which input driven dynamical system in particular is used is not important. This second property has made the reservoir computing framework appealing for researchers investigating new computing substrates and materials both in the brain-inspired research on neuromorphic hardware and other unconventional computing approaches [40]. A reservoir can be as simple as a bucket of water [62], an autonomous Boolean network reminiscent of Turing’s type-B machines [5], build out of high-tech materials such as carbon nanotubes [39], or networks build out of electrochemical transistors meant to act as embedded biosignal monitors [37]. Reservoir computing has opened the door to many novel and decisively different computing devices.

Of particular interest here is the single-node delay-coupled reservoir computer (DCR). In a DCR, a non-linear node is driven by an input signal and its own delayed feedback [6]. The input time-series $u(t)$ changes on the time-scale of this delayed feedback while the DCR undergoes rapid changes during each delay cycle. In a traditional reservoir computer we represent a non-linear expansion of the history of an input time-series in the parallel vector of activation of all nodes at one point in time. The DCR does this mapping not in parallel but on the much faster time-scale of its internal, complex dynamics. We can therefore rapidly sample the activity of the DCR multiple times during one delayed feedback cycle and get a representation of the stimulus’ past analogous to the reservoir but constructed sequentially in-time (Fig. 20b).

This method is particularly well suited to neuromorphic hardware approaches than can leverage very high frequency updates and can be as simple as a single Boolean node with delayed feedback [93] and has frequently been used in optical or opto-electronic computing devices with delay lines [25, 97, 184]. Networks of delay-coupled

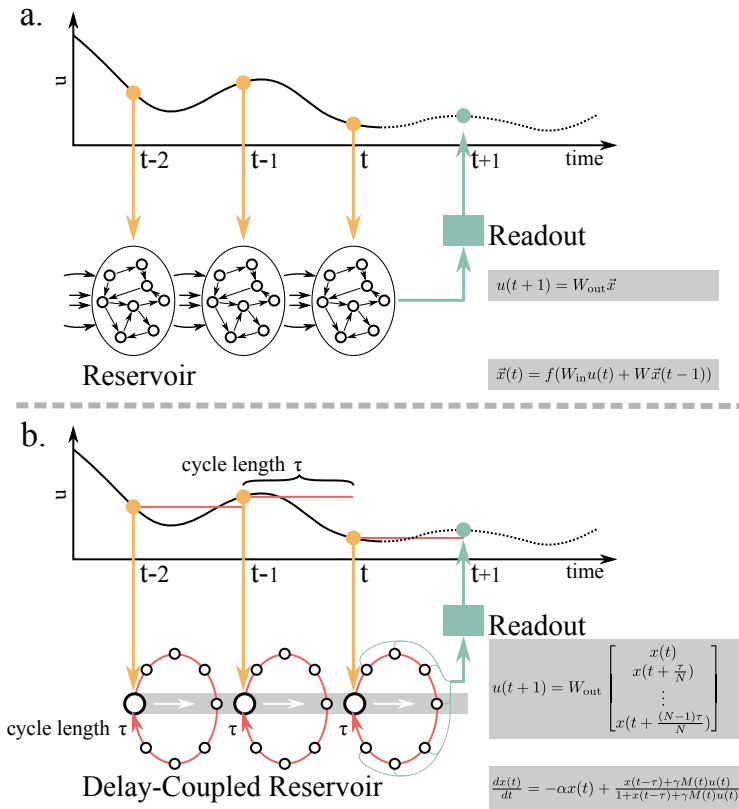


Figure 20: RESERVOIR COMPUTERS REPRESENT PAST INPUTS IN THEIR ACTIVATION VECTOR.

a. Traditionally, reservoir computers are simple recurrent networks with unspecific, random internal connections W that represent past inputs from a time-series in the activation of their nodes $\vec{x}(t)$ at one point in time. This information can be used by linear models in the simplest case to predict the time-series.

b. In a delay-coupled reservoir computer, a single node coupled to a delayed feedback of its own activity can compute a vector that represents the past not by recurrently connected parallel nodes but by undergoing rapid, dynamic change according to a delay differential equation. Each feedback cycle of length τ corresponds to one new input. Sampling the DCR activity N times during this cycle gives a vector that can be mapped onto a prediction target analogous to the recurrently connected reservoir. The specific form of the DCR used here is based on Mackey-Glass dynamics [153].

systems show, that the delay-coupling approach can be embedded in larger computational systems [187] and the dynamics of a single node with multiple delayed feedback signals can be used to emulate much more complicated deep neural networks [233].

But the concept may be more general: Small, delay-coupled systems can represent a long history of information on the timescale of their delays in their inherent dynamics. If another systems can read and integrate the dynamics on a timescale potentially much faster than that of the delay and input, it can use the delay-couple system as a

reservoir and predict the target signal.

Whether the brain can make use of delay-coupled mechanics to retain information in small circuits or single neurons remains speculative for now. However, neurons are known to form autapses [255], electrical or chemical self-synapses, that have been proposed as a particular implementation of working memory [216]. They are often modeled as delayed feedback from soma to dendrite [197, 261] and have functionally been linked to coherent activity in interneurons. This property is shared with electrical synapses, or gap junctions, generally. These types of synapses can be traced to form large networks between inhibitory interneurons [72, 73]. Electrical synapses and autapses may form single-node systems or small motifs in which delayed interactions can lead to memory much longer than the delay.

Unlike a recurrently connected neural network, a single-node system does not have degrees of freedom in the weight matrix between nodes (Fig. 20a) that can be changed to adapt the dynamics and is instead governed by a delay differential equation (DDE). Figure 20 shows the delay differential equation for a Mackey-Glass system [153] that includes an exponential decay and a non-linear dependency on the activity at delay τ and can be used as a DCR. Input to the system is constant for one τ -cycle, randomly perturbed by a mask M that is periodic in τ and mixed with the delayed feedback in the non-linear term of the the DDE. In a physical system that implements the dynamics, we can sample the activity N times during each τ cycle and compute the prediction of our input based on this sampling vector. Analytically, DDEs are difficult to handle. Solutions can typically only be defined for one τ cycle i , given that the solution on the previous τ cycle $i - 1$ is already known. Because each solution is a function, solving a DDE requires a map from function to function that could in theory be infinitely dimensional for each τ .

Outside of exchanging the DDE used in the DCR, these intricate dynamics are difficult to change. In keeping with relatively simple physical realization of the computing system that makes the DCR so interesting, one approach is to simply add additional delayed feedback lines. However, given that the first delay already made the dynamics of the DCR difficult to analyse, can we understand the dynamics that enable computation in a two delay DCR? Nieters, Leugering and Pipa [177] show how this simple addition of a second delay can indeed increase the performance of a DCR on a non-linear, history dependent time-series prediction task significantly. But it can also lead to decrease in performance by the same factor if the second delay is chosen incorrectly. The reason is that the second delay introduces a new layer of computation that can interfere with or add to the ability of the DCR to represent a non-linear expansion of the past.

To analyze the system, it is useful to think of the sampled activity along one τ cycle in the DCR as a network of virtual nodes and derive the equation that updates each virtual node directly from an analytic solution of the Mackey-Glass DCR [212]. If we sample N virtual nodes $\mathbf{v} = [v_1, \dots, v_N]^T$ equidistantly, they are each spaced $\theta = \tau/N$ apart. The update equation for the DCR becomes:

$$\mathbf{v}^i = \mathbf{b}(\frac{\theta}{2}f_0^i + \theta_0^i) + \frac{\theta}{2}C\mathbf{f}^i \quad (15)$$

$$\begin{aligned} \text{where} \quad \mathbf{b} &= \begin{bmatrix} e^{-\alpha\theta} \\ e^{-2\alpha\theta} \\ \vdots \\ e^{-N\alpha\theta} \end{bmatrix} \\ \text{and} \quad C &= \begin{pmatrix} 1 & 0 & \dots & 0 \\ 2e^{-\alpha\theta} & 1 & & 0 \\ \vdots & & \ddots & 0 \\ 2e^{-(N-1)\alpha\theta} & \dots & 2e^{-\alpha\theta} & 1 \end{pmatrix} \\ \text{and} \quad \mathbf{f}^i &= \begin{bmatrix} f_1^i \\ f_2^i \\ \vdots \\ f_N^i \end{bmatrix} \end{aligned}$$

\mathbf{b} captures the dependence of each solution v^i for the i -th τ -cycle on the initial value at t_0^i and C is the implicit connectivity of the network of virtual nodes. The vector \mathbf{f} is the non-linear activation of the nodes activity one τ cycle ago and the current input, similar to the activation of a node in a neural network: $f_j^i = f(x(t_0^i + j\theta - \tau) + M(j\theta)u(t_0^i))$ and $f(x) = \frac{x}{1+x}$. If we add a second delay, this simply becomes $f_j^i = f(x(t_0^i + j\theta - \tau_1) + x(t_0^i + j\theta - \tau_2) + M(j\theta)u(t_0^i))$. In this formulation, we can distinguish the implicit dependency structure in the DCR due to the inertia of the system and represented by C and the explicit dependency introduced by additional delays in \mathbf{f} .

Figure 21a shows different explicit dependency structures depending on how τ_2 is chosen in relation to τ_1 . Integer multiples for example reinforce the dependency of a node v_k^i onto itself in previous cycles v_k^{i-1} . Generally, for $\tau_2 = \tau_1 + m\theta$, additional dependencies on different virtual nodes v_l^j and $l \neq k$ in different cycles $j < i$ are introduced. Independent of the specific cycle, each new node in the dependency set D_k of the original node v_k is recursively dependent on virtual nodes by the same rules. The size of this dependency set is given by

$|D_k| = \frac{N}{d}$ where $d = \text{GCD}(\tau_1, \tau_2)$ is the greatest common divisor between the two delays (see section B.4 for more details).

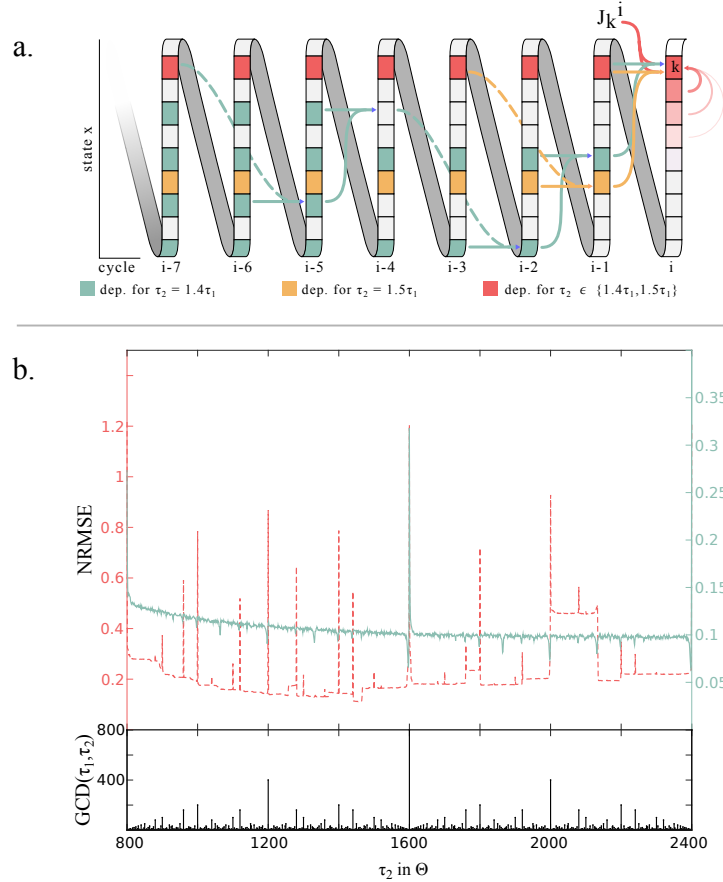


Figure 21: COMPUTATION IN DELAY-COUPLED SYSTEMS WITH MULTIPLE DELAYS a. Diagram of dependencies of virtual nodes in a delay-coupled reservoir with more than one delay. In each τ -cycle $i, i-1, \dots$, nodes are coupled via the exponential decay of the system. Across cycles, virtual nodes are coupled via the delay term. While the first delay determines the cycle, the second delay can add new dependencies.

b. Which nodes are newly coupled via the second delay has significant effects. In a time-series prediction task the correct choice of the second delay can massively increase or deteriorate performance (green curve). Disabling the exponential decay in the system reveals that this structure depends on the GCD of the two delays as measured in virtual nodes in one τ -cycle (red curve and bottom panel). Measured is the normalized root mean squared error (NRMSE) on a non-linear autoregressive moving average time-series (NARMA) with a time horizon of 10 time steps.

By increasing the speed of the decay α – i.e. $\alpha \rightarrow \infty$ – in equation 15, we can eliminate the effect of the implicit dependencies and investigate the effect of explicit dependencies only $\mathbf{v}^i = \frac{\theta}{2} \mathbf{f}^i$. We found that performance was consistently bad if the GCD between two delays was large – such as $\tau_2 = 2\tau_1$ and $\tau_2 = \frac{3}{2}\tau_1$ – and consistently very

good if the GCD was small – i.e .1 when the two delays were co-prime (Fig. 21b). Reintroducing the implicit coupling via C blurs this effect but a large dependency set remains the correct strategy for a long history dependence and optimal performance.

In sum, introducing delayed dependencies into simple systems is a simple yet effective strategy to maintain a flat representation of the past that can be used to compute predictions. Adding additional delays can vastly expand the memory of the system without requiring significantly longer delays. Instead, the longer memory is achieved with an additional layer of dependencies resulting from the interaction of the two delays over time. This suggests that discrete delays may be a computational building block in neural circuits that rely on a flexible working memory implementation [135].

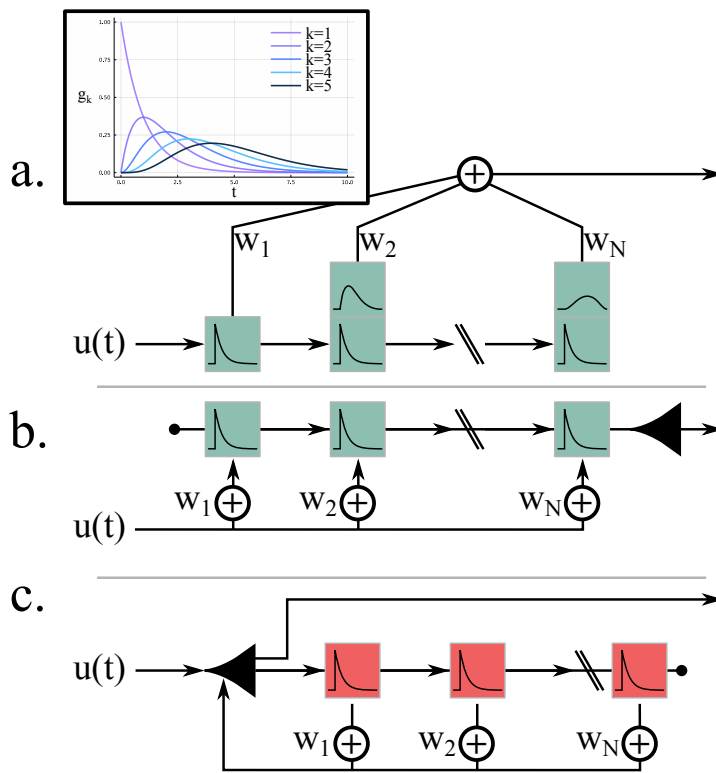


Figure 22: FILTER BANKS CAN REPRESENT A SIGNAL ON MULTIPLE TIMESCALES. Concatenated exponential filters lead to gamma response kernels of different degree. a. The individual responses can be combined to represent an adaptive, parameterized filter over a signal by a weighted summation of the individual taps. b. The linearity of the convolution allows for the multiplication to precede the filter operation. c. The filter can also be used on the output of a node as a feedback signal.

Another strategy to represent the past of a particular stimulus is to filter it on different timescales. A particular good candidate for this is a filter bank of gamma filters [195]. In it, each filter individually

has an exponential filter response, but their concatenation results in a gamma filter kernel $g_k(t) = \frac{\mu^k}{(k-1)!} t^{k-1} e^{-\mu t}$ for the k -th filter in the filter bank (Fig. 22a). For $k = 1$ we get an exponential filter response, for $k = 2$ we get the double-exponential response. Both are familiar as synaptic response kernels.

The responses of the different filter taps in the filter bank create a continuous representation of the signals past, with the highest degree kernel having the longest memory horizon, but also the widest low-pass filter response. Similar to the DCR, a readout model can weight the responses of each filter tap to create an adaptive, parameterized filter response. But the linearity of the convolution operation allows us to move this weighting operation around, and we can instead weight each input directly (Fig. 22b). This approach can also be used to create a feedback filter bank for single nodes or neurons (Fig. 22c) which one can think of as a continuous version of the previously discussed delay-coupling with multiple delays.

De Vries and Principe [44] formulated a model of neural computation based on gamma filter banks and delays [45] to implement recursive memory structures and extended it to maintain in memory the activation of an entire neural network [171].

In a proposal (section B.5), we have outlined how the different strategies to represent the past of the input and the activation can be combined in a single neuron model to detect keywords in an audio stream (Fig. 23a). The model combines filtering of input signals with a feedback filter bank and spike generation and uses a local, gradient-free learning rule adapted from the Tempotron learning mode [88] together with a variance based intrinsic plasticity [138]. As a proof of concept, all variables in the model are dynamically evolved in a differential equations solver. Learning is triggered based on an external trigger when the target was detectable and relies on internal traces for each weight to compute a weight update (Fig. 23b).

To test the system, an audio signal from the speech commands database [264] was decomposed into 10 signals according to the mel-frequency log scale [83], each of which was passed to a gamma filter bank with 10 taps, one filter bank was reserved as a feedback filter. The system was able to learn a smooth membrane response that lead to above chance performance on detecting "UP" (Fig. 23c).

The linearity of filtering operations allows for much flexibility in designing new learning rules and employing a vast number of different filters in contrast to the simple filtering operation and a small number of delays in a DCR. The same property also allows multiple inputs to be filtered by the same filter bank as long as learning signals can be kept separate. This would allow for straight-forward, adaptable analog hardware implementations of this adaptive filtering framework [195]. Conversely, the filter responses can be much

broader and less specific in time and can require relatively long filtering timescales depending on the problem to be solved. Nevertheless, models inspired by the original work by De Vries and Principe [44] can lead to intricate single neurons that absorb filtering into the computational process. This is particularly well suited for complex, fast and noisy temporal signals such as audio waveforms.

For modelling of neurons, the width of a synaptic response of a distal synapse as recorded at the soma suggests [238] that a variety of synaptic response kernels as degrees of a gamma response may be useful to increase the spatial resolution and memory of single neurons and include distance to the soma as a factor in the model.

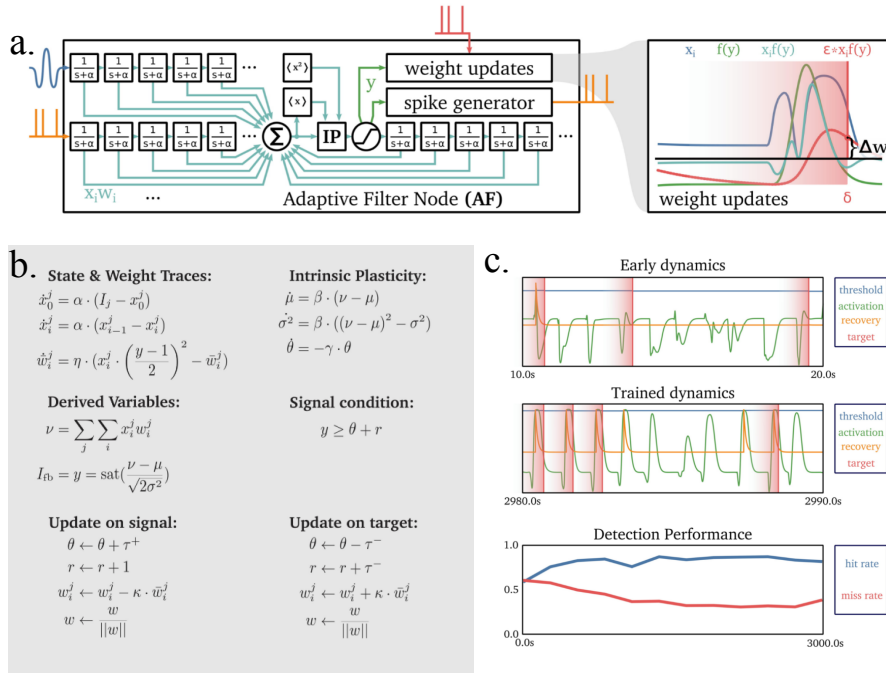


Figure 23: PROOF OF CONCEPT: ADAPTIVE FILTER NEURON a. The adaptive filter neuron combines input and feedback filter gamma filter banks with an adapted version of the Tempotron learning rule by dynamically tracking the activity at each filter tap and adding an adaptive spiking mechanism ("Signal"). b. The system can be entirely described as a coupled differential equation with missed or triggered spikes adapting weights. c. The procedure can learn to recognize the command "UP" from a mel-frequency decomposition of a noisy audio file.

In theory, the continuous and unstructured representations of the past of a stimulus or a system we have presented here can be sufficient to provide the memory for a version of a universal Turing machine [151]. In practice, using a dynamic computer that can keep information about the past in its state when the input and target have different inherent has proven difficult. Because the timescale of the input, the reservoir and the output are tightly linked, adaptations to the

model are required to accommodate some independence [158, 244]. Nonetheless, the approach provides an elegant solution to store information in a flat, unstructured way that makes the past of a stimulus or signal expanded in the non-linear dynamics of the system available at a single point in time. Combined with a read-out model that can be surprisingly simple this embedding is ideally suited to predict and forecast the evolution of time-series and many dynamical systems. It is in these important problems that the concept shines and suggests that the circuitry required to solve these problems may be much simpler than previously imagined.

LEARNING GENERALIZED DYNAMICS FROM TRAJECTORIES.

In the previous chapter we have seen how trajectories can be forecast into the future based on a limited set of previously sampled points along it. This intuition enabled a number of different dynamically evolving computing systems as simple as single-nodes to solve non-linear prediction tasks. But this approach is not sufficient to capture the general dynamics of more complex systems that change their behavior. Sampled trajectories then only reflect the version of the system currently observed.

Finding general descriptions in the form of governing *differential equations* of dynamical systems is the domain of many scientific disciplines. It not only enables the forecasting of trajectories from any given state without requiring any further memory of additional previous states, it also permits the study of *tipping points*: Sudden, fundamental changes that lead to entirely different dynamical behavior of a system.

A system's dynamics are described by trajectories in state space, a space defined by all variables of the system. It contains stable and unstable manifold, separatrices and different types of attractors that repel, attract, and separate trajectories and form the organizing structure that determines the dynamic behavior of the system [237]. A change in this organizing structure due to the shift of an external parameter is called a bifurcation of the system.

Unfortunately, stories about tipping points are now familiar to many from the news: The impact of oceanic flows on both the global and local climates [223], the impact of CO₂ on the global warming of the climate [232], the population levels of different species fighting for survival, and many other examples [63].

Differential equations have a second advantage over purely predictive models. Because they define the governing equations of a system completely, trajectories from the system can be generated from any initial value without the need to sample previous activations or any memory at all. The most overt example of generative systems in behaving animals is movement.

Here, dynamical system analysis has been used to analyze the movement of professional athletes [41], and dynamical systems are used as pattern generators and control policies in robotics [104, 209]. In these cases, bifurcations in the dynamical system can be used to smoothly switch between different movement patterns in an action [120].

In both ecological models and models describing the movement pat-

terns of animals and humans, careful observation, analysis, and hypothesis generation and testing is usually required to find the appropriate set of differential equations. Often, for example in the case of phytoplankton [219], our knowledge of the system is incomplete. In case we can gather data from the system in question, new data-driven methods and technologies are now being developed in machine learning that can help discover missing pieces in these models, or to learn black box dynamical models from scratch.

In the framework of *Universal Differential Equations* (UDE) [198], we reformulate the general equation for a dynamical system slightly. Instead of the general $\dot{\mathbf{x}} = f(\mathbf{x}, t)$ we also include a universal function approximator UA_p that can be learned during data-driven training of the model:

$$\dot{\mathbf{x}} = f(\mathbf{x}, t, UA_p(\mathbf{x}, t))$$

Because neural networks are shown to be universal function approximators [38, 103], they are a good candidate for the function UA . In this case, the parameters p represents the weights of the neural networks. We can include algebraic terms in the function f if we already know something about the dynamics or train UA to approximate the entire time differential.

The critical technological advancement is the training procedure to find p . It is directly connected to the recent explosion in deep learning technologies. As Yann LeCun, one of the central figures in deep learning research, exclaimed: “Deep Learning est morst. Vive Differentiable Programming!”¹. Differentiable programming means the construction of a computer program in which parameters can be adapted by gradient-descent. This is done by automatic differentiation tools that can apply the chain rule to arbitrary code by tracking parameters. It is the basis of deep learning frameworks such as tensorflow [1], but the technique can also be applied to programs that aren’t a neural network. In the case of the UDE, this program includes the numerical solver for the differential equation.

The procedure then is very similar to training a neural network. We first guess a set of initial parameters and solve the UDE on an interval and initial condition for which we have previously collected training data from a dynamical system. This will produce a trajectory in state space that is likely considerably different from the observed trajectory. Now, calculate a loss function based on the mean-squared error (MSE) $L_{MSE} = \sum_i^n \sum_j^m (x_i^j - p_i^j)^2$ for an m -dimensional system and n sample points at which the trajectories are compared. \mathbf{x} is the target trajectory and \mathbf{p} is the prediction by the UDE. Automatic differentiation can then take the gradient and adapt the parameter p of

¹ Facebook users may find the post here <https://www.facebook.com/yann.lecun/posts/10155003011462143>. For everyone else, here is repost: <https://gist.github.com/halhenke/872708ccea42ee8cafd950c6c2069814>

the neural network UA. Using a set of tools in the Julia programming language, this process is straight-forward for the user [13, 107, 108].

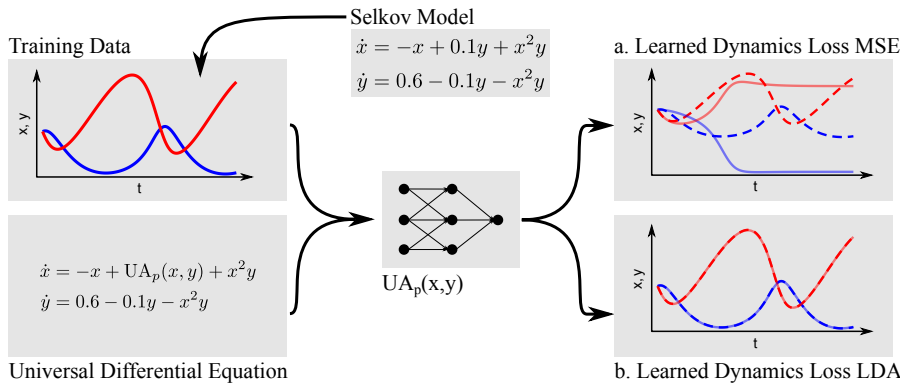


Figure 24: LEARNING MISSING TERMS IN THE SELKOV MODEL. A UDE of the Selkov model that replace a term of the original model with a neural network is trained to learn to generate the original dynamics. a. When the MSE loss is used, this reconstruction fails fundamentally. Instead of the oscillatory behavior of the system, the UDE has learned a steady state solution. b. When the LDA loss (see below) is used during training, the dynamics are reconstructed perfectly.

In Vortmeyer-Kley, Nieters and Pipa [260], we investigated whether this approach can be applied successfully to systems that can bifurcate. The simple two dimensional Selkov model [215] used to model the metabolic pathway of Glycolysis shows that a straightforward solution using the approach *as is* fails (Fig. 24a). The free parameters in the neural network UA are chosen randomly before training and can lead to a dynamical regime in the UDE that is different from our target trajectory. Instead of an oscillatory solution, we learn a steady state solution and gradient descent is not able to recover from a local minimum on the wrong side of the Hopf-bifurcation of the system. The effectiveness of gradient descent optimization in deep learning models is still not entirely understood, but for particular networks theoretical analysis has shown that even a single neuron can render local minima global [140] and the gradient descent finds the global minima [54].

In the differentiable programming setup of UDEs this does not hold. Figure 25b (blue curve) shows how measuring the loss of the UDE in a simplified model with $\text{UA}(x, y) = w_y y$ leads to a maximum in the loss landscape that separates the true global minimum from the locally minimal landscape for $w_y < 0$. We developed the simple intuition that we should instead measure the difference in length of

vectors along the trajectory and the difference in angle between these vectors separately and combine both in a new loss function (Fig. 25c).

$$L_{\text{LDA}} = \sum_{i=1}^n \left(k_1 \cdot \underbrace{\frac{\sqrt{(|\vec{x}_i| - |\vec{p}_i|)^2}}{|\vec{x}_i| + |\vec{p}_i|}}_{\text{length difference}} + k_2 \cdot \underbrace{\frac{1 - \frac{\vec{x}_i \cdot \vec{p}_i}{|\vec{x}_i| |\vec{p}_i|}}{2}}_{\text{angle difference}} \right) \quad (16)$$

Both components individually have local minima as well but at different values for w_y which means we can smooth out the separating global maximum in the parameter space. We hypothesized that in higher dimensional parameter spaces this could suffice to learn the dynamics of bifurcating systems if the correct weighting k_1 and k_2 is chosen.

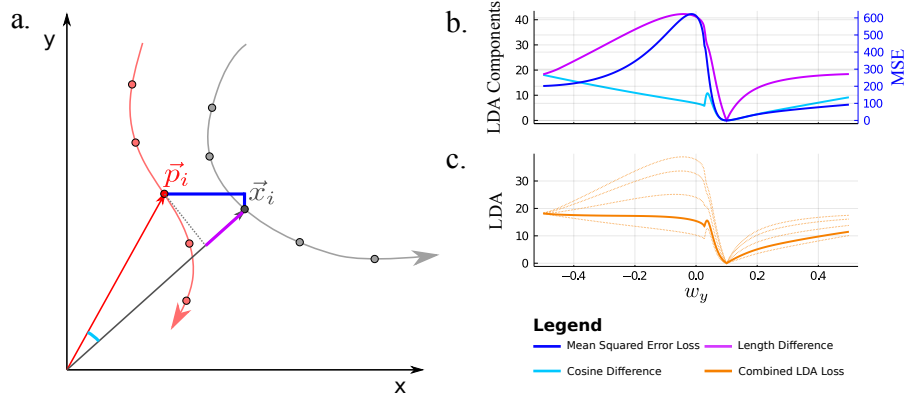


Figure 25: MSE AND LDA BASED LOSS FUNCTION a. The UDE estimated trajectory (red) and the target trajectory (gray) are compared at different sample points. The MSE (dark blue) compares the distance of each variable at the sample point independently. The LDA based loss compares the angle of the two vectors (cyan) and the difference in length (purple) independently, but both are dependent on both components of the state space vector. b. Shows how the errors compare in the Selkov model with the simple $UA(x, y) = w_y y$. All errors separate large parts of the parameter space from the global minimum at $w_y = 0.1$ due to a global maximum. c. A weighted sum of the LDA loss components allows a smoother combined loss, the LDA loss. Different weights are shown, $k_1 = 0.25$ (length) and $k_2 = 0.75$ (angle) is marked as the thick orange line.

In the Selkov model this was indeed the case and we were able find the correct solution (Fig. 24b). However, we are still somewhat reliant on the initial parameterization of the neural network. The effect of the LDA loss is that it increase the basin of attraction of the correct, global minimum and a significant portion of initial parameters learn the correct behavior. When we trained the system in different configurations close to the bifurcation of the system from steady state to

oscillation and plot the distribution of errors after training we can see this effect clearly (Fig. 26). To compare the solutions of both loss functions on the same scale, we calculated the trajectory difference between the final solution of the UDE and the training trajectory.

$$\text{TD} = \frac{1}{n} \sum_i^n \|x_i - p_i\|$$

If the system is close to bifurcation, the TD distribution over multiple training runs with different initial parameters becomes bimodal and the second mode corresponds to initial conditions that ended up in the wrong dynamic regime. Using the LDA loss, this happens much less frequently as the median of the distribution shows. Additional details on the experiment as well as additional simulations for different parameterizations of the dynamical system and a number of other bifurcating dynamical systems can be found in [section B.7](#).

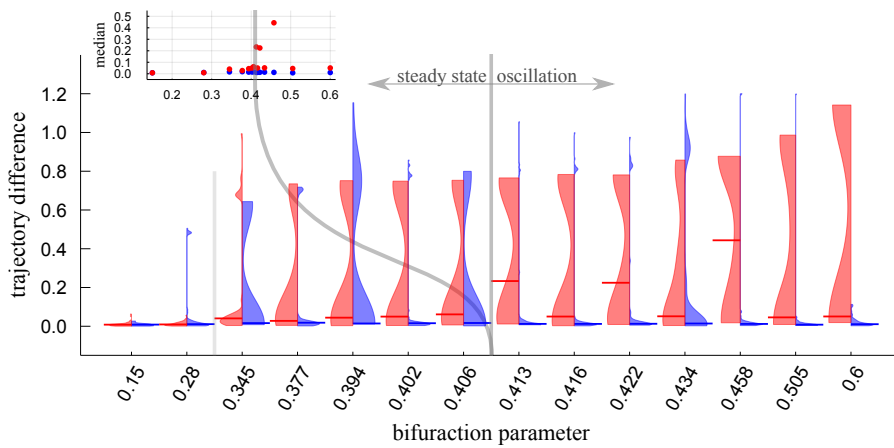


Figure 26: TD DISTRIBUTION SELKOV MODEL The UDE for the Selkov model is trained with a range of different bifurcation parameters (the first term in the equation, 0.6 in the previous example) used for the training data. We measured the trajectory difference (TD) between the final UDE trajectory and the target to compare solutions on the same scale. For each parameter, 50 different initial parameters were randomly drawn and the networks trained. Red violin plots show the resulting TD distribution for training with the MSE, blue violin plots the results for the LDA. The bars and the inset show the median of the distribution.

A particularly attractive property of the UDE approach is that it gives an extremely good estimate of the derivative of a dynamical system which can be used in system identification schemes like the recently propose “Sparse Identification of Nonlinear Dynamics” (SINDy) [24] to replace the black box neural network model with a white box algebraic equation, even in bifurcating systems[260]. This highlights the potential of data-driven approaches in all areas of science that are data-rich but in which it is difficult to formulate complete dynamical models. However, in order to confidently deal with bifurcating sys-

tems more work is required on understanding and eliminating the significant problem with local minima and underexploration of the parameter space that can occur during training.

The function approximation model used in the presented experiments is a simple, feedforward multilayer perceptron. The complexity of the approach did not originate in the neural network architecture but was instead related to how the dynamic trajectory generated by the system could be compared to the trajectory. The function approximation capabilities of deep learning networks may indeed also be approximated with trained spiking neural circuits in the brain [245, 272]. This suggests that dynamic control of, for example, movement may also be learned by approximating the governing, pattern generating dynamical system. Similar to the suggestions made in Richards and Lillicrap [202], our results on the learning of dynamical equation advocate for a focus on the objective function used in optimization instead of the complexity of the neural circuit. It is the problem statement that matters most.

Importantly, the neural network used here is memory-free and can still reliably organize dynamics in time by computing only the change to the current state vector. This means that patterns can be generated from a cold start which in turn is also required to freely switch between generated trajectories.

DISCUSSION

How can neural circuits can represent information about the past such that it is available for a computation at the right time? In our investigation, we learned that clever and quite different solutions to this problem can be found in surprisingly simple systems. Delay-coupled reservoirs obtain their ability to embed a long history of a signal by simple, delayed self-connections. Concatenated filters along a delay line can achieve a similar effect if inputs are connected at the correct position. The encoded signal history can then be used to predict it.

In Universal Differential Equations, we can replace part of a differential equation – or replace it in its entirety – with simple multi-layer perceptrons and learn from data. This would equate to a generalized model of a particular dynamical system. We do not even need memory for prediction and generation of patterns from the dynamical system if this generalization is available. Admittedly, we do need a representation of the current state that we can update.

Lastly, we learned that dendritic plateaus are a fantastic candidate to decode the rank-order of sequences of events that may operate on different clocks than the computing neuron does. We formulated the segmented dendritic tree neuron model to investigate whether the principle of overlapping plateaus across functional subunits in the neural dendrite should be regarded as a fundamental component of neural computation from a theoretical perspective. We learned that SDT neurons separate the external network of distributed representations over sub-symbolic spikes from an internal, locally symbolic representation. Computation in SDT neurons means to probabilistically evaluate an expression structured by the dendritic tree, and the response probability is proportional to the confidence in each constituent component.

Because this offers an exciting and new perspective, we will first discuss the most pressing open questions and what is next for the SDT model and then give a general outlook and conclusion to this thesis.

EXPERIMENTAL VERIFICATION OF THE SDT MODEL. We haven't yet discussed how the role of dendritic plateaus in computation as outlined by the SDT model can be validated experimentally. Because an antagonist for NMDAR channels, ketamine [81], has been known for some time, there are numerous behavioral studies in which NMDAR channel activity has been inhibited. Generally, ketamine can have a broad range of effects and result in cognitive deficits including schizophrenic and dissociative states [128]. Driesen et al. [51]

report a more specific effect of impaired performance in a spatial working memory task and reduced activation during “encoding and early maintenance” as measured by fMRI recordings. Wang et al. [262] show that specific NMDAR blocking reduces delay cell [84] firing in dorso-lateral prefrontal cortex (dlPFC) resulting in impaired working memory. In accordance with these studies, theoretical models have included the longer NMDAR induced post-synaptic potential as a source of working memory instead of synaptic change [55, 142]. The SDT model can reproduce the behavior of delay cells of the synaptic threshold at the soma is set to 0, and one key property of the model is the longer memory induced by dendritic plateaus. The results from the behavioral studies support the key function of NMDAR channel mediated active dendrites in computation and thus back a fundamental assumption of the SDT model as well.

However, this doesn’t yet implicate that the symbolic computation on rank-ordered sequences achievable based on the structured memory in SDT neurons is also used in the brain. Here, the theory relied strongly on the possibility that plateau potentials can interact in a branching dendritic tree with functional subunits. We argued that there is sufficient evidence that pyramidal neuron *can* implement this computation. To verify that they *do* would instead require that plateau initiation in dendritic trees is causally linked to previous plateaus in the dendritic tree. Kerlin et al. [119] did record isolated calcium events compartmentalized by dendritic branching. Dendritic spikes in the distal preforant path of CA1 pyramidal neurons are gated from reaching the soma if Schaffer-collateral synapses receive no input in experiments [112]. Together, this presents at least cursory evidence that this mechanism is a serious possibility. In section A.3 we suggest a method to measure the relative timing of calcium events and compare them to our model. The analysis could be used once even larger sections of the neural dendrite can be imaged at high temporal resolutions.

Conversely, our model also implicitly relied on spatiotemporal spike patterns in which synchronous volleys of spikes are central. Numerous studies have confirmed that these do in fact exist and are often associated with spine clusters on dendrites [241], but how often they occur depends on the data analysis used and has consequently sparked fierce debates [190, 218]. Whether rank-ordered spike volleys form a code is therefore not entirely clear. New methods to analyze recorded spatiotemporal patterns of brain activity are now being developed and may be used to independently verify that our assumed code of sequences of spike volleys is in fact expressed by neural circuits [229, 249].

The rapid development of experimental tools in recent years lead to the convincing evidence for sequential codes in the brain on timescales that match dendritic plateaus (chapter 2). At the same time,

new experiments are painting a more detailed view of the function of single neurons. To meet these advances and ever more detailed measurements, models with different levels of abstractions must be developed in which the plateau potential still takes center stage, but different, measurable biophysical variables such as true membrane voltages should be included to enable quantitative predictions. Further, procedures to fit the behavior of measured neurons with active dendrites similar to spike-response models [80] can provide additional insights into the importance of the long history dependency induced by plateau potentials to explain the current spiking behavior.

AN INCOMPLETE MODEL OF NEURAL COMPUTATION AND COGNITION. Despite numerous criticisms of deep learning and connectionist theories over the years [66, 160, 167], it is in many ways the only game in town that combines a plausible approach to cognition with the practicality of modelling the learning of a wide range of difficult artificial intelligence tasks.

In the SDT model, two puzzle pieces are still missing. Firstly, Hebbian plasticity mechanisms adapting both pre- and postsynapse can enable segments to find input populations that frequently emit synchronous spikes volleys. Experiments support the claim that plasticity favors coincident inputs and forms clusters to generate active dendritic responses Takahashi et al. [241]. Presynaptic release probabilities are also set by local dendritic activity [18]. The challenge is that these synaptic adaptations are dominated in experiments by the local plateau signal itself, which is absolutely required for long term potentiation (LTP) [19], and not by signals from other parts of the neuron such as backpropagating action potentials [90, 143]. While these results emphasize the importance of active dendritic events and dendritic plateaus for learning, they also pose a problem. The individual segments in SDT neurons however must learn to cooperate to express salient features, otherwise the neuron will never engage in plateau computation because the gating cascade from distal dendrite segments to soma is never engaged. A potential solution may involve a combination of the local plateau potential and backpropagating action potential [265] that can still be larger than the previously initiated plateau if propagated without failure [75] and the calcium hypothesis for plasticity [61, 86] that suggests that synaptic adaptations change in strength and sign (LTP or LTD) based on duration and amplitude of locally induced calcium concentrations. Together with a global modulation of plasticity based on neuromodulators [203], there is reason to believe that the credit assignment problem may be solved in complex SDT neurons and rank-ordered codes can be discovered directly from data.

An exciting prospect for SDT neurons is the inclusion of other plasti-

city mechanisms that are typically rarely considered in computational models but would have significant and important effects in our model. This is seen most clearly in structural plasticity: Whether a particular afferent population connects to a distal or more proximal dendrite segment is what defines the computational function of any particular SDT neuron. Butz, Wörgötter and Ooyen [29] review variety of structural changes in cortical circuits that include spontaneous creation of new synaptic connections, reactive structural changes after lesions and in response to electrical activity, and neurogenesis. Recent evidence suggests that pro-brain-derived neurotrophic factor (proBDNF) may play a role as a reward or punishment signal in the development of synapses and dendritic spines [114, 271]. New brain imaging techniques also reveal that even the brain's white matter, the axon's insulation, changes and may be involved in plasticity that optimizes the synchronous transmission of signals [64].

In sum, while the theory still needs a convincing plasticity rule that can coordinate learning between segments based on signals that are locally available, there is a vast body of evidence emphasizing the importance of active dendritic processes for plasticity. Further, forms of plasticity that are harder to investigate in experiments and collected under the umbrella term structural plasticity may have significant implications for learning in segmented dendritic trees.

Secondly, the coordination by recurrent network connections to create, enhance and manipulate the rank-ordered code of spike volleys is crucially important to complete the model. This is what links the internal symbolic model of one neuron to another. Since the hippocampus is well known to generate sequential activations, several models have been proposed which can account for the typical theta and gamma sequential rhythms that activate cellular assemblies in sequence (e.g. [115, 250]). The specific response of recurrently connected neurons can also specifically change based on past activation due to dynamic synapses [168, 251] or the state of each neuron at the current time as in the HTM model of sequence memory [92].

Particularly interesting is work by Korndörfer et al. [126] that shows how the coherence of spikes from a population can be dependent on the familiarity of a population with that specific stimulus. In the context of SDT neurons, this gives an alternative encoding axis for populations as not only the firing probability but also the coherence of emitted spikes can be used to increase the number of spikes per spike volley. The bias, interpreted as previous familiarity with the input, of a population may be encoded along this dimension, meaningfully extending the coincidence population code used in our model.

A DIVERSITY OF SOLUTIONS. The fact that network coordination in SDT network models may yet again require a slightly different perspective on computation compared to the single SDT neuron is em-

blematic of a common theme in this thesis: Diverse problems require diverse solutions. We have seen that SDT neurons are particularly well suited to computation over rank-ordered events and may offer a mechanism for symbolic computation at the neural level, but they are a bad fit when a time-series must be predicted and velocity is an important signal.

Delay-coupled reservoirs are ideal candidates to create non-linear embeddings of such continuous signals in very simple systems with fast internal dynamics.

If the signal itself is already very fast and the information must be filtered on multiple slower timescales to extract the relevant signal, a concatenated filter may be the right approach.

The study of data-driven learning of differential equations showed that some problems may not need memory after all, if the generalized dynamics can instead be learned by a function approximating neural network in the deep learning tradition. This however shifted the problem to correctly evaluating a loss or credit function, and time is still an important factor in these system to match the desired velocity in the generated dynamics.

It comes as no surprise that all solutions – some more than others – potentially have an analogous implementation in neural systems. Biology generates diversity as a strategy. Integrating these diverse strategies into coherent models is therefore an important task. A computational account of behavior ultimately includes the perception of visual scenes and sounds to symbolic reasoning processes and back to a motor command that orchestrates a specific pattern of motor movements.

Neuromorphic computing as the basis for autonomous, artificially intelligent systems behaving in and interacting with a real and uncontrolled environment provides an excellent platform to develop such integrated computing platform of diverse solutions – the range of different problems to be solved justifies it.

The increase in diversity also comes with an explosion in complexity. This has led to computational and theoretical models in neuroscience focusing sharply on specific phenomena and feigning ignorance about the rest. This thesis has certainly followed this trend in parts. For example, do integrative computation and dendritic plateau computation coexist in pyramidal neurons? If diverse solutions must be considered, we must make an effort to reintegrate. What are the distinct components of this integrated standard model of neural computation?

CONCLUSION: THE MANY FACES OF TIME. We began the thesis by exploring how the concept of universal time familiar to us – at least in modern everyday life with alarm clocks and appointments – vanishes from the fundamental descriptions of the universe and then reappears, ordering events into past, present and future. Usually, this is where the story ends. Ask any physicist.

But, as we kept walking deeper into the woods of computational mechanisms with which we perceive the world ordered in time, its universality was once again lost. Different percepts seem attached to different and sometimes varying clocks that may or may not be synchronized to the clock of our computing devices and they certainly do not seem to care. Even in the perception of a single, simple scene, time shows its many faces. The processing of perceivable discrete sequences requires a different computer than the processing of a perceived continuous signal. Different clocks *and* different representations.

We made an effort to accept this challenge, process information as we receive it, and find strategies to remember what was important to decode the many orders of time. In doing so, we found that time is a strong organizational principle in the brain [31] and the basis for structure in internal representations on a fundamental level. When writing a piece of code to analyze some data, we often make an effort to remove or abstract the temporal dimension only to turn around and execute our program which sequentially feeds data and instructions to the CPU to compute the result.

Time and the structure it imposes are close friends to computation.

Part I

APPENDIX

APPENDIX: ADDITIONAL METHODS

A.1 IMPLEMENTATION OF THE NAVIGATION EXPERIMENT

To simulate the stochastic movements of an animal in a two-dimensional environment, 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:

$$\begin{aligned}
 dX &= \cos(2\pi A)Vdt \\
 dY &= \sin(2\pi A)Vdt \\
 dA &= 0.25dW_A \\
 dV &= 10.0(0.25 - V)dt + 0.1dW_V
 \end{aligned} \tag{17}$$

A represents the angular heading of the animal, V represents its velocity in m s^{-1} 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 $10 \text{ cm} \times 9.5 \text{ cm}$, 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 200 ms. Three populations of place cells, each 20 neurons strong, are centered on a hexagonal grid with center-to-center distance of $r \approx 2.9 \text{ cm}$. Each population randomly emits spike volleys following a homogeneous Poisson process with rate $\lambda = 50 \text{ Hz}$. 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 $\sigma = 9.7 \text{ mm}$, 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 5 Hz to emulate background activity. Each spike is transmitted through stochastic synapses independently with probability 0.5.

Each of the simulated neuron's dendrite segments receives spiking input from the 20 neurons of one population and requires either 6 or 3 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 $v = \frac{3r}{200 \text{ ms}} \approx 43 \text{ cm s}^{-1}$ 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.

A.2 SIMULATION FRAMEWORK FOR DENDRITIC PLATEAU COMPUTATION

All simulations are implemented in a custom package developed in the Julia programming language [13], publicly available via the code repository hosted at <https://github.com/jleugeri/DPC.jl> and a fork specific to the thesis at <https://github.com/pnieters/DPC.jl>.

The simulator implements the neuron model outlined in this paper using a fast and extensible event-based formalism. All experiments and configuration files can be found in the `examples` subfolder of the repository.

Further documentation of the simulator, its interfaces, and implementation details can be found there as well.

A.3 A METHOD FOR EXPERIMENTAL VERIFICATION.

The method presented here was jointly developed with Johannes Leugering and was part of a previous pre-print version of [136] found at <https://www.biorxiv.org/content/10.1101/690792v3.full.pdf> with minor corrections.

Figure 27 demonstrates how the relative timings of plateaus in dendritic trees can reveal whether they follow the assumption of gating of dendritic plateaus by dendritic plateaus. If plateau timings and initiations can be measured, this method could form the basis of experimental verification of this assumption and enable analysis of dendritic structure by plateau timings.

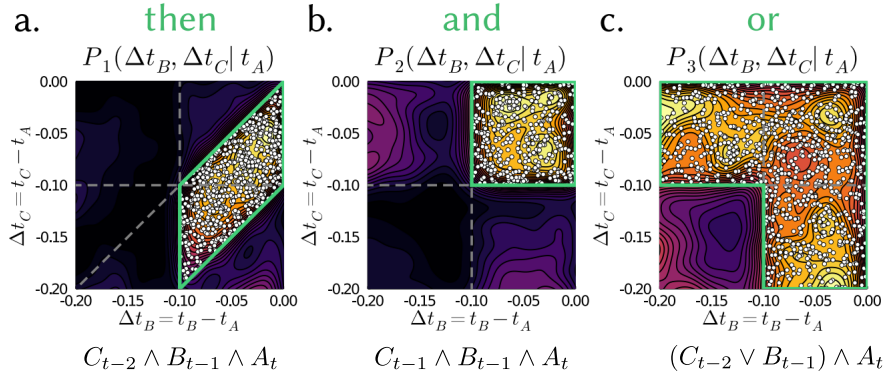


Figure 27: DENDRITIC MORPHOLOGY IMPOSES TIMING CONSTRAINTS.

The computational function implemented by the dendritic gating of plateau generation imposes constraints on the timing of plateau initiation that can be measured. Here, we show the timing of two connected segments, either consecutively or in parallel with threshold 1 or 2 in relation to a third segment in a joint distribution of $\Delta t_B, \Delta t_C$ of dendritic plateaus directly preceding a somatic spike or plateau event at t_A .

- If segments are connected consecutively, the distribution of timings has a parallelogram structure because B must always precede the spike at A, but C must have also preceded B by at most 100ms each in this configuration with plateau length $\tau_P = 100\text{ms}$. All measured events in a computationally produced sample fall into this boundary.
- A parallel AND connection instead enforces timings in a narrow 100ms window on both axes as plateaus must overlap but need not be ordered.
- A parallel OR connection results in a much broader distribution L-shaped joint distributions as a plateau in either segment would suffice.

PUBLICATIONS AND CONTRIBUTIONS

B.1 PAPER: A MINIMAL MODEL OF NEURAL COMPUTATION WITH
DENDRITIC PLATEAU POTENTIALS

Johannes Leugering, Pascal Nieters and Gordon Pipa. 'A minimal model of neural computation with dendritic plateau potentials.' In: *bioRxiv* (2021), p. 690792

Abstract

Over the last two decades, advances in neurobiology have established the essential role of active processes in neural dendrites for almost every aspect of cognition, but how these processes contribute to neural computation remains an open question. We show how two kinds of events within the dendrite, synaptic spikes and localized dendritic plateau potentials, interact on two distinct timescales to give rise to a powerful model of neural computation. In this theoretical model called dendritic plateau computation, a neuron's computational function is determined by the compartmentalization of its dendritic tree into functionally independent but mutually coupled segments. We demonstrate the versatility of this mechanism in a simulated navigation experiment, where it allows an individual neuron to reliably detect a specific movement trajectory over hundreds of milliseconds with a high tolerance for timing variability. We conclude by discussing the implications of this model for our understanding of neural computation.

A MINIMAL MODEL OF NEURAL COMPUTATION WITH DENDRITIC PLATEAU POTENTIALS.

Johannes Leugering*

Fraunhofer Institute for Integrated Circuits
johannes.leugering@iis.fraunhofer.de

Pascal Nieters*

Osnabrück University, Germany
pnieters@uni-osnabrueck.de

Gordon Pipa

Osnabrück University, Germany
gpipa@uni-osnabrueck.de

May 31, 2021

ABSTRACT

Over the last two decades, advances in neurobiology have established the essential role of active processes in neural dendrites for almost every aspect of cognition, but how these processes contribute to neural *computation* remains an open question. We show how two kinds of events within the dendrite, *synaptic spikes* and localized *dendritic plateau potentials*, interact on two distinct timescales to give rise to a powerful model of neural computation. In this theoretical model called *dendritic plateau computation*, a neuron's computational function is determined by the compartmentalization of its dendritic tree into functionally independent but mutually coupled segments. We demonstrate the versatility of this mechanism in a simulated navigation experiment, where it allows an individual neuron to reliably detect a specific movement trajectory over hundreds of milliseconds with a high tolerance for timing variability. We conclude by discussing the implications of this model for our understanding of neural computation.

1 Introduction

1 The vast majority of neural tissue is occupied by neural dendrites [1], the extensively branching tree structures on which
2 almost all synapses terminate. Yet, most simplified neuron models have focused on the diverse dynamics underlying
3 somatic spike generation [2]. In recent years, however, it has become increasingly evident that the computational
4 function of a neuron is largely determined by properties and dynamics of its dendrite [3]. In particular, a steadily
5 increasing number of studies find dendrites that generate active responses to spiking input. Further, these active dendritic
6 processes appear to be fundamental to brain function [4, 5, 6, 7]. The observed effects range from short-lived Na^+
7 spikes [8] to the particularly striking dendritic plateau potential [9] — an intricate dynamic response mediated by
8 NMDA receptors that maintains a strong depolarization of the local dendritic membrane potential for long periods
9 of time. The complexity of these mechanisms and the diversity of neuron types makes determining the right level of
10 abstraction for a single model of neural computation difficult [10].

11
12 For example, a recent study by Ujfalussy et al. [11] concluded that the somatic membrane potential in layer II/III
13 pyramidal neurons can largely be explained by a linear statistical model. A simple non-linear model can improve the
14 result, but the introduction of additional non-linearities only leads to minor improvements. Li et al. [12] also argue, that
15 the properties of dendritic integration can be approximated well by a point-neuron model, if specific synaptic current
16 effects are incorporated. On the other hand, seminal work by Poirazi et al. [13] suggests that in fact a 2-layer artificial
17 neural network may be necessary to capture the input-output mapping of a single neuron, implying that the neuron's
18 expressive power may be on a similar level. Whereas these earlier results analyzed the neuron in a static framework,
19 Beniaguev et al. [14] incorporate temporal dynamics as well, and instead conclude that a temporally convolutional deep

*Both authors contributed equally.

20 neural network is better suited to model the single neuron's behavior. However, it is difficult to draw direct conclusions
21 about the computational capabilities of a complex neuron from the complexity of a quantitative model of the neuron's
22 membrane potential.

23 The most popular metaphor for neural computation to date is the linear-nonlinear (LN) point neuron that inspired the
24 development of artificial neural networks [15]. LN neuron models make no use of dendritic complexity at all, and
25 instead rely on a complex network of synaptic interconnections between individually simple neurons. This approach
26 has worked exceedingly well for deep learning [16] and provides a compelling model for some forms of fast sensory
27 processing e.g. in early visual areas [17], but it neglects a dimension critical for any interaction with the real world —
28 time. Many behavioral or higher cognitive tasks require the ability to integrate, process and retain information across
29 multiple, dynamically varying time scales. Consider, for example, a rodent navigating an environment in search of food.
30 Receptive fields of place and grid cells tile a spatial map of this environment and encode the current position by their
31 population activities [18, 19]. To navigate successfully, the animal needs to know not only its present location, but also
32 the path it took through the environment. Decoding this path from sequential place and grid cell activity requires the
33 integration of information on behavioral time scales that can span hundreds of milliseconds or more [20, 21]. Similar
34 long sequential patterns can be found also in olfaction [22, 23] and cortical auditory processing [24], and they are likely
35 involved in higher cognitive tasks such as language understanding, as well.

36 But how can such long temporal patterns of neural activity be processed by volatile neurons with membrane dynamics
37 on the time scale of only tens of milliseconds or less [25]?

38 Our main idea is this: Neurons with active dendrites that generate dendritic plateau potentials have a form of working
39 memory on a much longer time scale than that of individual spike responses. The interaction of these dendritic plateaus
40 establishes a computation that enables single neurons to process information on long time scales and in a structured
41 way. We derive this concept from a wealth of recent biological findings, which we categorize into four fundamental
42 modelling assumptions (Section 2), and find that:

- 43 • A qualitative neuron model captures dendritic plateau computation in a tree structure of dendrite segments. The
44 computational complexity of the model results from the interactions of these segments on the long timescale
45 of plateau potentials (Section 3)
- 46 • The computational capabilities of dendritic trees can be characterized by a small set of elementary motifs
47 (Section 4)
- 48 • Dendritic segments can robustly decode sequential activations of neuron populations. An example of path
49 detection from place cell activity illustrates two key properties of dendritic plateau computation, timing
50 invariance and a graded stochastic response, that lead to an intricate spatiotemporal receptive field (Section 5)
- 51 • Single neurons can implement structured computations over symbol-like inputs, motivating a new view on
52 neural computation (Section 6)

53 2 Biological evidence for neural computation based on dendritic plateaus

54 **Active generation of localized dendritic plateau potentials.** Most of a cortical pyramidal neuron's excitatory synaptic
55 inputs terminate on dendritic spines [26], where post-synaptic ion channels are activated via the stochastic, pre-synaptic
56 release of glutamate-carrying vesicles [27, 28]. The activated channels, primarily controlled by α -amino-3-hydroxy-
57 5-methyl-4-isoxazolepropionic acid receptors (AMPA) [29], become conductive to a mixture of ions, which leads
58 to a brief depolarization in the corresponding spine, referred to as the *excitatory post-synaptic potential* (EPSP) [30].
59 In addition to AMPARs, the synaptic release of glutamate can also activate N-methyl-D-aspartate receptor (NMDAR)
60 gated ion-channels [29], but they do not become conductive unless a channel-blocking Mg^{+} ion is first displaced by a
61 sufficiently strong depolarization [31, 32]. However, coincident EPSPs from multiple nearby spines can accumulate
62 and thus induce this required depolarization of the local dendritic membrane potential [33]. Experimental as well as
63 simulation studies report that this requires a volley of 4-20 or even up to 50 spikes within 1 ms to 4 ms, depending
64 on the location along the dendritic tree [33, 34, 35, 36]. The opening of NMDAR channels triggers a massive influx
65 of different ionic currents that lead to a full depolarization of the local dendritic membrane potential. Although the
66 isolated NMDAR response itself is reported to only last on the order of around 25 ms [37], in vivo recordings reveal
67 that voltage-gated channels in the dendritic membrane [38] prolong this effect, resulting in an actively maintained
68 depolarization that can last from tens to hundreds of milliseconds [39] (see **Fig. 1b** for an illustration of this mechanism).
69 Such active long-lasting dendritic processes, *dendritic plateau potentials*, are ubiquitous [9, 40], and provide neurons
70 with potentially useful memory traces that can last hundreds of milliseconds. Because NMDAR channels are gated by
71 both depolarization and the presence of glutamate, plateau potentials remain localized, and do not actively propagate
72 along the dendrite [41].

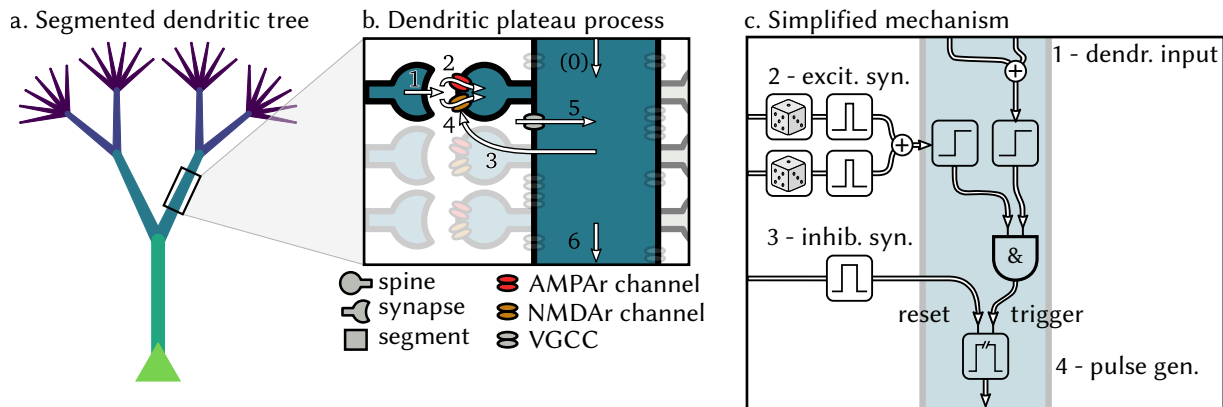


Figure 1: How dendritic plateau potentials are generated, and how they interact. (a.) A stylized neuron with dendritic arbor. **(b.)** Summary of the biological processes involved. A spike (1) releases glutamate, which opens AMPAR-gated ion-channels that depolarize the post-synaptic spine and cause an EPSP (2). If sufficiently many EPSPs coincide with up-stream dendritic input (0), the local membrane potentials rises (3) and NMDAR-gated ion-channels become de-inactivated, causing a further localized depolarization (4). Additional voltage-gated calcium channels can amplify and prolong this process (5) and cause a plateau potential, which can in turn moderately depolarize the parent segment (6 & 0). **(c.)** An algorithmic approximation of the biological mechanisms. If a dendrite segment is depolarized by sufficiently strong input from its child segments (1) and receives sufficiently strong excitatory input from its stochastic synapses (2), a local plateau potential is initiated. If the plateau is not interrupted by shunting inhibitory input (3), it depolarizes the parent segment for an extended period of time.

73 **Passive asymmetrical propagation of membrane potentials.** The passive propagation of membrane potential through
 74 the dendrite is described by neural cable theory [42]. Only for very specific branching patterns, the complex dendritic
 75 tree can be reduced to an equivalent model of a cylinder, *Rall's ball-and-stick model*, in which the contribution of
 76 individual synaptic inputs sum (sub-)linearly and nearly instantaneously [43]. Due to its simplicity, this special case is
 77 often used to motivate abstract point-neuron models such as leaky integrate-and-fire neurons, which ignore the spatial
 78 dimension of the dendritic tree entirely and instead model the neuron as if it were a single electric compartment [44].
 79 But in general, the passive spread of membrane potentials depends on the morphology and electrical impedance of
 80 the dendritic tree, specifically on the relationships of branch-diameters at the branching points [8]. For example, a
 81 back-propagating action potential moving in retrograde direction from soma to apical dendrite is only slightly attenuated
 82 if the dendritic branches become progressively thinner. However, the attenuation of signals in the anterograde direction
 83 is so strong, that synaptic input onto thin apical dendrites has little measurable effect on the membrane potential at the
 84 soma [45, 38]. One proposed solution to restoring “dendritic democracy” [46] and ensuring similar contribution of
 85 all synapses to the somatic membrane potential regardless of its position along the dendrite is an increased synaptic
 86 efficacy at distal synapses, which has been observed in hippocampal pyramidal neuron [47]. Another solution is
 87 the active amplification of distal synaptic input by active dendritic processes, which was shown to be required for
 88 somatic spiking [48]. Complex spike bursts in particular require the activation of NMDA receptors [49]. The resulting
 89 plateau potentials are also subject to anterograde attenuation along the dendritic cable, and thus only have a moderately
 90 depolarizing effect on their immediate neighborhood [50]. This effectively raises the local resting potential for the
 91 duration of the plateau potential, thus lowering the amount of coinciding spikes required to initiate a plateau potential in
 92 this neighbourhood [51].

93 **Functional compartmentalization of dendrites.** The structure of dendritic arbors has long been conjectured to play an
 94 important role for neural computation. Koch et al. [52] calculated, that due to impedance mismatch at branching points
 95 in the dendrites of various types of retinal ganglion cells, distinct electrically isolated *functional subunits* emerge, i.e.
 96 regions with a roughly equal local membrane potential throughout, which are only weakly coupled to their neighboring
 97 regions. For example, experiments in rats confirmed that thin dendrites in neocortical pyramidal neurons can act as
 98 independent computational subunits that provide neurons with an additional non-linear integration site, increasing
 99 the potential computational power of the single neuron [53]. This behavior is not limited to pyramidal neurons, but
 100 rather appears to be a general principle that can be found in various forms across different cell types. For example,
 101 Purkinje cells in the cerebellum also generate localized Ca^{2+} events in response to coincident input on individual
 102 dendrite segments [54, 55], and thalamo-cortical neurons respond to strong synaptic input by localized plateaus in distal
 103 dendritic branches [56]. Branco and Häusser [57] identify such functional subunits with individual dendritic branches,

104 which they suggest constitute the “atomic unit” of computation in neural systems.
105 Rather than a single branch, one such functional compartment can also, stretch across multiple nearby branches, as
106 long as synapses carrying correlated input signals cooperate to trigger local, regenerative events. Wybo et al. [58]
107 present evidence for such compartmentalization and point out that structural plasticity may even allow the neuron to
108 dynamically change the structure of its functional compartmentalization. We view dendrites as complex structures
109 composed of functional subunits in this sense and will refer to them as *dendrite segments*.² The segmentation implies,
110 that only nearby synapses cooperate to trigger local regenerative events such as plateau potentials. To effectively drive a
111 neuron, inputs therefore have to be clustered such that correlated spikes arrive at the same dendrite segment at the same
112 time. This is observed in experiments [59] and suggests an alternative view of spike-based communication in which
113 clustered groups of synaptic spines receive highly synchronized spiking inputs [60]. Since the resulting simultaneous
114 EPSPs are required to trigger the NMDAr response, we therefore consider these highly synchronized *spike volleys* as
115 the atomic unit of spike-based communication.

116 **Stochastic excitation and shunting inhibition.** An AMPAr or NMDAr response to an afferent neuron’s input spike
117 requires the prior release of neurotransmitter at the pre-synaptic terminal of a synapse. This process, however, is
118 stochastic and best described by a “quantal” theory of neurotransmitter release [27, 28], according to which the
119 successful transmission of a spike at a synapse is a random event with probability p_r . Branco et al. [61] found that in
120 hippocampal synapses, p_r is distributed with a median of 0.22, which emphasizes the fundamentally stochastic nature
121 of neural computation. The considerable variance of this distribution can be largely explained by the location of the
122 synapse in the dendritic tree. The release probabilities of nearby synapses are much more homogeneous, which provides
123 further evidence of the aforementioned functional segmentation of the dendrite.
124 AMPAr mediated EPSP responses as well as NMDAr mediated plateau responses can also be modified by inhibitory
125 input at GABA_A or GABA_B synapses. The resulting shunting inhibition current can have both a subtractive and divisive
126 effect on post-synaptic membrane potential [62]. The effect of shunting inhibition on active dendrites can be even more
127 dramatic, outright stopping the generation of plateau potentials [63]. The interactions of plateaus and inhibition can
128 be intricate [64], and inhibitory synapses tend to be placed critical positions within the dendrite to control dendritic
129 excitability [65] or gate layer specific input from reaching the soma [66].

130 3 A computational model for dendritic plateau computation

131 From the biological observations outlined above, we derive a simple, qualitative model of active dendrites. At the core
132 of this model lies the interaction of two types of events on distinct time-scales — short, spike-triggered EPSPs and long,
133 actively generated dendritic plateau potentials — in a tree structure of dendrite segments. We define a *segment* as a
134 minimal part of the dendritic tree, e.g. a single physical branch or stretching across multiple branches, that behaves as
135 one functional, electrically isolated integration site. In other words, the synaptic inputs of a segment can cooperatively
136 generate a plateau potential that stays confined to the segment. These dendrite segments form a tree structure with the
137 soma at its root and thin dendrite branchlets as leaves.

138 Let’s consider the function of one individual dendrite segment i in more detail (see figure 1c for a schematic).
139 We distinguish excitatory and inhibitory synapses, which respectively produce excitatory (EPSPs) and inhibitory
140 postsynaptic potentials (IPSPs). An excitatory synapse from neuron k to segment i only successfully transmits each
141 spike with probability $p_{i,k}$. If the synapse transmits the spike, it induces an EPSP $\kappa_E(t)$ with duration τ_E and a
142 magnitude $w_{i,k}$, which depends on the synaptic efficacy. Likewise for an inhibitory synapse, only that the duration τ_I of
143 the IPSP is typically slightly longer. We model the shape of the post-synaptic potentials by rectangular pulses:

$$\kappa_E(t) = \begin{cases} 1 & \text{if } 0 \leq t \leq \tau_E \\ 0 & \text{otherwise} \end{cases}, \quad \kappa_I(t) = \begin{cases} 1 & \text{if } 0 \leq t \leq \tau_I \\ 0 & \text{otherwise} \end{cases}$$

144 We use exc_i and inh_i to represent the set of excitatory and inhibitory neurons targeting segment i , we denote the time
145 of the m^{th} spike by neuron k with t_k^m , and introduce the i.i.d. random variables $\xi_{i,k}^m \sim \text{Bernoulli}(p_{i,k})$ to simplify
146 notation. We can then define the combined effect of excitatory as well as inhibitory input for segment i ³:

²We avoid the term “compartment” to prevent confusion with the concept of multi-compartment neuron models, which are commonly used as a spatially discretized solution to partial differential equation models of neurons.

³We assume that spike arrival times $t_{i,k}^m$ are at least τ_E apart.

$$\text{EPSP}_i(t) = \sum_{k \in \text{exc}_i} \sum_{m | t_k^m \leq t} \xi_{i,k}^m w_{i,k} \kappa_E(t - t_k^m) \quad (1)$$

$$\text{IPSP}_i(t) = \sum_{k \in \text{inh}_i} \sum_{m | t_k^m \leq t} \xi_{i,k}^m w_{i,k} \kappa_I(t - t_k^m) \quad (2)$$

$$\text{PSP}_i(t) = \text{EPSP}_i(t) - \text{IPSP}_i(t) \quad (3)$$

147 One of the necessary preconditions for generating a dendritic plateau potential is a sufficiently strong net depolarization
 148 of the dendrite by synaptic input, i.e. larger than a segment-specific synaptic threshold TS_i , caused by the coincidence
 149 of multiple synchronous spikes. In thin dendrite branchlets, i.e. the leaf nodes of our tree structure, this is sufficient
 150 to trigger a plateau potential. But in the general case, additional depolarizing input from dendritic child branches is
 151 required. Here, we are only interested in the large depolarizing effects that actively generated plateau potentials have on
 152 directly adjacent segments, and we ignore the much weaker passive propagation of sub-threshold voltages along the
 153 dendrite.

154 We therefore introduce additional notation: child_i denotes the set of the direct children of segment i (if any), and
 155 $O_k(t), k \in \text{child}_i$ is the effect that the child segment k exerts on i at time t . Just like we did for the post-synaptic
 156 potentials, we can then define the total *dendritic input* $D_i(t)$ into segment i :

$$D_i(t) = \sum_{k \in \text{child}_i} O_k(t) \quad (4)$$

157 The segment-specific dendritic threshold TD_i determines, how much dendritic input is required in addition to synaptic
 158 input to trigger a plateau potential in segment i . For leaf nodes of the dendritic tree, i.e. segments without any children
 159 of their own, we set $\text{TD}_i = 0$.

When both conditions become satisfied, i.e. there is sufficient synaptic and dendritic input, then a plateau potential is initiated. We use T_i^m to denote the starting-time of the m^{th} plateau potential in segment i :

$$T_i^m = \min t \geq T_i^{m-1} \quad \text{such that} \quad \text{PSP}_i(t) \geq \text{TS}_i \wedge D_i(t) \geq \text{TD}_i \quad (5)$$

160 The plateau then typically ends at time $\tilde{T}_i^m = T_i^m + \tau_P$ after the fixed duration τ_P , unless it is interrupted by
 161 shunting inhibition, or it is prolonged by additional synaptic inputs.⁴ We formalize these special cases as follows: The
 162 first inhibitory spike, if any, from neuron $k \in \text{inh}_i$ at time $t_k^l \in [T_i^m, T_i^m + \tau_P]$ can end the plateau, i.e. in that case
 163 $\tilde{T}_i^m = t_k^l$. Otherwise, if another plateau is triggered at time $T_i^{m+1} \in [T_i^m, T_i^m + \tau_P]$ before the previous plateau has
 164 run its course, the first seamlessly flows into the second, i.e. $\tilde{T}_i^m = T_i^{m+1}$.

165 We can now define the output of segment i as a sequence of binary pulses, the plateau potentials:

$$O_i(t) = \begin{cases} 1 & \text{if } \exists m : t \in [T_i^m, \tilde{T}_i^m] \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

166 This formalism can be iteratively applied to all segments of a neuron, including the soma, only that the segment produces
 167 a spike event followed by a brief refractory period τ_{refrac} instead of each long-lasting plateau potential.⁵

168 Conceptually, each dendrite segment acts first and foremost as a coincidence detector for a volley of synchronized
 169 spikes on the fast time scale of EPSPs. On the second, slower time scale of dendritic plateaus each segment is gated
 170 by its children in the dendritic tree. The computation of the neuron thus depends on a sequence of activations of its
 171 segments by spike volleys, which can be interrupted by shunting inhibition.

172 4 Motifs of dendritic plateau computation

173 The structure of the tree of dendrite segments determines which activation patterns lead to dendritic plateau potentials in
 174 all dendrite segments of a neuron, and therefore determines the computation implemented by the neuron. Each segment

⁴In engineering terms, this resembles a re-triggerable monoflop with reset.

⁵In addition to the forward-propagation of membrane potentials that we focused on so far (i.e. from child branches to the parent), the reverse direction typically has an even stronger effect — strong enough for the parent segment to depolarize its child segments by itself. To capture this effect, we recursively define that a neuron segment k 's membrane potential $V_k(t) = O_k(t) \vee V_i(t), k \in \text{child}_i$ is depolarized whenever either the segment itself or any of its ancestors produces a plateau potential. However, while this peculiarity may be relevant for learning, it cannot impact the forward model of dendritic plateau computation that we present here.

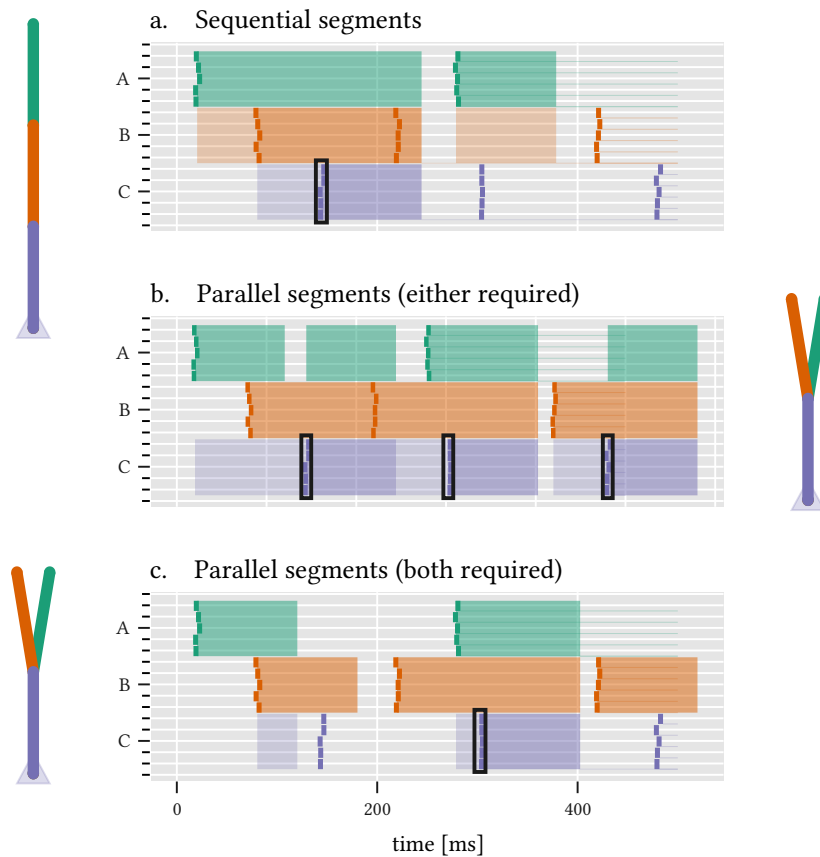


Figure 2: Various dendrite motives respond to different inputs. dendrite segments A, B and C receive spike volleys from corresponding neuron populations (color coded). We indicate for each segment, when it is enabled by its children (weakly shaded) or in a plateau state itself (shaded). **(a.)** If segments A, B and C form a chain, then C can only be activated while B is in a plateau state, whereas B can only be activated while A is in a plateau state. **(b.)** If both A and B are child-branches of C, either of which suffices to enable C, then C can be activated at any point where either A or B is in a plateau state. **(c.)** If both A and B are required, then C can only be activated while both A and B are in a plateau state.

175 is a coincidence detector for spike volleys, but additional input from a number of child segments may be necessary for
 176 a plateau potential to be triggered. The morphology of the dendritic tree defines these parent-child relations and the
 177 thresholds TD for required dendritic input. Changing these two variables changes the computation implemented, which
 178 we demonstrate in three prototypical motifs of dendritic plateau computation. Inhibition augments these motifs, for
 179 example to increase the specificity of pattern detection. Finally, stochastic synapses turn the otherwise deterministic
 180 neuron into a probabilistic pattern detector.

181 In the following examples, we look at neurons with several dendrite segments, each of which is connected to a small
 182 population of neurons that occasionally emits a volley of synchronized spikes. We are primarily interested in which
 183 patterns of spike volleys successfully trigger a somatic spike, and which do not. All experiments are simulated using
 184 open-source software (Section 8.2).

185 Dendrite structure determines computation

186 For example, dendritic segments can form a chain (**Fig. 2 a**), where each segment requires the previous one to be active
 187 ($TD_i = 1$). A spike volley of at least five coincident spikes ($TD_i = 1$) can therefore only trigger the most proximal
 188 segment, if a specific sequence of spike volleys activates each segment in the chain in the correct consecutive order.

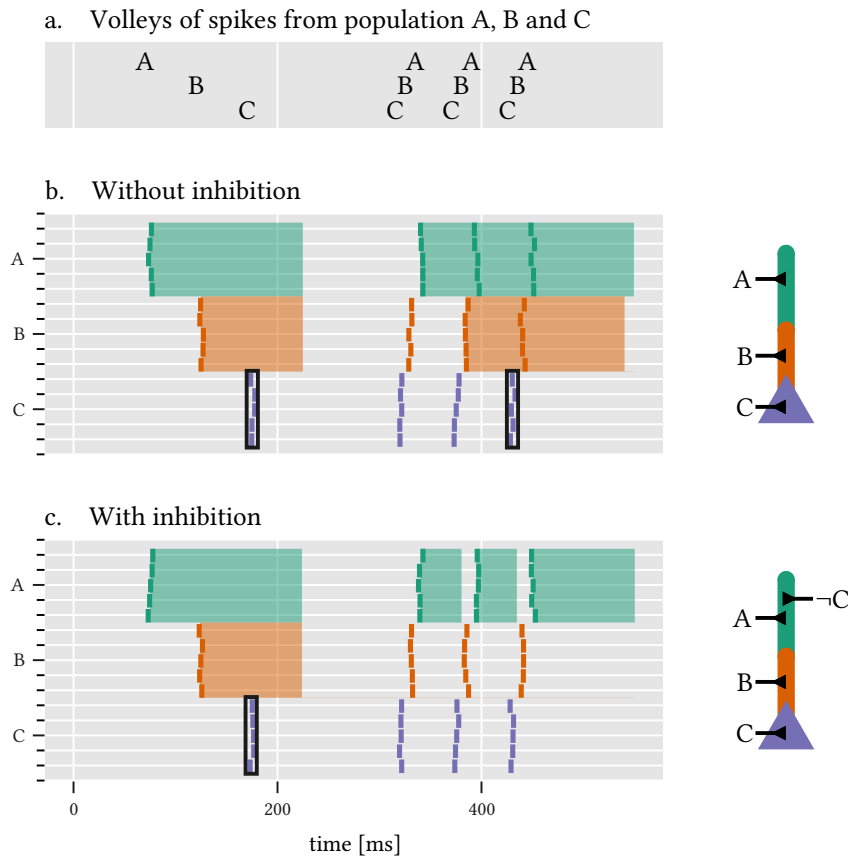


Figure 3: Shunting inhibition can prevent false detections. **(a.)** A neuron receives a sequence of spike volleys from three populations A , B and C . **(b.)** A neuron with a chain of dendrite segments A and B and soma C fires whenever they are activated in the correct order $A \rightarrow B \rightarrow C$, e.g. at time t_1 . This also results in a false detection at t_2 if the desired sequence $A \rightarrow B \rightarrow C$ is contained in fast repetitions of the undesired sequence $C \rightarrow B \rightarrow A$. **(c.)** By adding shunting inhibition, the false detection at t_2 can be prevented.

189 However, because plateau potentials last for a comparatively long duration, the exact timing of the spike volleys within
 190 these time-intervals is not crucial, making the detection of spike-volley sequences largely timing invariant.

191 In the example **Fig. 2 a**, a spike volley from population A can trigger a plateau in the first segment, which in turn
 192 enables the second segment for the duration of the plateau $\tau_P = 100$ ms. If a volley from population B arrives at the
 193 second segment during that time-interval, it will trigger another plateau there, which in turn enables the third segment,
 194 and so on. If the first segment is not triggered first, or if its plateau has already ended, a volley from population B
 195 to the second segment remains ineffective. The chain of three segments shown here would therefore detect the sequence
 196 $A \rightarrow B \rightarrow C$ of spike volleys from populations A , B and C in that order, as long as the volleys come within 100 ms of
 197 each other or less.

198 What happens when one dendrite segment branches into two child segments? If either of the two child segments can
 199 provide enough dendritic input to enable the parent segment, i.e. $TD_i = 1$ (see **Fig. 2 b**), then a spike volley from
 200 either population A or B can trigger a plateau that enables the third segment. In this motif, the third segment fires
 201 whenever a volley from C occurs within 100 ms after a volley from A or a volley from B . If instead the dendritic input
 202 from both child segments is required, i.e. $TD_i = 2$ (see **Fig. 2 c**), then the third segment only fires if a volley from C
 203 occurs within 100 ms after both a volley from A and B .

204 Shunting inhibition prevents false positives

205 So far, we only looked at excitatory synaptic inputs and how they generate plateau potentials, but the shunting effect of
206 inhibitory synapses plays an equally important role. To illustrate this, consider the example in **Fig. 3**. Our objective
207 is to detect (within some timing constraints) any sequence $A \rightarrow B \rightarrow C$ of spike volleys from the populations A , B
208 and C in that order. A chain of two dendrite segments and the soma, a motif we already saw above, will do just that.
209 However, if we rely solely on excitatory synaptic input, any additional unnecessary inputs have no effect. This may be
210 desirable in some cases but it may lead to false positives in others. For example, we might actually want to recognize
211 the sequence $A \rightarrow A \rightarrow B \rightarrow B \rightarrow C \rightarrow C$ to contain the desired sub-sequence $A \rightarrow B \rightarrow C$. But the sequence
212 $C \rightarrow B \rightarrow A \rightarrow C \rightarrow B \rightarrow A \rightarrow C \rightarrow B \rightarrow A$ shows three fast repetitions of the undesired sequence $C \rightarrow B \rightarrow A$,
213 while still containing the desired sub-sequence $A \rightarrow B \rightarrow C$, and the neuron would fire all the same (see **Fig. 3 a**).

214 To prevent the response to the anti-pattern $C \rightarrow B \rightarrow A$, we can add inhibitory input from population C to the two
215 dendrite segments tasked with detecting A and B (see **Fig. 3 b**). In that case, a volley from population C would
216 terminate any ongoing plateau potentials in these two segments, thus preventing a response to the undesired sequence
217 $C \rightarrow B \rightarrow A \rightarrow C \rightarrow B \rightarrow A \rightarrow C \rightarrow B \rightarrow A$ while leaving the response to the desired sequence $A \rightarrow B \rightarrow C$
218 unaffected. We can write this sequences with inhibition as $(A \wedge \neg C) \rightarrow (B \wedge \neg C) \rightarrow C$.

219 Shunting inhibition is therefore an important complementary mechanism for dendritic plateau computation, in particular
220 if we consider that in our model, inhibiting the output of one segment at a branching point can effectively “veto” the
221 entire computation of the corresponding subtree.

222 Stochastic synapses enable probabilistic computation

223 The various motifs shown above in combination with shunting inhibition can realize a wide range of operations via
224 dendritic plateau computation. However, this mechanism responds to a rather long sequence of incoming spike volleys
225 (potentially hundreds of milliseconds long) with an all-or-none response, i.e. a somatic spiking or nothing. Because the
226 inputs to a neuron are also typically noisy, this might make an individual neuron’s output too sparse and unreliable
227 to base important decisions on it. We can overcome this problem, because the inherent stochasticity of synaptic
228 transmission turn the probability that the neurons response into a graded response.

229 Let’s consider an individual dendrite segment $i = 0$, that receives a spike volley from a population of $n = 10$ neurons
230 (see **Fig. 4 a**). If each synapse independently transmits each spike it receives with the same probability P_0 , then the
231 number of actually transmitted spikes in a spike volley is a binomial random variable $\sim \text{Binomial}(P_0, n)$, and the
232 probability that this number suffices to trigger a plateau potential depends on both P_0 and the segment’s synaptic
233 threshold TD_0 . For a given threshold, the plateau probability $P_{\text{plateau}} = f(P_0)$ is hence a non-linear, sigmoidal function
234 of both the volley size and the synaptic transmission probability. Despite the fact that the neuron has an all-or-none
235 response for any individual spike volley, the expected value of its output, i.e. the probability to fire, is non-linear, graded
236 response that reflects the size of the incoming volley.

237 If we extend this analysis to motifs of multiple dendrite segments, then the neuron’s probability to fire is a non-linear
238 function of the size of all incoming spike volleys. For example, to trigger a chain of two sequential segments with high
239 probability, both segments have to be individually triggered with high probability, i.e. the neuron will only respond with
240 high probability if both incoming spike volleys are large (see **Fig. 4 b**). The AND-like operation between plateaus that
241 we observed in the deterministic case thus becomes a multiplication $P_{\text{chain}} = f(P_1) \cdot f(P_2)$ of plateau probabilities in
242 the stochastic case. Similarly, if only one of two parallel segments needs to be activated (see **Fig. 4 c**), this happens
243 with a probability $P_{\text{or}} = 1 - (1 - f(P_1)) \cdot (1 - f(P_2)) = f(P_1) + f(P_2) - f(P_1) \cdot f(P_2)$. The shown simulation
244 results confirm this prediction.

245 This procedure can be applied inductively to more complex dendritic trees, as well. The neuron responds with a
246 probability that depends on the size of all incoming spike volleys, and the expected value of the spike response thus
247 encodes the “confidence” of the neuron in the result of a computation or detection. By combining multiple neurons
248 with identical structure and synaptic input from the same source populations, we can construct an ensemble of neurons
249 with a graded, probabilistic response. This ensemble can then respond to any potentially relevant sequence of incoming
250 spike volleys with a volley of its own, such that the size of the emitted volley encodes the “confidence” of the ensemble
251 in this detection.

252 5 Detecting movement trajectories from place cell activity

253 A good example to illustrate how dendritic plateau computation can function in a close-to-real-world example is
254 the detection of sequential patterns in place cells. The location of an animal in its environment is represented by

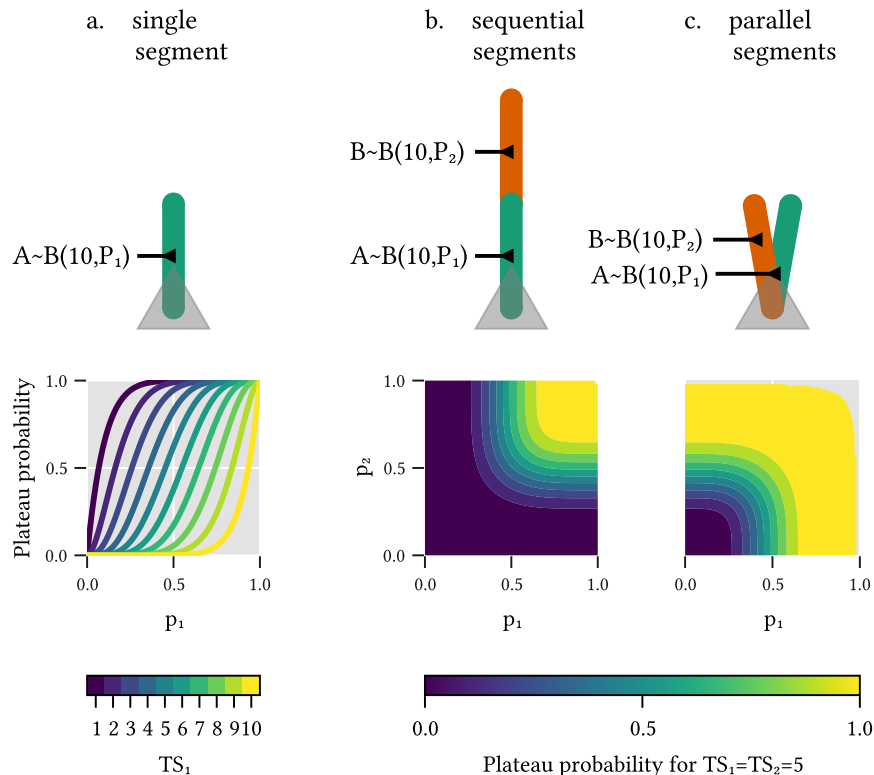


Figure 4: Stochastic synapses allow for graded responses. **(a.)** Out of a volley of ten spikes, the number of transmitted spikes is a Binomial distribution $B(10, P_1)$. A single dendrite segment, excited by such a volley, thus generates a plateau with a probability that depends on the synaptic transmission probability P_1 and the threshold TS_1 (color coded). **(b.)** Assuming two appropriately timed spike volleys activate two chained segments, then the probability that the second segment fires depends on both transmission probabilities P_1 and P_2 of the two synapse populations. The result resembles a probabilistic AND-gate. **(c.)** The probability of triggering at least one of two parallel segments resembles a probabilistic OR-gate.

255 place-cells [18, 19], each of which has a “receptive field” centered at a specific location. Navigation naturally produces
 256 sequential activation patterns as different locations are visited. The time scale of these patterns can be long and is
 257 variable because it is directly linked to the movement speed of the animal [21]. Further, active dendritic process
 258 have been shown to be selective for specific sequences of synaptic inputs [67] and dendritic spikes occur much more
 259 frequently in cortical pyramidal neurons of freely moving rats [68]. Applying our model to the problem of path decoding
 260 at varying movement shows how single neurons can solve this detection problem across multiple time scales.

261 We numerically simulate a rat moving through a small, 2-dimensional environment by generating stochastic paths at
 262 varying movement speed (more details in Section 8.1). The environment is tiled in a hexagonal grid by the receptive
 263 fields of place cell populations, each 20 neurons strong. These populations emit spike volleys with a varying magnitude
 264 that depends on the animal’s distance to the center of the respective receptive field (**Figure 5 a and b**). Dendritic
 265 plateau computation allows a single neuron to detect specifically those paths, that traverse the receptive fields of three
 266 place cell populations in the correct order: from the bottom left (in green) through the center (in orange) to the top
 267 right (in purple). The neuron is composed of two sequentially chained dendrite segments and the soma, each of which
 268 receives synaptic input from exactly one of the place-cell populations and requires $TS_i = 8$ coincident spikes to fire a
 269 plateau. In the presence of noise, this requires the fast detection of coincident spikes from each place cell population in
 270 order to distinguish legitimate spike volleys from background noise, as well as the interaction of long-lasting plateau
 271 potentials to detect the slow transition from one receptive field to the next on a behavioral time-scale. The problem thus
 272 has to two distinct time scales: fast estimation of the current location and slow integration of the traversed path.

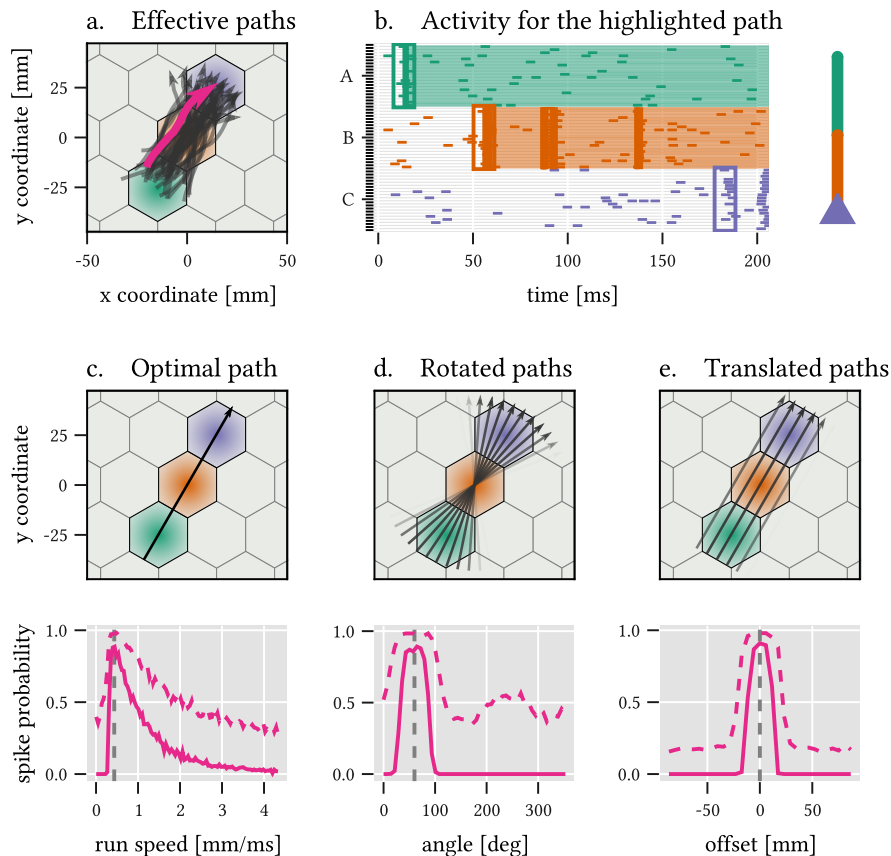


Figure 5: A simple neuron with three dendrite segments as shown to the right of panel b can detect directed paths on a timescale of 300 ms. **(a.)** The receptive fields of place cell populations tile the environment into a hexagonal grid. Random trajectories are generated through a stochastic process with randomized initial positions, velocities and angular heading to simulate the animal's movements. Only those trajectories are shown that elicit a spike response by the neuron. **(b.)** While the animal follows the highlighted trajectory (purple) through space, the place cell populations generate spikes, which in turn trigger plateau potentials in the corresponding dendrite segments (color coded). After initiation, plateau potentials can be extended by super-threshold inputs, as shown by the vertical lines. **(c.)** The neuron responds with highest probability to a path that traverses the center of the desired receptive fields at an optimal speed (top). Varying the movement speed (bottom) affects the probability of the neuron to detect the sequence (solid line). For a decreased threshold, the neuron's sensitivity is decreased, and the overall firing-probability is increased (dotted line). **(d.)** Changing the orientation or **(e.)** laterally shifting the path away from the optimal path rapidly decreases the probability of the neuron to fire, as well. Note that while the neuron is highly selective to orientation and offset, the firing probability only gradually changes as speed varies over an order of magnitude.

273 We can characterize the behavior of this path-detecting neuron by changing the speed, orientation and lateral offset of
274 the path through then environment, and recording the neuron’s probability to respond with a spike (**Fig. 5 c-e**). Firstly,
275 the response probability for the optimal path is largest (almost 90 %) with an optimal run-speed of around 0.5 m s^{-1} ,
276 but even for a three times faster run speed, the neuron would still be able to detect it with roughly 30 % probability.
277 This is due to the fact that the dendritic plateau computation is, within some limits, invariant to the specific timings of
278 individual spike volleys. However, if the animal moves so slowly that the time-difference between spike volleys exceeds
279 the plateau duration, or if it moves so quickly that a place-cell population fails to produce a spike volley, at all, then
280 the probability to fire decreases. If a lower threshold $TS_i = 4$ is chosen for each segment i , the sensitivity decreases,
281 because even locations relatively far from the center of a receptive field can occasionally generate spike volleys of small
282 magnitude (**Fig. 5 c dashed line**).

283 Secondly, our model neuron can be highly sensitive to the specific orientation and lateral offset of the desired path
284 in space (**Fig. 5 d and e**). For a high threshold $TS_i = 8$, only a narrow range of $\pm 30^\circ$ around the optimal direction
285 of 60° are reliably detected. By lowering the threshold to $TS_i = 4$, each segment can be made less selective and
286 the orientation-specificity of the neuron decreases substantially (**Fig. 5 d dashed line**), but as a side-effect the false
287 detection probability also increases. When we shift the path orthogonally to the direction to the optimal path, only
288 those paths shifted by at most $\pm 10 \text{ mm}$ from the center are reliably detected (**Fig. 5 e**). Just like for the rotated paths,
289 decreasing the plateau firing threshold in each segment decreases sensitivity and increases the noise floor in the neuron’s
290 expected response.

291 6 Structured computation in single neurons

292 The issue of how working memory bridges the fast time scales of synaptic responses and the slower time scales of
293 behavior is often addressed in recurrent networks of neurons, for example by slow emergent network dynamics [69], by
294 fast synaptic plasticity [70], or as fading memory inherent in network dynamics [71]. However, if neurons generate
295 dendritic plateaus in dendrite segments, then they already have access to internal memory. Moreover, this internal
296 memory enables structured computation: the plateaus maintain a hidden state that can only be advanced if the correct
297 input is seen at the correct segment in the correct time-frame. More formally, the single neuron is a (hidden-) state
298 machine that accepts expressions of the form “ A and B , then C ” or similar, where the relative timing is restricted to the
299 interval in which plateau potentials remain active. The specific form of the expression is determined by the dendritic
300 tree itself, i.e. by the location of synapses and the strength of the coupling between the individual segments.

301 The “symbols” in these expressions are encoded into spike volleys and the timing of these spike volleys matters. Their
302 order is particularly important, because it determines whether a sequence of spike volleys can activate a neuron or
303 not. Therefore, neurons can process information in an event-based fashion where coherent spike events on a fast time
304 scale trigger interacting plateau events on a slower time scale. This mode of computation allows neurons to respond
305 much faster than a rate-code would permit, which is in line with empirical evidence on the level of single spikes in
306 somatosensory cortex [72].

307 Structured, symbolic representations of information have been explored extensively in cognitive psychology, cognitive
308 science, linguistics and artificial intelligence, but the most widely used analogy for neural computation are artificial
309 neural networks, which are rooted in a connectionist view of cognition [73]: Simple and homogeneous individual units
310 interact in a complex network to form distributed representations. Our perspective lies between these two extremes, and
311 may help to resolve the apparent disconnect [74]: We conjecture that much of the richness of neural computation is
312 derived from the structured internal memory processes of diverse and intricate neurons.

313 In our model, dendritic plateau computation allows neurons to detect specific, rare sequences of events and indicate
314 this detection with as little as a single spike. This can result in an extremely sparse and hence metabolically efficient
315 code, but it naturally comes at a price: if each pattern to be detected can last hundreds of milliseconds, then each neuron
316 can only reliably detect one of these every couple of hundred milliseconds. The independent response of neurons in
317 an ensemble can help in this regard. We suggest that the temporal coherence of spikes reflected in the magnitude of
318 spike volleys is a mechanism through which a graded response can be encoded, for example the uncertainty about
319 the occurrence of an input symbol A . As a result of the independent and stochastic synaptic transmission of spikes,
320 dendrite segments can respond to this input symbol with a probability proportional to the volley magnitude. This in
321 turn leads ensembles of neurons, which are sensitive to the same incoming patterns of spike volleys, to respond with
322 spike volleys of their own. We therefore expect such ensembles that decode and encode information in spike timing and
323 magnitude of spike volleys to be an integral building block in neural computation.

324 7 Discussion

325 Our qualitative model of dendritic plateau computation aims to explain and formalize the mounting biological evidence
326 of neural computation in active dendrites. It asserts, that dendrites are segmented into functional units, each of which
327 can generate and maintain a plateau potential when excited by a volley of spikes, and that the interaction of these plateau
328 states allows a single neuron to detect remarkably complex temporal patterns. Our work is closely related to recent work
329 by Hawkins and Ahmad [75], which proposes the use of active coincidence detection in dendrite segments to generate a
330 long UP state at the soma. Similarly, Brea et al. [76] present an elegant two compartment model and a corresponding
331 learning rule in which a basal dendrite segment learns to predict activation at the soma. In the “hierarchical temporal
332 memory” model of neural computation, longer temporal sequences are detected by laterally connected neurons of this
333 type [77].

334 We analyzed computation in single neurons, but what are the broader implications of this shift in perspective? Firstly, we
335 have not addressed plasticity and learning. In fact, the reliance of our model on long-lasting plateau potentials and the
336 ordering, rather than precise timing, of spike volleys poses a real challenge to most commonly used learning rules: On
337 the one hand, this complicates temporal credit assignment for training paradigms that rely on instantaneous error signals,
338 such as gradient backpropagation or related methods. On the other hand, the substantial, long-lasting depolarization of
339 the membrane potential by localized plateau potentials within the dendrite with an accompanying high Ca^{2+} was shown
340 to be the primary driver of synaptic plasticity [78, 79], which calls the role of backpropagating action potentials into
341 question. In our model, the plateaus precede any potential somatic spiking. Besides the magnitude of synaptic weights,
342 our model also makes extensive use of other properties of synapses, namely the transmission probability, the location of
343 the synapse within the dendrite, and the delay, which can affect the synchronization of spike volleys. This increases the
344 importance of structural plasticity [80], homeostatic processes that adjust synaptic transmission probabilities [81]
345 and recently proposed mechanisms for optimizing transmission delays through controlled (de-)myelination [82] and
346 opens the door for new learning rules inspired by these mechanisms.

347 Given a better understanding of plasticity and learning in dendritic plateau computation, we can approach the second
348 challenge: constructing and optimizing large and useful networks from such complex neurons. The central theoretical
349 questions are how the structured representation of information *inside* a neuron can be utilized in the context of a network,
350 and in turn, how networks can reliably produce the synchronous spike volleys that encode relevant information in their
351 magnitude, for example feature familiarity [83]. Recent advances in analysis techniques of brain data have shown
352 promising results in this direction, which may help to establish direct empirical evidence of the spike-volley based
353 representation of information we have proposed in this paper [84, 85]. On a more conceptual level, this event-based,
354 symbolic view of neural computation could help to substantially reduce the gap between neural networks and cognitive
355 architectures [74].

356 The prospect of energy efficient dendritic computation has also motivated research of potential implementations in
357 neuromorphic hardware. For example, Intel’s Loihi chip [86] and the DYNAPSE architecture [87] support some form
358 of active non-linear processing in functionally isolated dendrite segments. Our model provides a new perspective
359 on how these existing capabilities could be utilized for computation. But the simplicity of our proposed mechanism
360 also suggests novel hardware implementations that use complex dendrite structures, rather than complex ion-channel
361 dynamics or larger networks, to boost computational efficiency. We believe that this trade-off between structural and
362 dynamic complexity of neurons will remain a critical topic for further research.

363 8 Materials and Methods

364 8.1 Implementation of the navigation experiment

365 To simulate the stochastic movements of an animal in a two-dimensional environment, random paths are generated
366 with time-varying location $l(t) = (X(t), Y(t)) \in \mathbb{R}^2$ as solutions of the following system of stochastic differential
367 equations:

$$\begin{aligned}dX &= \cos(2\pi A)Vdt \\dY &= \sin(2\pi A)Vdt \\dA &= 0.25dW_A \\dV &= 10.0(0.25 - V)dt + 0.1dW_V\end{aligned}\tag{7}$$

368 A represents the angular heading of the animal, V represents its velocity in m s^{-1} and W_A, W_V represent independent
369 standard Brownian motion processes. Each path is generated with a randomized initial position within a rectangular

370 domain of $10\text{ cm} \times 9.5\text{ cm}$, a random angular heading and a random velocity according to the marginal stationary
371 distribution of V in the equation above, and is simulated for a fixed duration of 200 ms. Three populations of place cells,
372 each 20 neurons strong, are centered on a hexagonal grid with center-to-center distance of $r \approx 2.9\text{ cm}$. Each population
373 randomly emits spike volleys following a homogeneous Poisson process with rate $\lambda = 50\text{ Hz}$. The magnitude of each
374 spike volley is determined by the population's mean activity at the time, which depends on the animal's location within
375 the environment through a receptive field tuning curve. The tuning curves model the probability of each individual
376 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
377 coefficient $\sigma = 9.7\text{ mm}$, centered on the tiles of the hexagonal grid. The total number of spikes emitted during a
378 volley from population i at time t is therefore a random variable distributed according to a Binomial distribution with
379 population size $n = 20$ and probability $p = f_i(l(t))$. Additionally, each neuron in the population emits random spikes
380 at a rate of 5 Hz to emulate background activity. Each spike is transmitted through stochastic synapses independently
381 with probability 0.5.

382 Each of the simulated neuron's dendrite segments receives spiking input from the 20 neurons of one population and
383 requires either 8 or 4 coincident spikes to trigger a plateau potential. The three segments are connected in a chain that
384 requires sequential activation by spike volleys from the input populations in correct order to fire a spike. A random
385 path is considered to be accepted by the neuron, if the neuron responds with a spike at any point in time during the
386 corresponding simulation run.

387 To evaluate the rotation and location sensitivity of the neuron, we also generate straight paths with constant movement
388 speed $v = \frac{3r}{200\text{ ms}} \approx 43\text{ cm s}^{-1}$ that are either rotated around the center of the environment by an angle α or offset from
389 the center by a distance Δx orthogonal to the optimal movement direction. For each angle or offset, respectively, the
390 empirical firing probability of the neuron in response to that path is estimated by simulating the path and the neuron's
391 responses 500 times each.

392 8.2 Simulation framework for dendritic plateau computation

393 All simulations are implemented in a custom package developed in the Julia programming language [88], publicly
394 available via the code repository hosted at <https://github.com/jleugeri/DPC.jl>. The simulator implements the neuron
395 model outlined in this paper using a fast and extensible event-based formalism. All experiments and configuration files
396 can be found in the `examples` subfolder of the repository.

397 Further documentation of the simulator, its interfaces, and implementation details can be found there as well.

398 References

- 399 [1] Valentino Braitenberg and Almut Schüz. *Cortex: statistics and geometry of neuronal connectivity*. Springer
400 Science & Business Media, 2013.
- 401 [2] Eugene M Izhikevich. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572,
402 2003.
- 403 [3] Michael London and Michael Häusser. Dendritic computation. *Annu. Rev. Neurosci.*, 28:503–532, 2005.
- 404 [4] Hongbo Jia, Nathalie L Rochefort, Xiaowei Chen, and Arthur Konnerth. Dendritic organization of sensory input
405 to cortical neurons in vivo. *Nature*, 464(7293):1307–1312, April 2010.
- 406 [5] Ning-Long Xu, Mark T Harnett, Stephen R Williams, Daniel Huber, Daniel H O'Connor, Karel Svoboda, and
407 Jeffrey C Magee. Nonlinear dendritic integration of sensory and motor input during an active sensing task. *Nature*,
408 492(7428):247–251, December 2012.
- 409 [6] Naoya Takahashi, Thomas G Oertner, Peter Hegemann, and Matthew E Larkum. Active cortical dendrites
410 modulate perception. *Science*, 354(6319):1587–1590, 2016.
- 411 [7] Aaron Kerlin, Mohar Boaz, Daniel Flickinger, Bryan J MacLennan, Matthew B Dean, Courtney Davis, Nelson
412 Spruston, and Karel Svoboda. Functional clustering of dendritic activity during decision-making. *Elife*, 8:e46966,
413 2019.
- 414 [8] Nelson Spruston, Greg Stuart, and Michael Häusser. Principles of dendritic integration. *Dendrites*, 351(597):1,
415 2016.
- 416 [9] Srdjan D Antic, Wen-Liang Zhou, Anna R Moore, Shaina M Short, and Katerina D Ikonomu. The decade of the
417 dendritic NMDA spike. *J. Neurosci. Res.*, 88(14):2991–3001, November 2010.
- 418 [10] Andreas V M Herz, Tim Gollisch, Christian K Machens, and Dieter Jaeger. Modeling single-neuron dynamics
419 and computations: a balance of detail and abstraction. *Science*, 314(5796):80–85, October 2006.

- 420 [11] Balázs B Ujfalussy, Judit K Makara, Máté Lengyel, and Tiago Branco. Global and multiplexed dendritic
421 computations under in vivo-like conditions. *Neuron*, 100(3):579–592.e5, November 2018.
- 422 [12] Songting Li, Nan Liu, Xiaohui Zhang, David W McLaughlin, Douglas Zhou, and David Cai. Dendritic computa-
423 tions captured by an effective point neuron model. *Proc. Natl. Acad. Sci. U. S. A.*, 116(30):15244–15252, July
424 2019.
- 425 [13] Panayiota Poirazi, Terrence Brannon, and Bartlett W Mel. Pyramidal neuron as two-layer neural network. *Neuron*,
426 37(6):989–999, 2003.
- 427 [14] David Beniaguev, Idan Segev, and Michael London. Single cortical neurons as deep artificial neural networks.
428 *Cold Spring Harbor Laboratory*, page 613141, March 2020.
- 429 [15] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error
430 propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.
- 431 [16] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.
- 432 [17] Nikolaus Kriegeskorte. Deep neural networks: a new framework for modeling biological vision and brain
433 information processing. *Annual review of vision science*, 1:417–446, 2015.
- 434 [18] J O’Keefe and J Dostrovsky. The hippocampus as a spatial map. preliminary evidence from unit activity in the
435 freely-moving rat. *Brain Res.*, 34(1):171–175, November 1971.
- 436 [19] Torkel Hafting, Marianne Fyhn, Sturla Molden, May-Britt Moser, and Edvard I Moser. Microstructure of a spatial
437 map in the entorhinal cortex. *Nature*, 436(7052):801–806, August 2005.
- 438 [20] Martin Stemmler, Alexander Mathis, and Andreas V M Herz. Connecting multiple spatial scales to decode the
439 population activity of grid cells. *Sci Adv*, 1(11):e1500816, December 2015.
- 440 [21] Howard Eichenbaum. On the integration of space, time, and memory. *Neuron*, 95(5):1007–1018, August 2017.
- 441 [22] Brice Bathellier, Derek L Buhl, Riccardo Accolla, and Alan Carleton. Dynamic ensemble odor coding in the
442 mammalian olfactory bulb: sensory information at different timescales. *Neuron*, 57(4):586–598, February 2008.
- 443 [23] Bede M Broome, Vivek Jayaraman, and Gilles Laurent. Encoding and decoding of overlapping odor sequences.
444 *Neuron*, 51(4):467–482, August 2006.
- 445 [24] Huan Luo and David Poeppel. Phase patterns of neuronal responses reliably discriminate speech in human
446 auditory cortex. *Neuron*, 54(6):1001–1010, June 2007.
- 447 [25] Ofer Melamed, Wulfram Gerstner, Wolfgang Maass, Misha Tsodyks, and Henry Markram. Coding and learning
448 of behavioral sequences. *Trends Neurosci.*, 27(1):11–4; discussion 14–5, January 2004.
- 449 [26] C Beaulieu and M Colonnier. A laminar analysis of the number of round-asymmetrical and flat-symmetrical
450 synapses on spines, dendritic trunks, and cell bodies in area 17 of the cat. *J. Comp. Neurol.*, 231(2):180–189,
451 January 1985.
- 452 [27] J del Castillo and B Katz. Quantal components of the end-plate potential. *J. Physiol.*, 124(3):560–573, June 1954.
- 453 [28] C F Stevens. Quantal release of neurotransmitter and long-term potentiation. *Cell*, 72 Suppl:55–63, January 1993.
- 454 [29] Michael Hollmann and Stephen Heinemann. Cloned glutamate receptors. *Annual review of neuroscience*,
455 November 2003.
- 456 [30] JC Watkins and RH Evans. Excitatory amino acid transmitters. *Annual review of pharmacology and toxicology*,
457 21(1):165–204, 1981.
- 458 [31] H Monyer, N Burnashev, D J Laurie, B Sakmann, and P H Seeburg. Developmental and regional expression in the
459 rat brain and functional properties of four NMDA receptors. *Neuron*, 12(3):529–540, March 1994.
- 460 [32] T Götz, U Kraushaar, J Geiger, J Lübke, T Berger, and P Jonas. Functional properties of AMPA and NMDA
461 receptors expressed in identified types of basal ganglia neurons. *J. Neurosci.*, 17(1):204–215, January 1997.
- 462 [33] Attila Losonczy and Jeffrey C Magee. Integrative properties of radial oblique dendrites in hippocampal CA1
463 pyramidal neurons. *Neuron*, 50(2):291–307, April 2006.
- 464 [34] Sonia Gasparini, Michele Migliore, and Jeffrey C Magee. On the initiation and propagation of dendritic spikes in
465 CA1 pyramidal neurons. *J. Neurosci.*, 24(49):11046–11056, December 2004.
- 466 [35] Sonia Gasparini and Jeffrey C Magee. State-dependent dendritic computation in hippocampal CA1 pyramidal
467 neurons. *J. Neurosci.*, 26(7):2088–2100, February 2006.
- 468 [36] Jacopo Bono and Claudia Clopath. Modeling somatic and dendritic spike mediated plasticity at the single neuron
469 and network level. *Nat. Commun.*, 8(1):706, September 2017.

- 470 [37] Paul Rhodes. The properties and implications of NMDA spikes in neocortical pyramidal cells. *J. Neurosci.*, 26
471 (25):6704–6715, June 2006.
- 472 [38] Nelson Spruston. Pyramidal neurons: dendritic structure and synaptic integration. *Nat. Rev. Neurosci.*, 9(3):
473 206–221, March 2008.
- 474 [39] Guy Major, Matthew E Larkum, and Jackie Schiller. Active properties of neocortical pyramidal neuron dendrites.
475 *Annu. Rev. Neurosci.*, 36:1–24, July 2013.
- 476 [40] Katerina D Oikonomou, Mandakini B Singh, Enas V Sterjanaj, and Srdjan D Antic. Spiny neurons of amygdala,
477 striatum, and cortex use dendritic plateau potentials to detect network UP states. *Front. Cell. Neurosci.*, 8:292,
478 September 2014.
- 479 [41] R Angus Silver, Andrew F MacAskill, and Mark Farrant. Neurotransmitter-gated ion channels in dendrites.
480 *Dendrites*, 3rd edn. Oxford University Press, New York, pages 217–257, 2016.
- 481 [42] Steven S Goldstein and Wilfrid Rall. Changes of action potential shape and velocity for changing core conductor
482 geometry. *Biophysical journal*, 14(10):731–757, 1974.
- 483 [43] W Rall. Electrophysiology of a dendritic neuron model. *Biophys. J.*, 2(2 Pt 2):145–167, March 1962.
- 484 [44] A N Burkitt. A review of the integrate-and-fire neuron model: I. homogeneous synaptic input. *Biol. Cybern.*, 95
485 (1):1–19, July 2006.
- 486 [45] Greg Stuart and Nelson Spruston. Determinants of voltage attenuation in neocortical pyramidal neuron dendrites.
487 *Journal of Neuroscience*, 18(10):3501–3510, 1998.
- 488 [46] Michael Häusser. Synaptic function: dendritic democracy. *Current Biology*, 11(1):R10–R12, 2001.
- 489 [47] Jeffrey C Magee and Erik P Cook. Somatic epsp amplitude is independent of synapse location in hippocampal
490 pyramidal neurons. *Nature neuroscience*, 3(9):895–903, 2000.
- 491 [48] Tim Jarsky, Alex Roxin, William L Kath, and Nelson Spruston. Conditional dendritic spike propagation following
492 distal synaptic activation of hippocampal CA1 pyramidal neurons. *Nat. Neurosci.*, 8(12):1667–1676, December
493 2005.
- 494 [49] Christine Grienberger, Xiaowei Chen, and Arthur Konnerth. Nmda receptor-dependent multidendrite ca²⁺ spikes
495 required for hippocampal burst firing in vivo. *Neuron*, 81(6):1274–1281, 2014.
- 496 [50] Matthew E Larkum, Thomas Nevian, Maya Sandler, Alon Polsky, and Jackie Schiller. Synaptic integration in tuft
497 dendrites of layer 5 pyramidal neurons: a new unifying principle. *Science*, 325(5941):756–760, 2009.
- 498 [51] Guy Major, Alon Polsky, Winfried Denk, Jackie Schiller, and David W Tank. Spatiotemporally graded NMDA
499 spike/plateau potentials in basal dendrites of neocortical pyramidal neurons. *J. Neurophysiol.*, 99(5):2584–2601,
500 May 2008.
- 501 [52] C Koch, T Poggio, and V Torre. Retinal ganglion cells: a functional interpretation of dendritic morphology. *Philos.*
502 *Trans. R. Soc. Lond. B Biol. Sci.*, 298(1090):227–263, July 1982.
- 503 [53] Alon Polsky, Bartlett W Mel, and Jackie Schiller. Computational subunits in thin dendrites of pyramidal cells.
504 *Nat. Neurosci.*, 7(6):621–627, June 2004.
- 505 [54] Yunliang Zang, Stéphane Dieudonné, and Erik De Schutter. Voltage- and Branch-Specific climbing fiber responses
506 in purkinje cells. *Cell Rep.*, 24(6):1536–1549, August 2018.
- 507 [55] CF Ekerot and O Oscarsson. Prolonged depolarization elicited in purkinje cell dendrites by climbing fibre impulses
508 in the cat. *The Journal of physiology*, 318(1):207–221, 1981.
- 509 [56] Sigita Augustinaite, Bernd Kuhn, Paul Johannes Helm, and Paul Heggelund. NMDA spike/plateau potentials in
510 dendrites of thalamocortical neurons. *J. Neurosci.*, 34(33):10892–10905, August 2014.
- 511 [57] Tiago Branco and Michael Häusser. The single dendritic branch as a fundamental functional unit in the nervous
512 system. *Curr. Opin. Neurobiol.*, 20(4):494–502, August 2010.
- 513 [58] Willem A M Wybo, Benjamin Torben-Nielsen, Thomas Nevian, and Marc-Oliver Gewaltig. Electrical compart-
514 mentalization in neurons. *Cell Rep.*, 26(7):1759–1773.e7, February 2019.
- 515 [59] Matthew E Larkum and Thomas Nevian. Synaptic clustering by dendritic signalling mechanisms. *Curr. Opin.*
516 *Neurobiol.*, 18(3):321–331, June 2008.
- 517 [60] Naoya Takahashi, Kazuo Kitamura, Naoki Matsuo, Mark Mayford, Masanobu Kano, Norio Matsuki, and Yuji
518 Ikegaya. Locally synchronized synaptic inputs. *Science*, 335(6066):353–356, January 2012.
- 519 [61] Tiago Branco, Kevin Staras, Kevin J Darcy, and Yukiko Goda. Local dendritic activity sets release probability at
520 hippocampal synapses. *Neuron*, 59(3):475–485, August 2008.

- 521 [62] Jiang Hao, Xu-Dong Wang, Yang Dan, Mu-Ming Poo, and Xiao-Hui Zhang. An arithmetic rule for spatial
522 summation of excitatory and inhibitory inputs in pyramidal neurons. *Proc. Natl. Acad. Sci. U. S. A.*, 106(51):
523 21906–21911, December 2009.
- 524 [63] Michael Doron, Giuseppe Chindemi, Eilif Muller, Henry Markram, and Idan Segev. Timed synaptic inhibition
525 shapes NMDA spikes, influencing local dendritic processing and global I/O properties of cortical neurons. *Cell*
526 *Rep.*, 21(6):1550–1561, November 2017.
- 527 [64] K Du, Y W Wu, R Lindroos, Y Liu, and others. Cell-type-specific inhibition of the dendritic plateau potential in
528 striatal spiny projection neurons. *Proceedings of the*, 2017.
- 529 [65] Albert Gidon and Idan Segev. Principles governing the operation of synaptic inhibition in dendrites. *Neuron*, 75
530 (2):330–341, July 2012.
- 531 [66] William Muñoz, Robin Tremblay, Daniel Levenstein, and Bernardo Rudy. Layer-specific modulation of neocortical
532 dendritic inhibition during active wakefulness. *Science*, 355(6328):954–959, March 2017.
- 533 [67] Tiago Branco, Beverley A Clark, and Michael Häusser. Dendritic discrimination of temporal input sequences in
534 cortical neurons. *Science*, 329(5999):1671–1675, 2010.
- 535 [68] Jason J Moore, Pascal M Ravassard, David Ho, Lavanya Acharya, Ashley L Kees, Cliff Vuong, and Mayank R
536 Mehta. Dynamics of cortical dendritic membrane potential and spikes in freely behaving rats. *Science*, 355(6331),
537 March 2017.
- 538 [69] Yulia Sandamirskaya and Gregor Schöner. An embodied account of serial order: how instabilities drive sequence
539 generation. *Neural Netw.*, 23(10):1164–1179, December 2010.
- 540 [70] Gianluigi Mongillo, Omri Barak, and Misha Tsodyks. Synaptic theory of working memory. *Science*, 319(5869):
541 1543–1546, March 2008.
- 542 [71] Wolfgang Maass, Thomas Natschläger, and Henry Markram. Fading memory and kernel properties of generic
543 cortical microcircuit models. *Journal of Physiology-Paris*, 98(4):315–330, July 2004.
- 544 [72] Rufin VanRullen, Rudy Guyonneau, and Simon J Thorpe. Spike times make sense. *Trends Neurosci.*, 28(1):1–4,
545 January 2005.
- 546 [73] Paul Smolensky. On the proper treatment of connectionism. *Behav. Brain Sci.*, 11(1):1–23, March 1988.
- 547 [74] J A Fodor and Z W Pylyshyn. Connectionism and cognitive architecture: a critical analysis. *Cognition*, 28(1-2):
548 3–71, March 1988.
- 549 [75] Jeff Hawkins and Subutai Ahmad. Why neurons have thousands of synapses, a theory of sequence memory in
550 neocortex. *Frontiers in neural circuits*, 10:23, 2016.
- 551 [76] Johanni Brea, Alexis Tamás Gaál, Robert Urbanczik, and Walter Senn. Prospective coding by spiking neurons.
552 *PLOS Computational Biology*, 12(6):1–25, 06 2016. doi: 10.1371/journal.pcbi.1005003. URL <https://doi.org/10.1371/journal.pcbi.1005003>.
- 554 [77] Dileep George and Jeff Hawkins. Towards a mathematical theory of cortical micro-circuits. *PLoS Comput. Biol.*,
555 5(10):e1000532, October 2009.
- 556 [78] John Lisman and Nelson Spruston. Postsynaptic depolarization requirements for LTP and LTD: a critique of spike
557 timing-dependent plasticity. *Nat. Neurosci.*, 8(7):839–841, July 2005.
- 558 [79] Jason Hardie and Nelson Spruston. Synaptic depolarization is more effective than back-propagating action
559 potentials during induction of associative long-term potentiation in hippocampal pyramidal neurons. *J. Neurosci.*,
560 29(10):3233–3241, March 2009.
- 561 [80] Andreas Knoblauch and Friedrich T Sommer. Structural plasticity, effectual connectivity, and memory in cortex.
562 *Front. Neuroanat.*, 10:63, June 2016.
- 563 [81] Gina Turrigiano. Homeostatic synaptic plasticity: local and global mechanisms for stabilizing neuronal function.
564 *Cold Spring Harbor perspectives in biology*, 4(1):a005736, 2012.
- 565 [82] R Douglas Fields. A new mechanism of nervous system plasticity: activity-dependent myelination. *Nat. Rev.*
566 *Neurosci.*, 16(12):756–767, December 2015.
- 567 [83] Clemens Korndörfer, Ekkehard Ullner, Jordi García-Ojalvo, and Gordon Pipa. Cortical spike synchrony as a
568 measure of input familiarity. *Neural computation*, 29(9):2491–2510, 2017.
- 569 [84] Rory G Townsend and Pulin Gong. Detection and analysis of spatiotemporal patterns in brain activity. *PLoS*
570 *Comput. Biol.*, 14(12):e1006643, December 2018.

- 571 [85] Min Song, Minseok Kang, Hyeonsu Lee, Yong Jeong, and Se-Bum Paik. Classification of spatiotemporal neural
572 activity patterns in brain imaging data. *Sci. Rep.*, 8(1):8231, May 2018.
- 573 [86] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios
574 Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, Yuyun Liao, Chit-Kwan Lin, Andrew Lines, Ruokun Liu, Deepak
575 Mathaikutty, Steven McCoy, Arnab Paul, Jonathan Tse, Guruguhanathan Venkataramanan, Yi-Hsin Weng, Andreas
576 Wild, Yoonseok Yang, and Hong Wang. Loihi: A neuromorphic manycore processor with On-Chip learning. *IEEE*
577 *Micro*, 38(1):82–99, January 2018.
- 578 [87] Saber Moradi, Ning Qiao, Fabio Stefanini, and Giacomo Indiveri. A scalable multicore architecture with
579 heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs). *IEEE Trans.*
580 *Biomed. Circuits Syst.*, 12(1):106–122, February 2018.
- 581 [88] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing.
582 *SIAM review*, 59(1):65–98, 2017.

B.2 CONFERENCE ABSTRACT: ACTIVE DENDRITES IMPLEMENT TEMPORAL LOGIC GATES

Pascal Nieters, Johannes Leugering and Gordon Pipa. 'Active dendrites implement probabilistic temporal logic gates.' In: *Proc. of the Computational Cognition Workshop Osnabrück*. 2019

Abstract

The recognition of patterns is a primary function of the brain. Often, relevant patterns occur on time-scales that vary in length. For example, place and grid cells encode location and therefore activate in sequential patterns that depend on an animal's varying movement speeds. Encoded in these patterns are directed paths through an environment that are important to – for example – find the way back out of a maze. Further patterns on time-scales on the order of hundreds of milliseconds have been observed elsewhere in cortex, for example in the sequential codes for different odors in the olfactory bulb, and the segmentation length of speech signals in auditory cortex. Neuron models used in the theoretical analysis of cortical circuits predominantly model the passive integration of synaptic currents, and are therefore limited to the short and fixed time-scale determined by the passive electrical properties of neurons. Since these integrate-and-fire neurons cannot inherently solve the problem of long, temporal integration, the implementation of the required working-memory is relegated to the dynamics of attractor networks and synapses undergoing fast, plastic changes. Here, we argue that intricate biophysical mechanisms along dendrites of pyramidal neurons, that have recently been detailed in experiments, implement an elegant alternative solution we call active dendritic sequence processing (ADSP).

Active dendrites implement probabilistic temporal logic gates.

Pascal Nieters
University Osnabrück

Johannes Leugering
Fraunhofer IIS

Gordon Pipa
University Osnabrück

The recognition of patterns is a primary function of the brain. Often, relevant patterns occur on time-scales that vary in length. For example, place and grid cells encode location [7] and therefore activate in sequential patterns that depend on an animal's varying movement speeds. Encoded in these patterns are directed paths through an environment that are important to – for example – find the way back out of a maze. Further patterns on time-scales on the order of hundreds of milliseconds have been observed elsewhere in cortex, for example in the sequential codes for different odors in the olfactory bulb [1], and the segmentation length of speech signals in auditory cortex [2].

Neuron models used in the theoretical analysis of cortical circuits predominantly model the passive integration of synaptic currents, and are therefore limited to the short and fixed time-scale determined by the passive electrical properties of neurons. Since these integrate-and-fire neurons cannot inherently solve the problem of long, temporal integration, the implementation of the required working-memory is relegated to the dynamics of attractor networks [5] and synapses undergoing fast, plastic changes [4]. Here, we argue that intricate biophysical mechanisms along dendrites of pyramidal neurons, that have recently been detailed in experiments [3], implement an elegant alternative solution we call *active dendritic sequence processing* (ADSP).

ADSP relies on the dynamics of NMDA α ion channels that activate when multiple spikes arrive at a local cluster of synapses in a short period of time. Their opening due to the availability of Glutamate and high enough postsynaptic depolarization induced by overlapping EPSPs leads to supra linear *dendritic plateau potentials* that can last for up to hundreds of milliseconds. However, they do not actively propagate along the dendrite and are subject to the strong attenuation of the dendritic cable as well as functional dendritic compartmentalization [6].

We call clusters of synapses that can initiate a plateau a dendritic segment that functions as a coincidence detector and only weakly interacts with neighboring dendritic segments. When one segment initiates a plateau potential in response to strong, coincident input, neighboring segments increase their resting voltage for the duration of the plateau, enabling them to initiate a plateau themselves. A cascade of overlapping plateau potentials can start in distal segments and propagates towards the soma. A successful cascade leads to a neuronal UP-state that signifies the recognition of a sequential pattern of coincident spike inputs on a time-scale of the plateau-length times the number of dendritic segments in the cascade.

In the tradition of McCulloch & Pitts, we abstractly model this dendritic computational behavior as temporal logic gates in a tree structure, with a second time-scale that enables local memory for asynchronous computation. To reflect richer computational functions, AND and OR - type gates can be implemented by requiring at least N child segments to have initiated a plateau for the current node to be active and be able to initiate its own plateau. This deterministic model of neural computation extends naturally to stochastic synapses as the probability of a neuronal UP-state is directly propor-

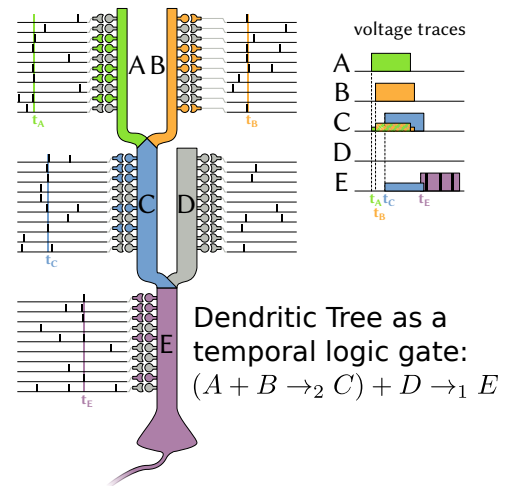


Figure 1: Plateaus in response to spike volleys interact in the morphology of dendritic segments. The function of this dendritic tree can be expressed as a temporal logic gate.

tional to the number of coincident spikes at individual segments. In a population ensemble of neurons with the same input connections and independent synapses, the number of coincident output spikes encodes this signal strength.

We show that ADSP can solve the temporal integration problem on long and varying time-scales with the example of path integration from place cell activity. The temporal logic gate model directly maps dendritic morphology to computational function. Cells other than pyramidal neurons may implement the same general computational principle by other biomechanistic implementations. We connect the sub-symbolic and dynamic processes of dendritic integration to logical functions on symbols defined by coincident spike events, addressing a long standing problem of connectionist models to explain and account for the inherent structure of reasoning.

REFERENCES

- [1] B. Bathellier, D. L. Buhl, R. Accolla, and A. Carleton. Dynamic ensemble odor coding in the mammalian olfactory bulb: sensory information at different timescales. *Neuron*, 57(4):586–598, Feb. 2008.
- [2] H. Luo and D. Poeppel. Phase patterns of neuronal responses reliably discriminate speech in human auditory cortex. *Neuron*, 54(6):1001–1010, June 2007.
- [3] G. Major, M. E. Larkum, and J. Schiller. Active properties of neocortical pyramidal neuron dendrites. *Annu. Rev. Neurosci.*, 36:1–24, July 2013.
- [4] G. Mongillo, O. Barak, and M. Tsodyks. Synaptic theory of working memory. *Science*, 319(5869):1543–1546, Mar. 2008.
- [5] A. Seeholzer, M. Deger, and W. Gerstner. Stability of working memory in continuous attractor networks under the control of short-term plasticity. *PLoS computational biology*, 15(4):e1006928, 2019.
- [6] N. Spruston. Pyramidal neurons: dendritic structure and synaptic integration. *Nat. Rev. Neurosci.*, 9(3):206–221, Mar. 2008.
- [7] M. Stemmler, A. Mathis, and A. V. M. Herz. Connecting multiple spatial scales to decode the population activity of grid cells. *Sci Adv*, 1(11):e1500816, Dec. 2015.

B.3 PATENT: NEUROMORPHIC PATTERN DETECTOR AND NEUROMORPHIC CIRCUITRY

Johannes Leugering, Pascal Nieters and Gordon Pipa. 'Neuromorphic Pattern Detector and Neuromorphic Circuitry Herewith'. Pat. DE:102019134044:A1. June 2021

English Abstract

The present invention relates to a neuromorphic pattern detector (2), which is designed to receive at least two 1-bit input signals (E1-EN) of a pattern to be recognized, with at least two comparison circuits (3), which are each designed to one of the 1- Bit input signals (E1-EN) to receive the number of "high" states or the "low" states of the respective 1-bit input signal (E1-EN) within a predetermined period of time, the number of counted states with a to compare the predetermined threshold value of the respective comparison circuit (3) and to indicate that the pattern to be recognized has been recognized when the threshold value is exceeded.



(12) **Offenlegungsschrift**

(21) Aktenzeichen: **10 2019 134 044.6**

(22) Anmeldetag: **11.12.2019**

(43) Offenlegungstag: **17.06.2021**

(51) Int Cl.: **G06N 3/063 (2006.01)**

(71) Anmelder:
Universität Osnabrück, 49074 Osnabrück, DE

(74) Vertreter:
Holz, Christian, Dipl.-Ing. Dr.-Ing., 30159 Hannover, DE

(72) Erfinder:
Leugering, Johannes, 49076 Osnabrück, DE;
Nieters, Pascal, 49124 Georgsmarienhütte, DE;
Pipa, Gordon, Prof. Dr., 49205 Hasbergen, DE

(56) Ermittelter Stand der Technik:
US 2016 / 0 292 569 A1

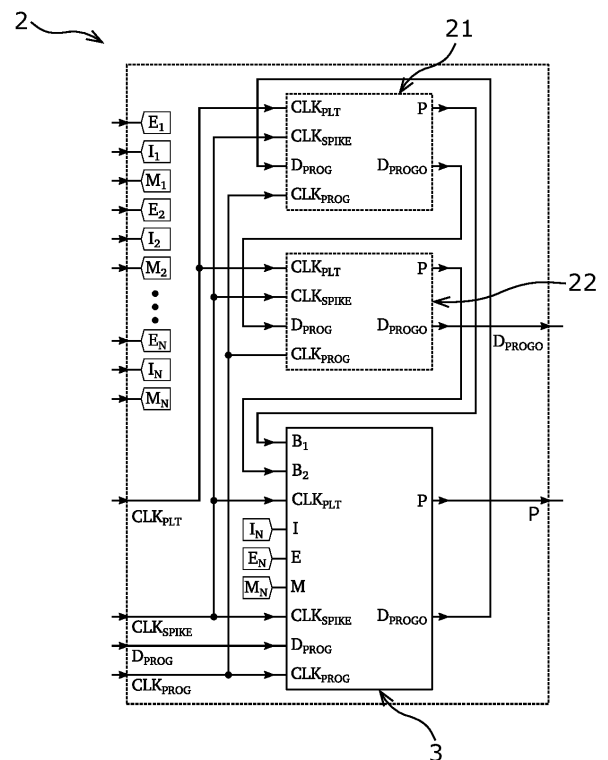
Leugering, Johannes; Nieters, Pascal; Pipa, Gordon: Event-based pattern detection in active dendrites. In: bioRxiv, 02.07.2019, 1-13. <https://www.biorxiv.org/content/early/2019/07/02/690792.full.pdf> [abgerufen am 11.12.2020]

Prüfungsantrag gemäß § 44 PatG ist gestellt.

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen.

(54) Bezeichnung: **Neuromorphen Musterdetektor und neuromorphe Schaltkreisordnung hiermit**

(57) 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.



Beschreibung

[0001] Die vorliegende Erfindung betrifft einen neuromorphen Musterdetektor gemäß dem Patentanspruch 1 sowie eine neuromorphe Schaltkreisanordnung gemäß dem Patentanspruch 13.

[0002] 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.

[0003] 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.

[0004] Zur Verarbeitung digitaler sowie analoger Informationen in Form digitaler Signale können auch sog. künstliche neuronale Netze bzw. Netzwerke ver-

wendet 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.

[0005] 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.

[0006] 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.

[0007] 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.

[0008] 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.

[0009] 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.

[0010] 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.

[0011] 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.

[0012] 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.

[0013] 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 geschlossen, 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.

[0014] Auf diese Art und Weise kann erfindungsgemäß vergleichsweise einfach mittels einer neuromorphen Schaltung eine Mustererkennung in einem digitalen Signal erfolgen.

[0015] 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.

[0016] 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.

[0017] 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.

[0018] 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.

[0019] 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.

[0020] 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.

[0021] 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.

[0022] 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.

[0023] Entsprechend können die Eigenschaften und Vorteile derartiger hierarchischer Datenstrukturen auf die erfindungsgemäße neuromorphe Schaltung übertragen und dort genutzt werden.

[0024] 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 Skalierbarkeit ermöglichen.

[0025] 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.

[0026] 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 ermög-

lichen, 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.

[0027] 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.

[0028] 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.

[0029] 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.

[0030] 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 Mustererken-

nung der Vergleichsschaltungen und damit auch des neuromorphen Musterdetektors in Abhängigkeit von wenigstens zwei unterschiedlichen Taktsignalen erfolgen.

[0031] 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.

[0032] 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.

[0033] 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.

[0034] 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.

[0035] 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.

[0036] 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.

[0037] 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-Ausgangssignaler neuromorphen Schaltkreisanordnung zusammenzuführen. Auf diese Art und Weise kann ein einziger resultierenden 1-Bit Datenstrom als Ausgangssignal der neuromorphen Schaltkreisanordnung erzeugt werden.

[0038] 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**.

[0039] **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**.

[0040] 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 **Fig. 2**.

[0041] Die Population **1** besteht dabei gemäß dem dargestellten Ausführungsbeispiel aus einer Anzahl **K** von Neuronen **2**, von welchen in der **Fig. 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 **E₁-E_N** besteht. Das Eingangssignal **E** enthält ein zu erkennendes Muster, welches auch als Pattern bezeichnet werden kann.

[0042] Jedes Neuron **2** erhält ferner parallel dasselbe Steuersignal **I**, welches aus einer Mehrzahl von einzelnen 1-Bit Steuersignalen **I₁-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.

[0043] 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 **E₁-E_N** und für jedes der **K** Neuronen **2** ein zufälliges 1-Bit Zufallsignal **M_{1,1}-M_{K,N}** zur Verfügung gestellt.

[0044] Des Weiteren erhält jedes Neuron **2** drei unterschiedliche Taktsignale CLK_{PLT} , $\text{CLK}_{\text{SPIKE}}$ 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 $\text{CLK}_{\text{SPIKE}}$ ist ein Taktsignal zur Steuerung 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.

[0045] 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**.

[0046] Jedes der K Neuronen **2** erzeugt ein binäres Ausgangssignal $\text{P}_1\text{-P}_K$, welche parallel einem Zeitmultiplexer **5** als dessen Eingangssignale $\text{S}_1\text{-S}_K$ zugeführt werden. Die Verarbeitung dieser Eingangssignale $\text{S}_1\text{-S}_K$ zu einem Ausgangssignal **O** des Zeitmultiplexers **5**, welches auch das Ausgangssignal **O** der gesamten Population **1** darstellt, wird weiter unten beschrieben.

[0047] 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**.

[0048] 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 Segment **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 ande-

ren Worten wird jeder Binärbaumzweig **21**, **22** entweder aus einem weiteren Neuron **2**, welches seinerseits wieder zwei Binärbaumzweige **21**, **22** aufweist, oder aus einem Abschlusszweig **20** gebildet.

[0049] 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 $\text{CLK}_{\text{SPIKE}}$ erhält. Die Binärbaumzweige **21**, **22** erhalten das Taktsignal CLK_{PLT} . Dies führt dazu, dass das Ausgangssignal $\text{P}_1\text{-P}_K$ des Neurons **2** als kurze Spikes mit dem Taktsignal $\text{CLK}_{\text{SPIKE}}$ und nicht lange Plateaus mit dem Taktsignal CLK_{PLT} aufweist.

[0050] Das Konfigurationssignal D_{PROG} des Neurons **2** wird jedem Binärbaumzweig **21**, **22**, jedem Abschlusszweig **20** sowie jedem Segment **3** zugeführt.

[0051] Jedes Segment **3** jeder Ebene des binären Baums erhält eines der 1-Bit Eingangssignale $\text{E}_1\text{-E}_N$ sowie das entsprechende 1-Bit Steuersignale $\text{I}_1\text{-I}_N$ und das entsprechende Zufallszahlensignal $\text{M}_{1,1}\text{-M}_{K,N}$. Die Funktion des Segments **3** wird weiter unten erklärt werden. Auch erhält jedes Segment **3** jeder Ebene des binären Baums die Taktsignale CLK_{PLT} , $\text{CLK}_{\text{SPIKE}}$ und CLK_{PROG} zu den zugehörigen Signalen.

[0052] 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 $\text{E}_1\text{-E}_N$ sowie das entsprechende 1-Bit Steuersignale $\text{I}_1\text{-I}_N$ und das entsprechende Zufallszahlensignal $\text{M}_{1,1}\text{-M}_{K,N}$.

[0053] 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.

[0054] 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 Eingangssignale 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 Ausgangs-

signal 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.

[0055] 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_{PROG} 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_{PROG} des Segments **3**.

[0056] Der Schiebefensterdetektor **4** erzeugt parallel zu dem Konfigurationssignals D_{PROG} 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**.

[0057] 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 zweiten 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 zweiten N-bit SIPO Schieberegisters **37** auf „low“, d.h. in den „low“-Zustand.

[0058] 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.

[0059] 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.

[0060] 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 linkschiebende 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**.

[0061] 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.

[0062] 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.

[0063] Ein $M+1$ -fach 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 $M+1$ -fach Multiplexers **43** und des gesamten Schiebefensterdetektors **4**.

[0064] Welches der $M+1 \leq 2^K$ (M plus 1 kleiner gleich 2 hoch K) Eingangssignale mittels des $M+1$ -fach 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.

[0065] 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.

[0066] 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**.

[0067] 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.

[0068] 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**.

[0069] 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.

[0070] 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} .

[0071] In der Verarbeitung des Eingangssignals **E**, des Kontrollsignals **I** sowie der binären Zufallszahlensignal **M** zur Mustererkennung gilt insbesondere, dass das Zeitfenster des Schiebefensterdetektors **4** $N * 1/f(CLK_{\text{SPIKE}})$ 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.

[0072] Das Taktsignal CLK_{PLT} steuert ausschließlich die Länge der Plateaus in den einzelnen Segmenten **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 $N * 1/f(CLK_{\text{PLT}})$ und nach maximal $(N+1) * 1/f(CLK_{\text{PLT}})$ 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.

[0073] 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 Im-

plementierung von Algorithmen zur Mustererkennung verwendet.

[0074] 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.

[0075] 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“.

[0076] 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.

[0077] 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**.

[0078] 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 Zufallsignalen **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.

[0079] 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.

[0080] 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 des 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**.

[0081] Jedes der Segmente **3** wird zunächst durch die Eingangssignale B_1, B_2 anderer im binären Baum untergeordneter Segmente **3** getrieben, sofern die-

se 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.

[0082] 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.

[0083] 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.

[0084] 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-Signa-

le in dem durch das Taktsignal CLK_{SPIKE} und durch die Länge der Delayline festgelegten Zeitfenster im Datenstrom D_{IN} zu finden waren.

[0085] 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.

[0086] Das Konfigurationssignal D_{PROG} , welches sich durch alle Bauteile der Population **1** zieht, ermöglicht die Konfiguration der Population **1** und aller enthaltener Bauteile.

[0087] 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**.

[0088] 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.

[0089] 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.

[0090] 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.

[0091] 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.

[0092] 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.

[0093] 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 Taktsignale **CLK_{SPIKE}**, **CLK_{PLT}** kann ein Segment **3** auf die speziellen externen Timing-Anforderungen der Anwendung angepasst werden.

[0094] 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.

[0095] 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

B₁	Eingangssignal eines Segments 3 seitens eines ersten Binärbaumzweigs 21
B₂	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, E₁-E_N	Eingangssignal der Population 1 ; eingehender Datenstrom
i	Zählindex
I, I₁-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
P₁-P_k	1-Bit Ausgangssignale der Neuronen 2 , der Abschlusszweige 20 , der Binärbaumzweige 21 , 22 und der Segmente 3
S₁-S_K	Eingangssignale des Zeitmultiplexers 5
SL	linksschiebendes Eingangssignal des zweiten bidirektionalen (M-bit SI-PO-) Schieberegisters 42 des Schiebefensterdetektors 4
SR	rechtsschiebendes Eingangssignal des zweiten bidirektionalen (M-bit SI-PO-) Schieberegisters 42 des Schiebefensterdetektors 4
W	Grad der Übereinstimmung zwischen zugeführtem Muster und konfigurierten vorbestimmten Muster

1	neuromorphe Schaltkreisanordnung; Population	52	selbst initialisierter (K-bit) Ringzähler
2	neuromorpher Musterdetektor; Neuron; Neuronen-Schaltkreis; Binärbaumwurzel	53 54	Oder-Gatter D-Flipflop

Patentansprüche

20	Abschlusszweig; Binärbaumblatt; Terminal Branch	1. Neuromorpher Musterdetektor (2), welcher ausgebildet ist, wenigstens zwei 1-Bit Eingangssignale (E_1 - E_N) eines zu erkennenden Musters zu erhalten,
21	erster Binärbaumzweig; erster innerer Knoten des Binärbaums; erster Nested Branch	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.
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	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.
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	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.
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	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
50	Flipflops	
51	Und-Gatter	

te 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.

Es folgen 7 Seiten Zeichnungen

Anhängende Zeichnungen

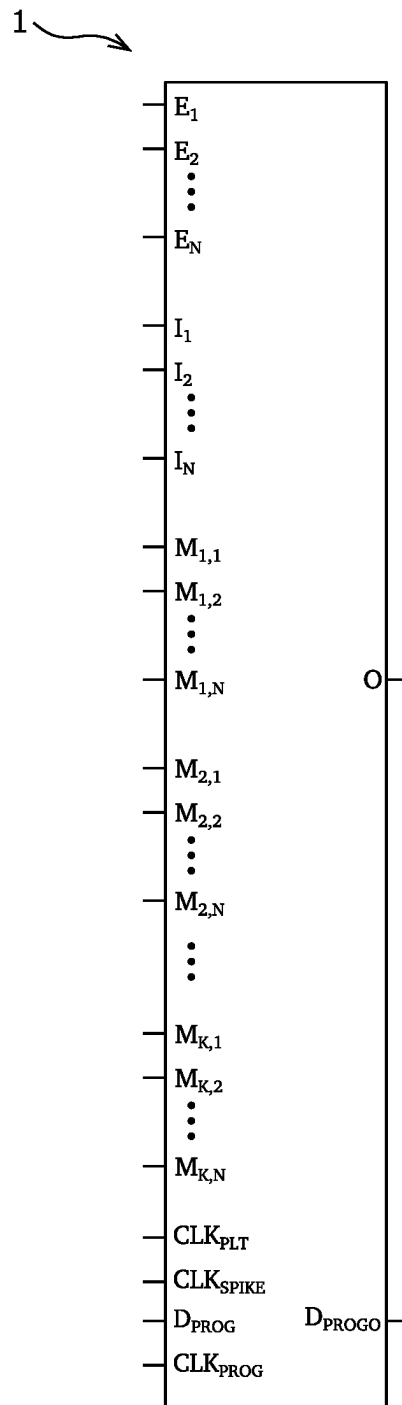


FIG. 1

1 ↗

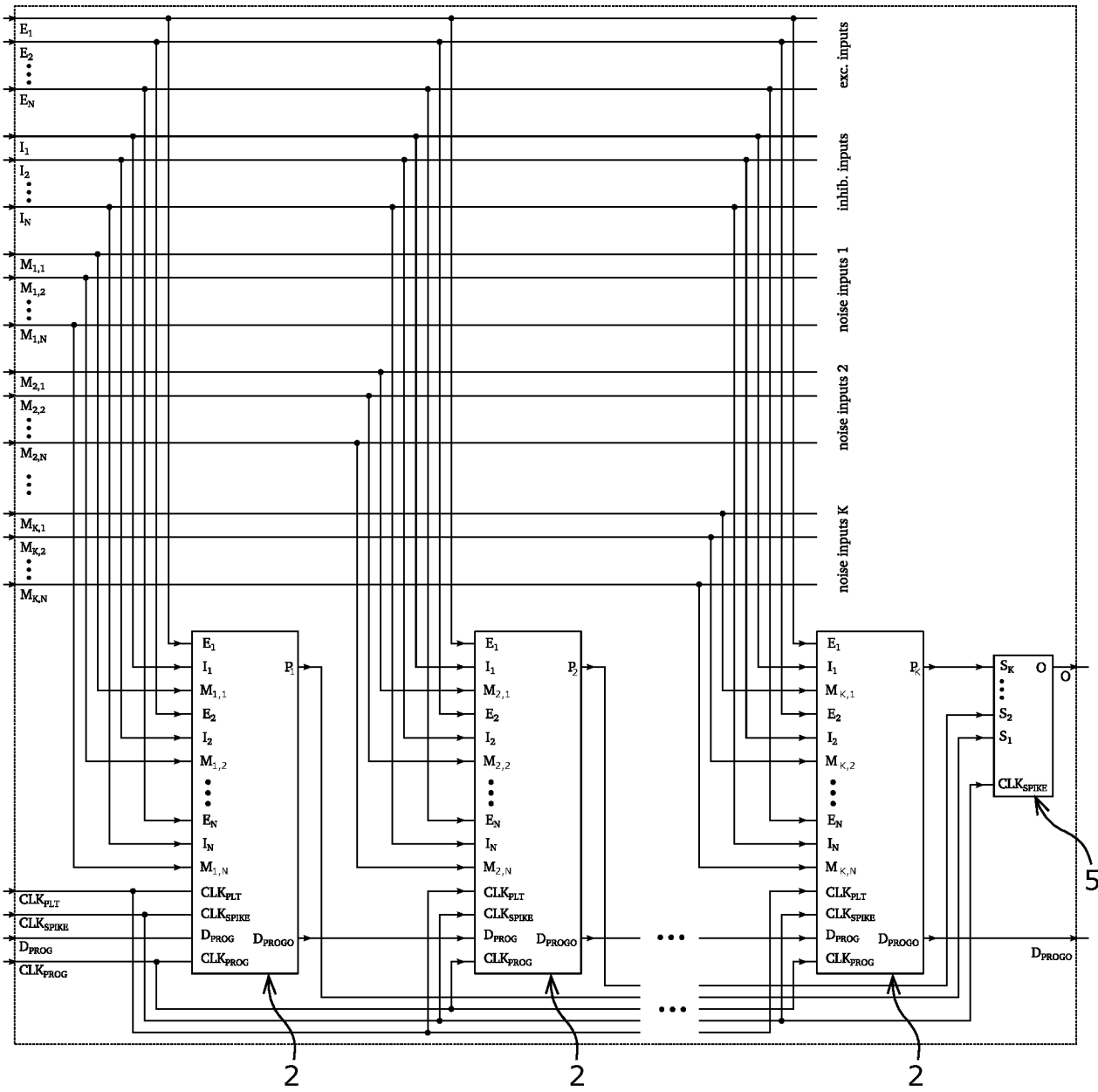


FIG. 2

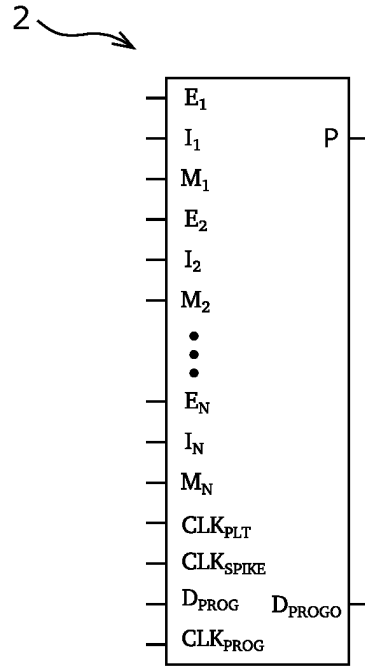


FIG. 3

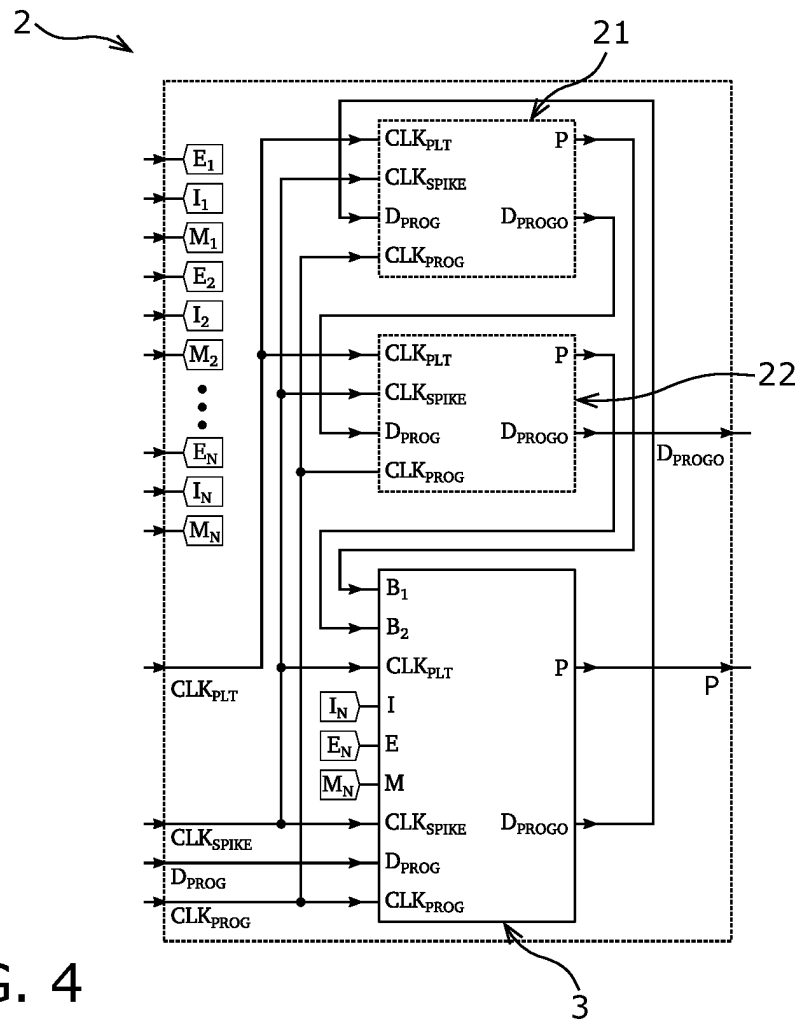


FIG. 4

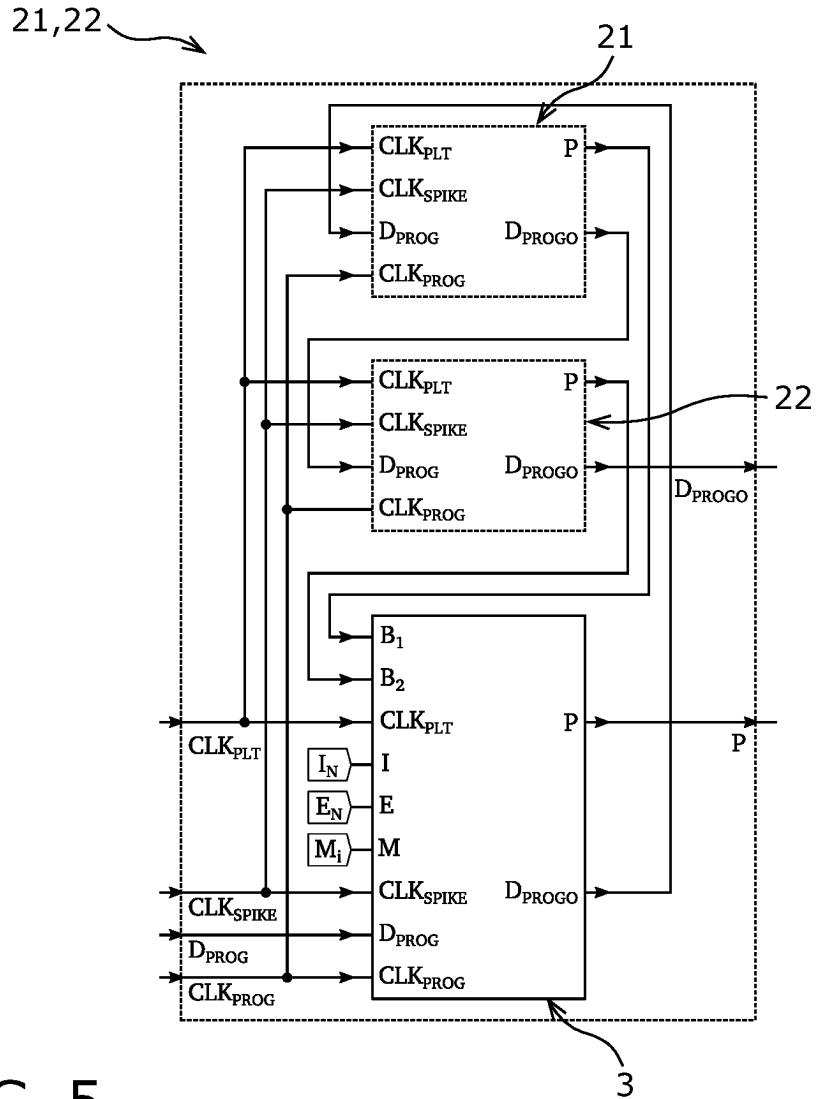


FIG. 5

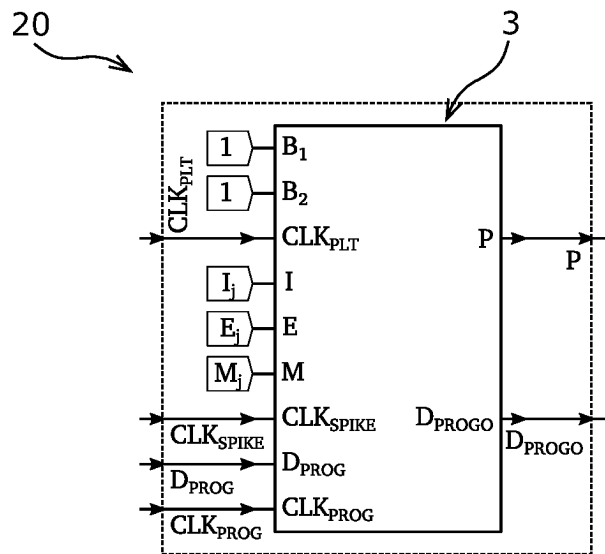


FIG. 6

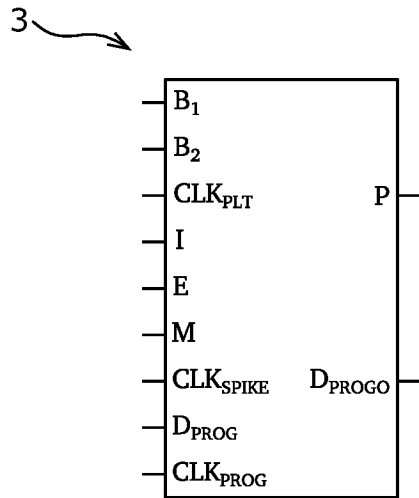


FIG. 7

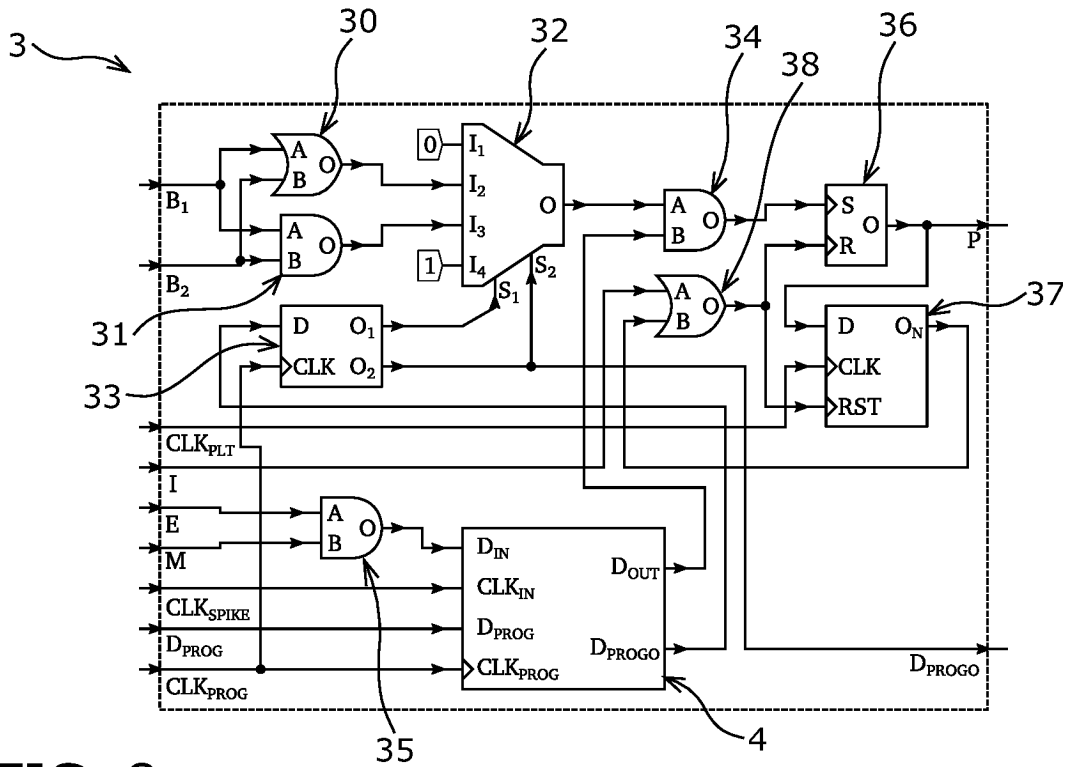


FIG. 8

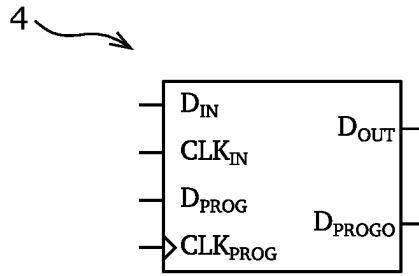


FIG. 9

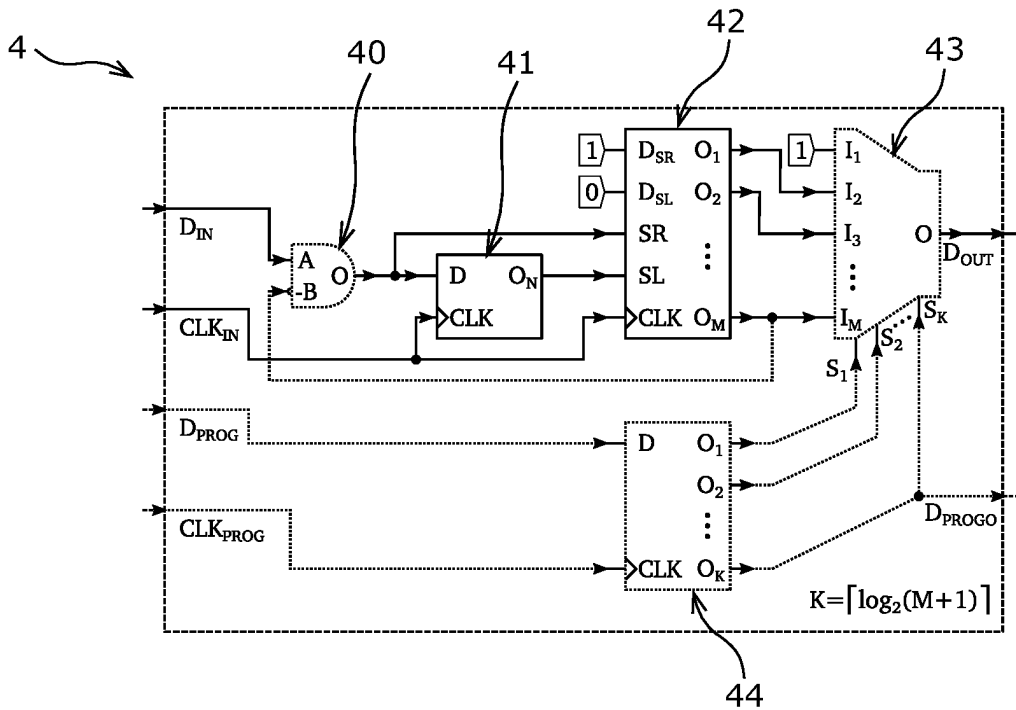


FIG. 10

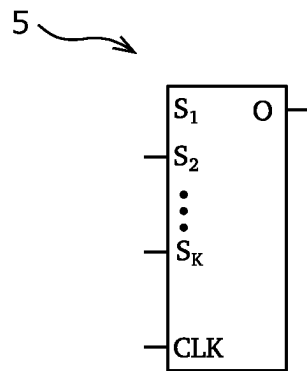


FIG. 11

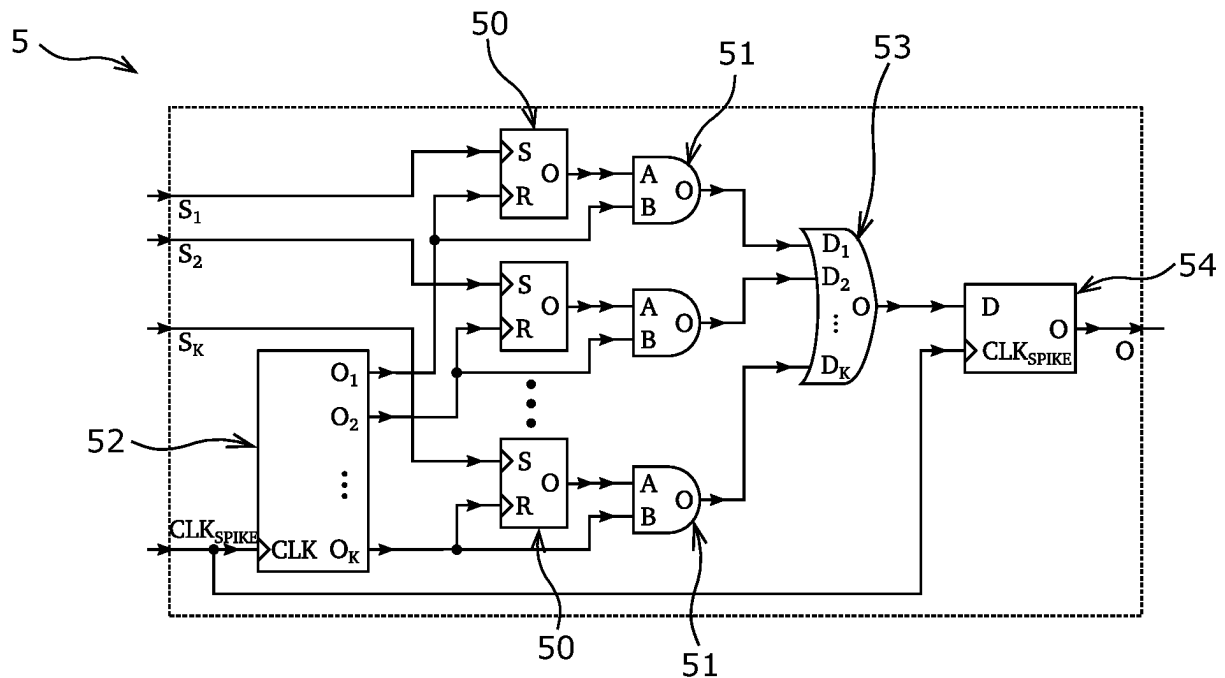


FIG. 12

B.4 PAPER: NEUROMORPHIC COMPUTATION IN MULTI-DELAY COUPLED MODELS

Pascal Nieters, Johannes Leugering and Gordon Pipa. 'Neuromorphic computation in multi-delay coupled models'. In: *IBM Journal of Research and Development* 61.2/3 (2017), pp. 8–7

Abstract

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.

Full-print is omitted from the online version of this thesis due to copyright reasons. The full DOI of the publication is:

[10.1147/JRD.2017.2664698](https://doi.org/10.1147/JRD.2017.2664698)

B.5 CONFERENCE ABSTRACT: NEUROMORPHIC ADAPTIVE FILTERS
FOR EVENT DETECTION, TRAINED WITH A GRADIENT FREE
ONLINE LEARNING RULE

Pascal Nieters, Johannes Leugering and Gordon Pipa. 'Neuromorphic Adaptive Filters for event detection, trained with a gradient free on-line learning rule'. In: *Cognitive Computing – Merging Concepts with Hardware*. 2018

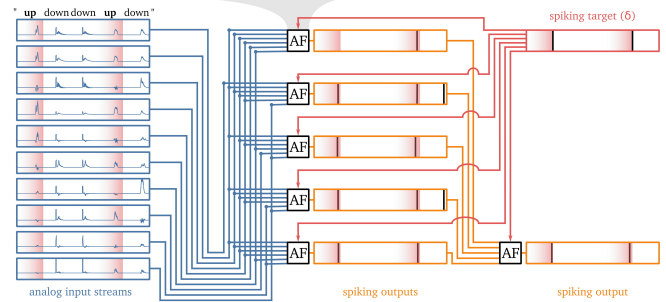
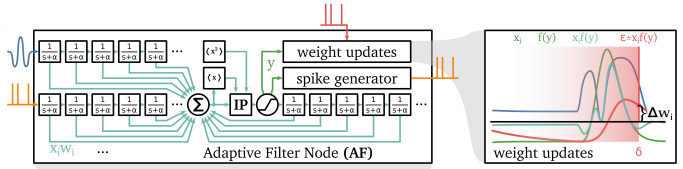
Abstract

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 problems, because their universal function approximation capability and differentiability can be leveraged for powerful gradient-based optimization algorithms. Neuromorphic hardware however, interacting with its environment in real time, faces challenges that defy this framework. One such example is the detection of specific events in real time, where the mapping from a continuous stream of noisy input signals onto a discrete set of events is to be learned. The temporal dimension of this task entails a credit assignment problem for learning, since the detector must evaluate a history of input signals and needs to be afforded some flexible processing delays, which makes defining a differentiable loss function for the event detection task difficult. This is aggravated in a setting where the target signals themselves are delayed. The constraints of neuromorphic hardware design further restrict the available learning algorithms to "any-time" computations implementable just by (traces of) locally available information, which precludes many of the established gradient-based optimization procedures. We propose a neuromorphic event detector, the Neuromorphic Adaptive Filter (NAF) and ensembles therefore, that utilizes Gamma Filter banks to learn a parameterized multidimensional signal filter through a supervised gradient-free on-line learning rule.



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 problems, because their universal function approximation capability and differentiability can be leveraged for powerful gradient-based optimization algorithms. Neuromorphic hardware however, interacting with its environment in real time, faces challenges that defy this framework. One such example is the **detection of specific events in real time**, where the mapping from a **continuous stream of noisy input signals** onto a **discrete set of events** is to be learned. The temporal dimension of this task entails a **credit assignment problem** for learning, since the detector must evaluate a history of input signals and needs to be afforded some flexible **processing delays**, which makes defining a differentiable loss function for the event detection task difficult. This is aggravated in a setting where the **target signals themselves are delayed**. The constraints of neuromorphic hardware design further restrict the available learning algorithms to "any-time" computations implementable just by (traces of) locally available information, which precludes many of the established gradient-based optimization procedures. We propose a neuromorphic event detector, the **Neuromorphic Adaptive Filter (NAF)** and ensembles thereof, that utilizes **Gamma Filter banks** [4,5] to learn a parameterized multidimensional signal filter through a **supervised gradient-free online learning rule**.



State & Weight Traces:

$$\begin{aligned} \hat{x}_0^j &= \alpha \cdot (I_j - x_0^j) \\ \hat{x}_1^j &= \alpha \cdot (x_{1-1}^j - x_1^j) \\ \hat{w}_1^j &= \eta \cdot (x_1^j \cdot \left(\frac{y-1}{2}\right)^2 - \hat{w}_1^j) \end{aligned}$$

Derived Variables:

$$\begin{aligned} \nu &= \sum_j x_1^j w_1^j \\ I_{th} &= \nu = \text{sat}\left(\frac{\nu - \mu}{\sqrt{2}\sigma^2}\right) \end{aligned}$$

Update on signal:

$$\begin{aligned} \theta &\leftarrow \theta + \tau^+ \\ r &\leftarrow r + 1 \\ w_i^j &\leftarrow w_i^j - \kappa \cdot \hat{w}_i^j \\ w &\leftarrow \frac{w}{||w||} \end{aligned}$$

Intrinsic Plasticity:

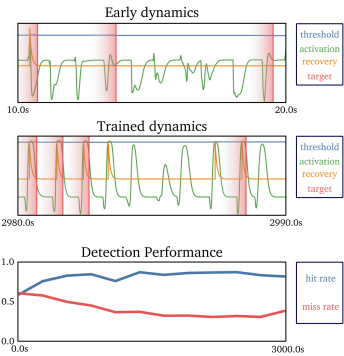
$$\begin{aligned} \hat{\mu} &= \beta \cdot (\nu - \mu) \\ \hat{\sigma}^2 &= \beta \cdot ((\nu - \mu)^2 - \sigma^2) \\ \hat{\theta} &\leftarrow -\gamma \cdot \theta \end{aligned}$$

Signal condition:

$$y \geq \theta + r$$

Update on target:

$$\begin{aligned} \theta &\leftarrow \theta - \tau^- \\ r &\leftarrow r + \tau^- \\ w_i^j &\leftarrow w_i^j + \kappa \cdot \hat{w}_i^j \\ w &\leftarrow \frac{w}{||w||} \end{aligned}$$



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 occurrence of the desired word ("UP") in the audio stream is shortly followed by a spiking target signal to each ensemble member.

ENAF 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.

References:

- [1] Güttig, 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.

- 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 incurs 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 weak 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**.

B.6 BOOK CHAPTER: COMPUTATIONAL ELEMENTS OF CIRCUITS

Johannes Leugering, Pascal Nieters and Gordon Pipa. 'Computational Elements of Circuits'. In: *The neocortex*. Ed. by Wolf Singer, Terrence J Sejnowski and Pasko Rakic. MIT Press, 2019, pp. 195–209

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

Full-print is omitted from the online version of this thesis due to copyright reasons. The ISBN for the complete book is: [9780262043243](#)

B.7 PAPER: A TRAJECTORY-BASED LOSS FUNCTION TO LEARN MISSING TERMS IN BIFURCATION DYNAMICAL SYSTEMS

Rahel Vortmeyer-Kley, Pascal Nieters and Gordon Pipa. 'A trajectory-based loss function to learn missing terms in bifurcating dynamical systems'. In: *Scientific reports* 11.1 (2021), pp. 1–13

Abstract

Missing terms in dynamical systems are a challenging problem for modeling. Recent developments in the combination of machine learning and dynamical system theory open possibilities for a solution. We show how physics-informed differential equations and machine learning – combined in the Universal Differential Equation (UDE) framework by Rackauckas et al. – can be modified to discover missing terms in systems that undergo sudden fundamental changes in their dynamical behavior called bifurcations. With this we enable the application of the UDE approach to a wider class of problems which are common in many real world applications. The choice of the loss function, which compares the training data trajectory in state space and the current estimated solution trajectory of the UDE to optimize the solution, plays a crucial role within this approach. The Mean Square Error as loss function contains the risk of a reconstruction which completely misses the dynamical behavior of the training data. By contrast, our suggested trajectory-based loss function which optimizes two largely independent components, the length and angle of state space vectors of the training data, performs reliable well in examples of systems from neuroscience, chemistry and biology showing Saddle-Node, Pitchfork, Hopf and Period-doubling bifurcations.



OPEN

A trajectory-based loss function to learn missing terms in bifurcating dynamical systems

Rahel Vortmeyer-Kley^{1,2}, Pascal Nieters^{1,2} & Gordon Pipa¹

Missing terms in dynamical systems are a challenging problem for modeling. Recent developments in the combination of machine learning and dynamical system theory open possibilities for a solution. We show how physics-informed differential equations and machine learning—combined in the Universal Differential Equation (UDE) framework by Rackauckas et al.—can be modified to discover missing terms in systems that undergo sudden fundamental changes in their dynamical behavior called bifurcations. With this we enable the application of the UDE approach to a wider class of problems which are common in many real world applications. The choice of the loss function, which compares the training data trajectory in state space and the current estimated solution trajectory of the UDE to optimize the solution, plays a crucial role within this approach. The Mean Square Error as loss function contains the risk of a reconstruction which completely misses the dynamical behavior of the training data. By contrast, our suggested trajectory-based loss function which optimizes two largely independent components, the length and angle of state space vectors of the training data, performs reliable well in examples of systems from neuroscience, chemistry and biology showing Saddle-Node, Pitchfork, Hopf and Period-doubling bifurcations.

Interacting oceanic flows governing our climate, increasing CO₂ levels causing global warming, competing species for coexistence, dominance or extinction—we live in a changing world. Understanding the impact of these often slow and gradual changes is important. But sometimes, a system can undergo a sudden fundamental change and end up in a totally different dynamical behavior. These points are sometimes called tipping points or regime shifts. Since the middle of the last century, the theory of dynamical systems has been applied to characterize the behavior of changing systems and real world phenomena in particular^{1–5} and examples therein.

The dynamics of the system can be described by the behavior of trajectories in a space defined by the state variables of the system. This state space contains repelling, attracting and separating structures—namely stable and unstable manifolds, and separatrices, and different types of attractors and coherent structures—that can act as organizing structures by governing the dynamical behavior of trajectories^{6,7}.

When modeling real world systems in simple equations we still face difficulties because our knowledge or understanding of the underlying processes is limited (e.g.^{8–10}). Here machine learning can help discover missing knowledge about the dynamical relationships between state variables from observed data by statistical inference. For example, Universal Differential Equations (UDE)¹¹ are a recently proposed method to learn dynamical systems from data with machine learning and can be combined with the Sparse Identification of Nonlinear Dynamics (SInDy) algorithm¹² to estimate an algebraic form of the dynamical system from data (see the next section for details). We show how this approach can be applied to learn missing terms in systems that can undergo sudden fundamental changes in their dynamics called bifurcation.

The goal of this work is to investigate the role of the loss function used to compare the learned dynamics and the training data given as time series data in learning a UDE. In particular, we propose a new loss function for optimization that compares angle and length of vectors in state space independently. Usually, the mean-squared error is used to compare time series for each variable in the system independently. Our idea is that this new Length Difference and Angle Difference (LDA) loss is more reliable when learning a UDE in many bifurcating systems and therefore is better suited to find missing terms in bifurcating systems. The examples we use for this comparison cover on the one hand a broad range of different bifurcation types (Saddle-Node, Pitchfork, Hopf and Period-doubling bifurcation) and on the other hand show the importance of bifurcations in various fields of research (e.g. neuroscience, biology, chemistry). Firstly, we set up a statistical comparison of how well the two loss functions perform in each of these systems in two different parameter regimes representing two different

¹Institute of Cognitive Science, Osnabrück University, Wachsbleiche 27, 49090 Osnabrück, Germany. ²These authors contributed equally: Rahel Vortmeyer-Kley and Pascal Nieters. ✉email: rahel.vortmeyer-kley@uni-osnabrueck.de

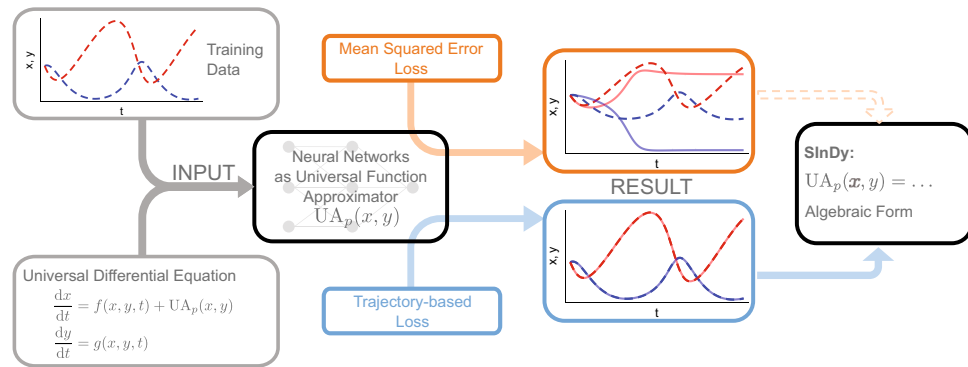


Figure 1. Schematic sketch of the concept of UDE¹¹. Depending on a successful approximation of the training data, the application of SInDy¹² is possible. The time series qualitatively correspond to the Selkov model¹⁵ in Table 2 second row. This figure is plotted using Inkscape (Version 1.0.1, <https://inkscape.org>) and Julia package Plots (Version v1.16.6, <https://github.com/JuliaPlots/Plots.jl>).

dynamics respectively. We then narrow this comparison by focusing on nested intervals around the bifurcation parameter as well as different trajectory starting points in state space. Finally, we show by way of example that an algebraic reconstruction of the missing terms using SInDy is possible if a UDE was trained successfully and discuss our results.

Universal differential equations (UDE) and sparse identification of nonlinear dynamics (SInDy). In this work, we investigate a complete, data-driven pipeline to discover the equations that describe a dynamical system in the context of bifurcating systems. The approach follows recently published work by Rackauckas *et al.* and combines the estimation of a UDE¹¹ from data and the subsequent identification of algebraic terms from the UDE using SInDy¹². We have illustrated the main steps involved in Fig. 1, starting with time series measurements from the system in question which will be used as training data.

Firstly, a Universal Differential Equation generally describes the dynamics of a system as $\dot{\mathbf{x}} = f(\mathbf{x}, t, UA_p(\mathbf{x}, t))$ where the function UA_p is a parameterized and differentiable universal function approximator. For example, a partially known two dimensional system with variables x and y may be described as:

$$\begin{aligned} \frac{dx}{dt} &= f(x, y, t) + UA_p(x, y) \\ \frac{dy}{dt} &= g(x, y, t) \end{aligned} \tag{1}$$

where functions f and g describe the known dynamics for x and y respectively, but we want to estimate an additional additive term in the dynamics of x . Given an initial $x(t_0)$ and $y(t_0)$ as well as initial parameters p_{init} the UDE can numerically be solved and trajectories $\hat{x}(t)$ and $\hat{y}(t)$ are estimated. Using a loss function, the Mean Squared Error loss (cf. next section) for each variable of a system as in the original work, the estimated trajectories based on the UDE can be compared against the actual system measured at sample points t_i . The entire program, from the definition of the unknown function UA_p and the UDE through to the numerical solution of the system and the calculation of the loss, can be differentiated with respect to parameters p using automatic differentiation in the Julia programming language^{13,14}. This enables the optimization of these parameters by gradient descent on the loss function, and thereby allows a user to find a UDE with parameterized function $UA_p^{optimal}$ that approximates the measured time series of the original system well.

Using differentiable models in machine learning problems is a generalized view of highly successful deep and artificial neural network (ANN) models¹⁶ that have previously been used to solve differential equations^{17,18}. Physics-informed neural networks, for example, aim to combine advantages of data-driven machine learning with knowledge about underlying physical laws in the training process to reconstruct dynamics of a system^{19–21}. In this paper, we use ANNs as the parameterized universal function approximator^{22,23} UA_p . The disadvantage is that neural networks are black boxes and do not allow us to learn about systematic relationships between the variables of the dynamical systems that may be the drivers of underlying dynamics.

Therefore, we also use the SInDy algorithm¹² to identify algebraic terms that can replace the neural network black box. In SInDy, finding the algebraic form of a differential equation is formulated as a linear regression problem. A matrix of non-linear functions Θ applied to the state vector \mathbf{x} of the system multiplied by a matrix of sparse coefficient vectors Ξ recovers the standard form of many non-linear dynamical systems: $\dot{\mathbf{x}} = f(\mathbf{x}) = \Theta(\mathbf{x}) \Xi$ (see the original paper¹² for examples). If measurements for both \mathbf{x} and $\dot{\mathbf{x}}$ are available at several sample points in time we can write $\dot{X} = \Theta(X) \Xi + \eta Z$ where X and \dot{X} have rows for each sample point, $\Theta(X)$ is a design matrix of non-linear functions applied to the data, and ηZ is independent and identically distributed Gaussian noise with magnitude η . This is a standard linear regression problem in multiple variables²⁴ where finding the sparse coefficient vectors $\Xi = [\xi_1, \dots, \xi_n]$ for a system with n state variables by using for example the LASSO algorithm²⁵ can find those non-linear functions in \mathbf{x} in the design matrix that best explain the data.

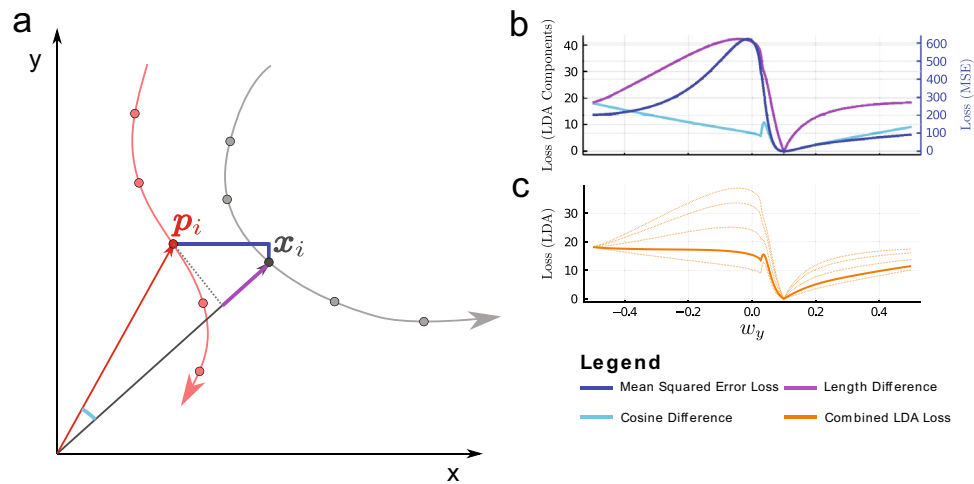


Figure 2. Schematic sketch of the concept of MSE and LDA loss function. **(a)** The UDE estimates trajectories $P = (\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n)$ (red) in state space, the training data is $X = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$ (gray). The MSE measures relative difference in each dimension of the sample point independently (blue lines), whereas the LDA measures the difference in angle (cyan) and difference in vector length (purple). **(b)** Comparing trajectories for the Selkov model¹⁵ (cf. Table 2 second row) based on MSE (blue), angular difference (cyan) via the cosine similarity and length difference (purple) all show a maximum that splits the loss landscape into two regions, one where the correct minimum can be reached and one where it cannot. Trajectories are compared based on a simple regression model in which only the weight for $y : w_y$ in the Selkov model is unknown and gives the x -axis here. **(c)** A weighted sum of the angular and length difference gives the LDA loss. Different sums are shown, with $k_1 = 0.25$ (length) and $k_2 = 0.75$ (angle) marked as the thick orange line that gives a good compromise between the two components. This figure is plotted using Inkscape (Version 1.0.1, <https://inkscape.org>) and Julia package Plots (Version v1.16.6, <https://github.com/JuliaPlots/Plots.jl>).

Often, data for \mathbf{x} is not abundant and $\dot{\mathbf{x}}$ is not immediately available. In these cases, we can still estimate the UDE that fits the data. Given a UDE, it is easy to calculate $\dot{\mathbf{x}}$ from any given \mathbf{x} and SInDy can be applied. Doing so will find functions in \mathbf{x} that are much easier to understand and analyze than a neural network and we have successfully removed the black box from the UDE. Furthermore, if partial knowledge about the system is included in the UDE, we can still use SInDy to find algebraic terms for the contribution by solving the regression problem $U A_p \text{optimal}(x) = \Theta(x)\Xi + \eta Z$ in isolation and complete the algebraic description of the dynamical system.

Results

The loss function and the trajectories. Generally speaking, the objective function or loss function in an optimization problem must be chosen in accordance with the goal of the optimized model. In neural network regression models, which we use here to find an approximation to missing terms in the UDE, this is typically done by minimization of the Mean Squared Error (MSE) loss function²⁶. However, more specialized practitioners of model optimization have sometimes found that particular choices of one subtly different loss function over another can significantly alter the quality of the model fit, for example in financial modeling of option pricing²⁷.

In the UDE approach, the MSE is used to compare observed dynamics to the estimated dynamics of the UDE by treating each component of the i -th sampled state space vector \mathbf{x}_i (observed) and \mathbf{p}_i (estimated) independently:

$$L_{\text{MSE}} = \sum_{i=1}^n \sum_{j=1}^m (\mathbf{x}_i^j - \mathbf{p}_i^j)^2, \tag{2}$$

where \mathbf{x}_i^j resp. \mathbf{p}_i^j is the j -th component in m dimensional state space, and the i -th point on a trajectory of length n (cf. blue difference vectors in Fig. 2a). This can work well in practice in many different UDE problems^{11,28}. However, when trying to find a missing term in a potentially bifurcating system, this particular choice of loss function can lead to unsatisfactory results.

For example, if the data generating system can undergo a Hopf bifurcation from a stable state solution to an oscillatory solution, a UDE trained with an MSE loss often does not achieve dynamics in the correct regime and does not produce dynamics that fit the original system's structure of nullclines and attractors (see animated state space portrait example of the oscillatory state of the Selkov model¹⁵ in the Supplementary video material S1 panel c).

How can the optimization fail so spectacularly? If we simplify the problem and replace the neural network by a simple linear regression with three parameters, one for each dimension of the state space and a bias parameter, we can visualize the loss function across any axis. If bias and weight for state variable x from the example in Fig. 1 (cf. Selkov model in Table 2 second row) are chosen correctly as 0, the correct solution for the weight for y is 0.1 in this case. Indeed, the MSE loss has its minimum at the correct value (Fig. 2b blue line), but it also has

a significant peak just below 0. For values smaller than 0, the gradient-descent based optimization of the UDE runs into an alternative local minimum. This alternative local minimum acts like a “black hole of loss” in the loss landscape and cannot be escaped. It becomes impossible to find the true solution. Thus, the quality of the approximation is entirely dependent on initial conditions (see results below as well as the animated state space portrait in the Supplementary video material S1 panel c).

In the setup using a neural network, we also observe the phenomenon that trajectories of the estimated and real system must first diverge so that the correct dynamical regime can be reached. While the loss landscape over many ANN parameters is not easily visualized, it can still contain local minima that represent alternative stable solutions and act as “black holes of loss”.

We realized that this problem is unlikely to be solved by a single loss function and were inspired by the view of the dynamics as a trajectory of state space vectors. These can also be described by their angle and length always having the origin in state space as their common reference.

We construct the Length Difference and Angle Difference Loss (LDA):

$$L_{LDA} = \sum_{i=1}^n \left(\underbrace{k_1 \cdot \frac{\sqrt{(|\mathbf{x}_i| - |\mathbf{p}_i|)^2}}{|\mathbf{x}_i| + |\mathbf{p}_i|}}_{\text{length difference}} + k_2 \cdot \underbrace{\frac{1}{2} \cdot \left(1 - \frac{\mathbf{x}_i \cdot \mathbf{p}_i}{|\mathbf{x}_i| \cdot |\mathbf{p}_i|} \right)}_{\text{angle difference}} \right), \quad (3)$$

where the first component describes the normalized length difference between the position vectors \mathbf{x}_i and \mathbf{p}_i of the i -th point on a trajectory of length n and the second component the normalized difference between the angles between the two position vectors (namely a normalized cosine similarity, cf. Fig. 2a).

Each component of LDA individually also has false local minima (Fig. 2b purple and cyan), but, crucially at different points. However, the global minimum is at the same location in both functions. The ratio of these two components can be weighted with the hyperparameter k_1 and k_2 (Fig. 2c) chosen for each problem. In most cases k_1 and k_2 are chosen equal to 0.5 except for the Selkov model, where k_1 equals 0.2 and k_2 equals 0.8. Exceptions to these values are indicated in the text.

This constructed loss function can be used to train the UDE successfully and is less dependent on the initial ANN parameter guess. The UDE now reconstructs the state space portrait of the original describing differential equation faithfully (see the animated state space portrait in the Supplementary video material S1 panel a).

In the simplified example, the combined LDA loss cannot fully eliminate a second erroneous local minimum using a simple weighting of each component. However, when using a neural network as the model, the parameter space has much larger dimensionality. We hypothesized that the basin of attraction for a correct minimum is much larger when using the LDA loss with appropriate weight parameters. This should result in a much higher chance to find a UDE that fits the data given random initial parameters than the MSE loss. Furthermore, we expected this effect to also depend on the bifurcation itself. In the following we tested both hypotheses extensively in computational experiments on four example systems with bifurcations.

The loss functions and different bifurcating systems. What happens if these loss functions are applied to different bifurcating systems from neuroscience, biology and chemistry? We considered the following four systems as examples of four different types of bifurcation:

- Saddle-Node bifurcation: The FitzHugh–Nagumo model^{29,30} describes resting and excited states of neurons. Specific parametrizations of the system can lead to a bistable behavior (“excitable”) with three fixed points where two are stable and one unstable (saddle). When the bifurcation parameter changes, the left stable fixed point and the saddle merge and disappear in a Saddle-Node bifurcation; the right stable fixed point remains and the system is monostable. (cf. Table 1)
- Pitchfork bifurcation: The Gardner model^{31,32} describes a genetic toggle switch in *Escherichia coli*. In one parametrization, the system is monostable with a stable fixed point but undergoes a Pitchfork bifurcation as the bifurcation parameter decreases, leading to bistable behavior with two stable fixed points and one saddle where each fixed point has its own basin of attraction. (cf. Table 1)
- Hopf bifurcation: The Selkov model¹⁵ describes oscillatory behavior in the enzyme reactions of the glycolysis process. The system shows either steady state behavior with a stable fixed point or—undergoing a Hopf bifurcation—ends up in oscillatory behavior with a stable limit cycle. (cf. Table 2)
- Period-doubling: The Rössler model^{33,34} can describe chemical reactions, but is often chosen as example for a simple period-doubling cascade. Here we focus on the transition from a period-one limit cycle to a period-two limit cycle with changing parametrization. (cf. Table 2)

We wanted to evaluate the performance of a UDE trained with the LDA loss function against a UDE trained with the MSE loss function for each model with two different parametrizations leading to the two different dynamical behaviors of the systems named above. Furthermore, we considered three different levels of normal distributed noise added to the training data to test the robustness of our results under more realistic conditions.

The universal function approximator $UA_p(x, y)$ (Equation 1) in all examples was a neural network with a single hidden layer and 16 neurons with *tanh*-activation function to accommodate a wide variety of possible non-linear functions with a limited number of ANN parameters. The initial weights of the Neural Network are chosen randomly using the Glorot initializer³⁵ with a Normal Distribution as its basis. We used the ADAM optimizer³⁶ and added weight decay to train the UDE (see Supplementary Table S1 for additional details).

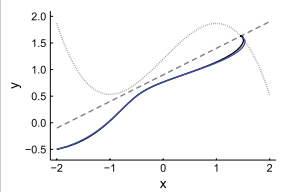
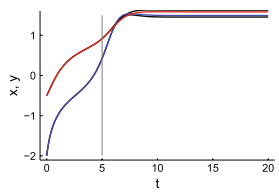
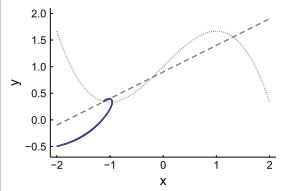
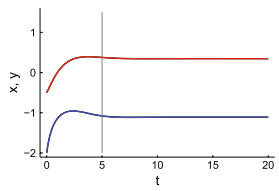
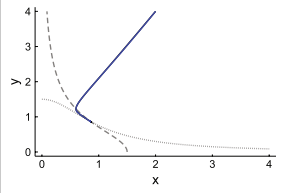
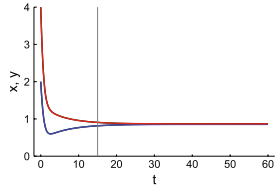
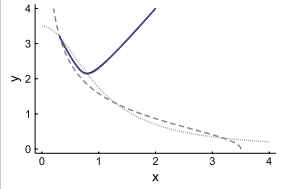
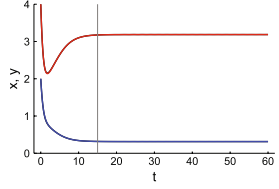
	UDE	State space	Time series	SInDy estimate	
Monostable FitzHugh–Nagumo	$\frac{dx}{dt} = x - \frac{x^3}{3} - y + 1.2$ $\frac{dy}{dt} = 1.25 \cdot (0.9 - y) + UA_p(x, y)$			$UA_p(x, y) = 0.625481 \cdot x,$ cf. missing term = $0.625 \cdot x$	Saddle-Node bifurcation
Bistable FitzHugh–Nagumo	$\frac{dx}{dt} = x - \frac{x^3}{3} - y + 1.0$ $\frac{dy}{dt} = 1.25 \cdot (0.9 - y) + UA_p(x, y)$			$UA_p(x, y) = 0.624665 \cdot x,$ cf. missing term = $0.625 \cdot x$	Saddle-Node bifurcation
Monostable Gardner	$\frac{dx}{dt} = UA_p(x, y) - x$ $\frac{dy}{dt} = \frac{1.5}{1+x^2} - y$			$UA_p(x, y) = \frac{1.499501}{1+y^2},$ cf. missing term = $\frac{1.5}{1+y^2}$	Pitchfork bifurcation
Bistable Gardner	$\frac{dx}{dt} = UA_p(x, y) - x$ $\frac{dy}{dt} = \frac{3.5}{1+x^2} - y$			$UA_p(x, y) = \frac{3.498237}{1+y^2},$ cf. missing term = $\frac{3.5}{1+y^2}$	Pitchfork bifurcation

Table 1. Overview of the models used with changing number of fixed points: FitzHugh–Nagumo model and Gardner model. First column: UDE for the systems in a general x and y notation with $UA_p(x, y)$ as machine learnable blank; second column: example of a trajectory’s longterm behavior trained with LDA (blue) in state space in comparison to the true solution’s trajectory (black). The nullcline given by the known part of the differential equation is dotted gray and the second nullcline of the true solution is gray dashed.; third column: corresponding time series of the longterm behavior of trained x (blue) and y (red) variable in comparison to the true solution (black, mainly masked by the predicted time series). The vertical gray line marks the end of the training data; fourth column: corresponding SInDy estimate of $UA_p(x, y)$ based on the approximation of the training data in comparison to the true missing term. The figures are plotted using Julia package Plots (Version v1.16.6, <https://github.com/JuliaPlots/Plots.jl>) and color-corrected and converted to eps using Adobe Illustrator (Version 25.4.1).

Tables 1 and 2 show representative results of the longterm prediction of the UDE model trained with LDA, where four times the length of the training data is used.

To compare estimated trajectories of the UDE trained with LDA or MSE loss objectively, we calculated the trajectory difference (TD) as the sum over the length of difference vectors over the sample points of the UDE and samples from the full differential equation:

$$TD = \frac{1}{n} \cdot \sum_{i=1}^n \|\mathbf{x}_i - \mathbf{p}_i\|, \tag{4}$$

where the true trajectory is described by the position vectors of the n trajectory points x_i and the approximated position vectors of the n trajectory points p_i .

Because the initial weights w_{init} of the machine learning model can considerably determine the success of UDE training (cp. Fig. 2), we independently drew 100 different sets of initial weights w_{init} per model, model parametrization and loss function and calculated TD for each independent experiment.

In Fig. 3 we show the distribution of a normalized version of TDs we get over all these 100 trained UDEs to compare the quality of the UDE solutions across different systems, parametrizations and loss function used in one figure. The general form of these distributions remains stable across all noise levels. All results for different noise levels can be found in Supplementary Figs. S1 and S2.

Systems with a changing number of fixed points. The Pitchfork and Saddle-Node bifurcations lead to a changing number of fixed points in the Gardner and FitzHugh–Nagumo systems, respectively. A good approximation of

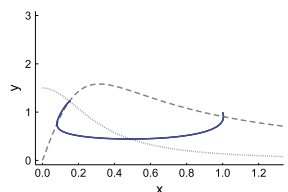
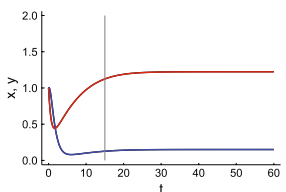
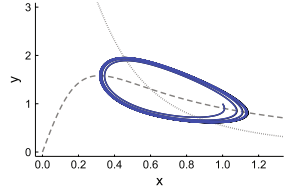
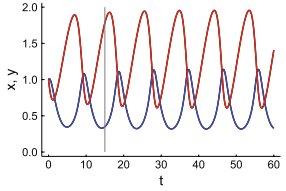
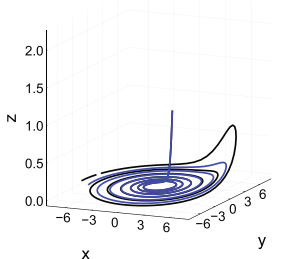
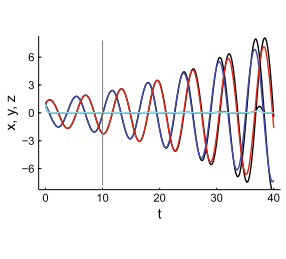
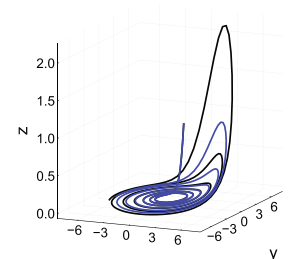
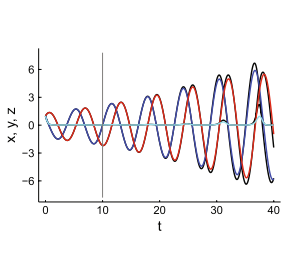
	UDE	State space	Time series	SInDy estimate	
Steady state Selkov	$\frac{dx}{dt} = -x + UA_p(x, y) + x^2 \cdot y$ $\frac{dy}{dt} = 0.15 - 0.1 \cdot y - x^2 \cdot y$			$UA_p(x, y) = 0.100014 \cdot y,$ cf. missing term = $0.1 \cdot y$	Hopf bifurcation
Oscillatory Selkov	$\frac{dx}{dt} = -x + UA_p(x, y) + x^2 \cdot y$ $\frac{dy}{dt} = 0.6 - 0.1 \cdot y - x^2 \cdot y$			$UA_p(x, y) = 0.099735 \cdot y,$ cf. missing term = $0.1 \cdot y$	Hopf bifurcation
Period-one Rössler	$\frac{dx}{dt} = -y - z$ $\frac{dy}{dt} = x + UA_p(x, y, z)$ $\frac{dz}{dt} = 0.1 + z \cdot x - 4.0 \cdot z$			$UA_p(x, y, z) = 0.100181 \cdot y,$ cf. missing term = $0.1 \cdot y$	Period-doubling
Period-two Rössler	$\frac{dx}{dt} = -y - z$ $\frac{dy}{dt} = x + UA_p(x, y, z)$ $\frac{dz}{dt} = 0.1 + z \cdot x - 6.0 \cdot z$			$UA_p(x, y, z) = 0.100183 \cdot y,$ cf. missing term = $0.1 \cdot y$	Period-doubling

Table 2. Overview of the models used with oscillatory behavior: Selkov model and Rössler model. First column: UDE for the systems in a general x and y resp. x, y and z notation with $UA_p(x, y)$ resp. $UA_p(x, y, z)$ as machine learnable blank; second column: example of a trajectory’s longterm behavior trained with LDA (blue) in state space in comparison to the true solution’s trajectory (black). In case of the Selkov model the nullcline given by the known part of the differential equation is dotted gray and the second nullcline of the true solution is gray dashed.; third column: corresponding time series of the longterm behavior of trained x (blue) and y (red) and in case of Rössler z (cyan) variable in comparison to the true solution (black, mainly masked by the predicted time series). The vertical gray line marks the end of the training data; fourth column: corresponding SInDy estimate of $UA_p(x, y)$ resp. $UA_p(x, y, z)$ based on the approximation of the training data in comparison to the true missing term. The figures are plotted using Julia package Plots (Version v1.16.6, <https://github.com/JuliaPlots/Plots.jl>) and color-corrected and converted to eps using Adobe Illustrator (Version 25.4.1).

the system’s longterm behavior, indicated by small TD between the approximated and the true solution’s trajectory, is in general possible with both loss functions (see Fig. 3).

However, for the FitzHugh–Nagumo model, the results using the MSE as the loss function show a higher median compared to the LDA loss and a larger distribution width in case of the monostable parametrization (cf. Fig. 3a). The reason why is that the predicted trajectories vary more around the true solution’s trajectory which leads to a number of trajectories ending in alternative stable states along the given nullcline (see Supplementary Fig. S3c,d). This effect is also visible in the outliers of the TD distribution for the bistable case for the MSE loss function, which make up around 12% of all trajectories end up in alternative stable states along the nullcline given by the known part of the differential equation in the region of the second fixed point (cf. Fig. 3b and Supplementary Fig. S3b). This second stable fixed point is also the only stable fixed point in case of the monostable parametrization (cf. Table 1). The LDA loss function does not lead to this behavior, because the angular difference between trajectories favors the correct curvature of the trajectory compared to absolutely correct values and guides the trajectory during training towards the correct fixed point. This reflects more precisely what we hope to achieve with the learned UDE, and leads to a compact TD distribution for the longterm prediction.

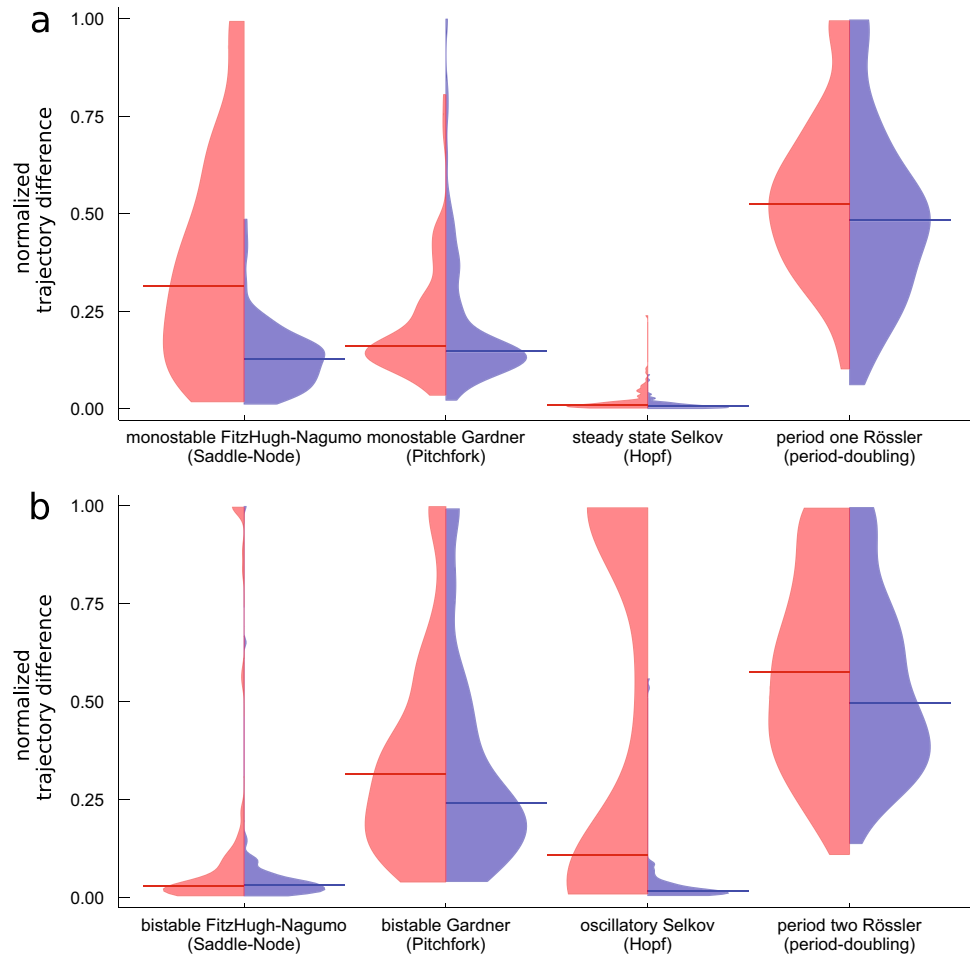


Figure 3. Distribution of the trajectory difference (TD) for the longterm prediction of the bifurcating systems over 100 neural network initializations trained using MSE loss function (red) or LDA loss function (blue). The horizontal bar indicates the median of the respective distribution. The training data contain additive normal distributed noise of noise level 0.0001. The trajectory difference is normalized by $TD = 1.0$ for FitzHugh–Nagumo, by $TD = 0.1$ for the Gardner, by $TD = 1.0$ for the Selkov and by $TD = 2.0$ for the Rössler model. All normalized values larger than one are clipped to one. The starting positions of the trajectories are chosen as for the examples in Tables 1 and 2. (a) Monostable, steady state and period-one parametrization (b) bistable, oscillatory and period-two parametrization. This figure is plotted using Julia package Plots (Version v1.16.6, <https://github.com/JuliaPlots/Plots.jl>) and color-corrected and converted to eps using Adobe Illustrator (Version 25.4.1).

Only around 2% of the trajectories in the bistable case end in the alternative stable state (cf. Fig. 3b and Supplementary Fig. S3a).

The same but somewhat weakened effect is observed for the bistable Gardner model (cf. Fig. 3b). Here, the results using the MSE loss function show 6–7% of the predicted longterm trajectories terminating in the alternative stable state on the given nullcline close to the fixed point the true solution's trajectory ends up in (cf. Table 1 and Supplementary Fig. S4b). The results using the LDA loss function only show up to 5% of the trajectories ending in alternative stable states (see Supplementary Fig. S4a).

The main difference in the bistable parametrization of the FitzHugh–Nagumo compared to the bistable Gardner model is, that the alternative states in the Gardner model are not linked to either the second stable fixed point or the stable fixed point in the monostable parametrization. Rather, they only occur spuriously in the functions permitted by the UDE in its current parametrization. The underlying dynamical structure of the Gardner system separates the state space into regions that specify to which fixed point all trajectories starting in this region converge. This dynamical behavior cannot be broken during the training. The alternative stable states can likely be linked to a new parametrization of the bistable Gardner system that co-evolves during training of the UDE.

In case of the monostable parametrization of the Gardner system, both loss functions lead to approximately equally good approximations of the longterm behavior of the system (cf. Fig. 3a and Supplementary Fig. S4c,d).

Systems with oscillatory behavior. The Hopf and period-doubling bifurcation lead to changing dynamical behavior in the Selkov and Rössler systems, from non-oscillatory to oscillatory behavior or from period one to period two oscillatory behavior respectively.

For steady state behavior of the Selkov system, both loss functions lead to good approximations of the longterm behavior of the system without strong dependence on the noise level, with a median TD of around 0.006 for the LDA loss function and around 0.012 for the MSE (cf. Fig. 3a and Supplementary Fig. S5c,d). By contrast, for the case of the oscillatory parametrization of the Selkov system, the results using the MSE loss function have a bimodal TD distribution, whereas the trajectory-based LDA loss function still leads to a narrow TD distribution and consistently good approximations (cf. Fig. 3b).

This is because many initial conditions in the MSE case lead to an alternative steady state solution during optimization which is a local minimum on the loss function that cannot be left (cf. Fig. 2b and Supplementary video material S1 panel b). The bimodal nature of the TD distribution reflects this: if the initial condition is favorable, the learning process finds the correct solution, if it is not, it fails completely (see supplemental Fig. S5b). The calculated median of the TD distribution shifts depending on the number of successful approximations and complete failures (see Supplementary Fig. S2b).

The LDA loss function on the other hand benefits from the combination of angular loss and the vector length loss, which allow for detours in state space (cf. Supplementary video material S1 panel a) bringing the approximated solution during training in the region of the training data. The solutions reliably reproduce the correct oscillations of the original system in nearly all cases (see Supplementary Fig. S5b).

For the case of the Rössler system, the approximations of the training data for both parametrizations show median values of TD around 0.04 for both parametrizations for the MSE and around 0.03 for both parametrizations for the trajectory-based loss function. The distribution is compact around the median with a few outliers (see Supplementary Fig. S6). By contrast, the longterm prediction of both used loss functions show median values of TD around 1.04 (period-one parametrization) resp. 1.22 (period-two parametrization) for MSE and around 0.95 (period-one parametrization) resp. 1.10 (period-two parametrization) for the LDA loss function (cf. Fig. 3a,b). Only a few runs yield good approximations. The reason why is that the training data do not include the typical excursions of the Rössler system in z-direction in state space which are included in the longterm prediction (cf. Table 2). These excursions are not approximated well by either of the trained networks. Longer training data could improve these results. Nevertheless, the accurate approximation of the short term training data enables a proper reconstruction of the algebraic form of the missing term as in Table 2.

The loss functions around the bifurcation. In the previous section we observed very different distributions for the trajectory difference depending on the chosen loss function and the parametrizations of the systems. To investigate if the distributions show a gradual change from one type to the other across the bifurcation, we performed 50 training runs for the FitzHugh–Nagumo and the Selkov model each with 14 different parametrization chosen from an interval nesting of the bifurcation parameter around the bifurcation. The bifurcation parameter for the case of FitzHugh–Nagumo is the 4th term in the first equation in Table 1 and in case of Selkov the first term in the second equation in Table 2. The FitzHugh–Nagumo and the Selkov model were chosen because they show the most different distributions of the trajectory difference TD for the two different dynamical behaviors (cf. Fig. 3). The results are presented in Fig. 4.

Firstly, we see a change of the shape of the MSE loss distribution as soon as the curvature of the trajectory increases (FitzHugh–Nagumo model Fig. 4a) or as soon as strongly damped oscillations start (Selkov model Fig. 4b between the light gray and the gray bar). The median of the LDA TD distribution is only sensitive to this in a narrow band of parameters shortly after the bifurcation in the FitzHugh–Nagumo model (cf. inset in Fig. 4a). Thus, the distributions reflect the bifurcation, too. As the curvature of the trajectory changes drastically across the bifurcation for both systems, the difference in the quality of the approximation using either loss function is markedly different. The reason for this behavior is that MSE only optimizes the difference in each variable in state space independently. This approach misses differences in the curvature of the training data trajectory and the current approximated solution. Thus, the MSE is often not suited to approximate a proper solution for a randomly chosen first guess of the approximated solution that belongs to a different dynamical regime than the training data and must be transferred to another dynamical regime during training by temporarily changing the curvature of the approximated solution drastically.

In contrast, the LDA loss function indirectly contains information about the curvature difference in its angle part of the loss function at a specific position in state space (length difference part of the loss function). Therefore, the LDA loss can capture changes in curvature earlier and permits the randomly chosen first guess UDE in the wrong dynamical regime to change to the dynamical regime of the training data more reliably. On the other hand considering small changes in the direction of a trajectory can immediately cause bad performance of LDA for specific noisy trajectory passing very close to a saddle. These small changes in the direction of the trajectory can lead to a switch of the dynamical behavior and a broader TD distribution (cf. Fig. 4a parameter 1.139 and 1.143).

In sum, there is an abrupt change from one type of distribution to another around the bifurcation reflecting the ability of the loss functions to reliably capture changes in curvature or not.

The loss functions and different starting points of trajectories. To investigate the impact of the starting position of the trajectories in state space on the performance of a UDE trained with the LDA loss function against a UDE trained with the MSE loss function, we again trained 50 UDEs for the FitzHugh–Nagumo and the Selkov model in the two parametrizations used in Tables 1 and 2 respectively. We chose starting points with very different dynamical behavior of the trajectories, namely slow and fast dynamics in state space as well as weak and strong curvatures of the trajectories. The results are presented in Fig. 5.

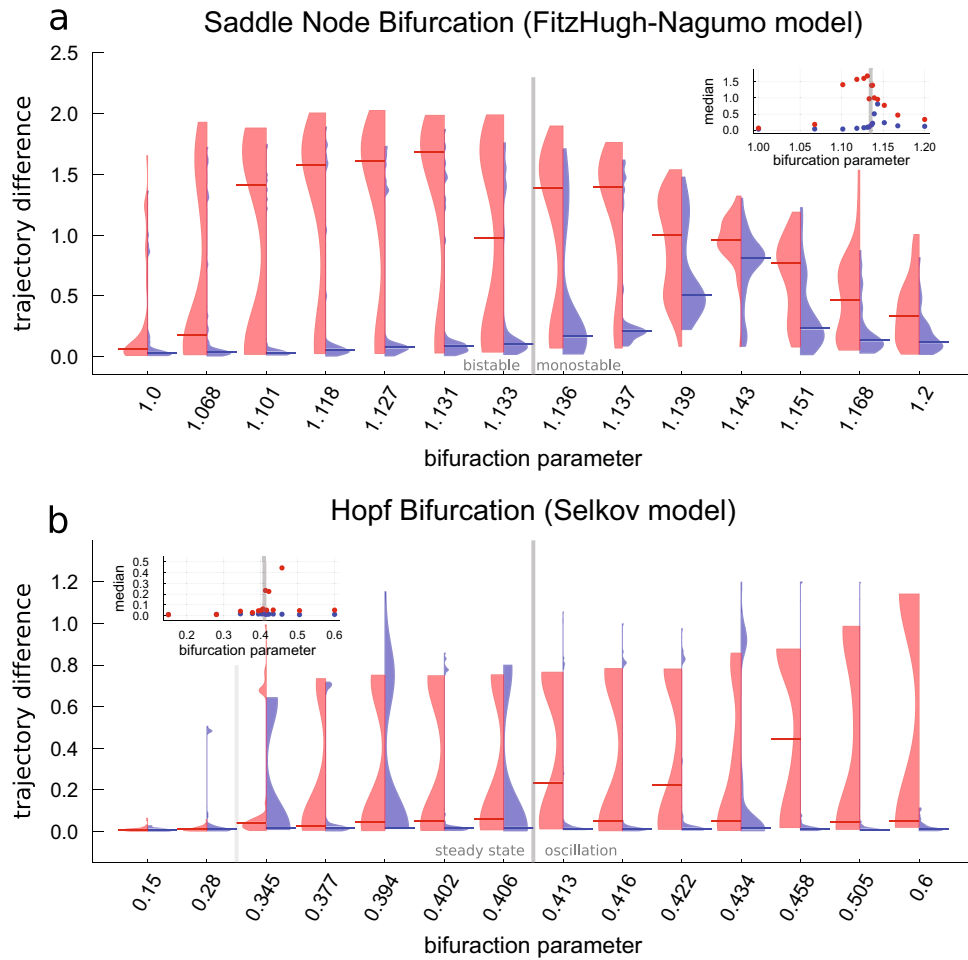


Figure 4. Distribution of the trajectory difference (TD) for the longterm prediction of the bifurcating systems over 50 neural network initializations trained using the MSE loss function (red) or LDA loss function (blue) for FitzHugh–Nagumo (a) and Selkov model (b) around the bifurcation (gray bar). The light gray bar in case of the Selkov model indicates the onset of strongly damped oscillations. The horizontal bar indicates the median of the respective distribution. The median of the distribution is plotted in the inset. The training data contain additive normal distributed noise with the noise level set to 0.0001. The trajectory difference larger than 1.2 are clipped to 1.2 in case of the Selkov model (b). The bifurcation parameter in case of FitzHugh–Nagumo is the 4th term in the first equation in Table 1 and in case of Selkov the first term in the second equation in Table 2. The trajectories start at $(-2.0, -0.25)$ in case of FitzHugh–Nagumo and at $(1.0, 1.0)$ in case of Selkov. This figure is plotted using Julia package Plots (Version v1.16.6, <https://github.com/JuliaPlots/Plots.jl>) and color-corrected and converted to eps using Adobe Illustrator (Version 25.4.1).

For both models we observe a strong dependence of the success of the training on the starting position in state space when using MSE. Less successful approximations are often linked with strong curvatures or oscillatory behavior. In case of the LDA loss function for the monostable FitzHugh–Nagumo model we observe less successful training for starting position B, C and E (cf. Fig. 5b). B and C share the common feature that they pass very slowly close to the saddle and the training data trajectory ends in this region, thus small changes of the trajectory due to noise can lead to a broadening of the distribution of the longterm prediction. Starting position E corresponds to a trajectory with a fast dynamics in case of the monostable parametrization and an extremely curved trajectory in case of the bistable parametrization. While we observe a wider distribution of trajectory differences in the monostable case, the LDA loss still captures the correct dynamical regime in case of the bistable parametrization and does not converge to a different fixed point as is possible for the MSE (cf. Fig. 5c).

When learning the Selkov model, the MSE shows a bimodal distribution for the oscillatory parametrization as expected, but this is also the case for starting point E in the steady state parametrization (cf. Fig. 5e). E is very close to second nullcline in state space which is unknown for the UDE. One possible explanation is that during training two different solution classes (steady state and oscillatory) are realized. We even observe a bimodal distribution in case of the LDA for the steady state and the oscillatory parametrization if we apply the standard values for the factors k_1 and k_2 used in the experiments above. If we adjust k_1 to 0.1 and k_2 to 0.9, the bimodal distribution vanishes and successful training is possible (cf. E in Fig. 5e,f). The same effect occurs for D in case

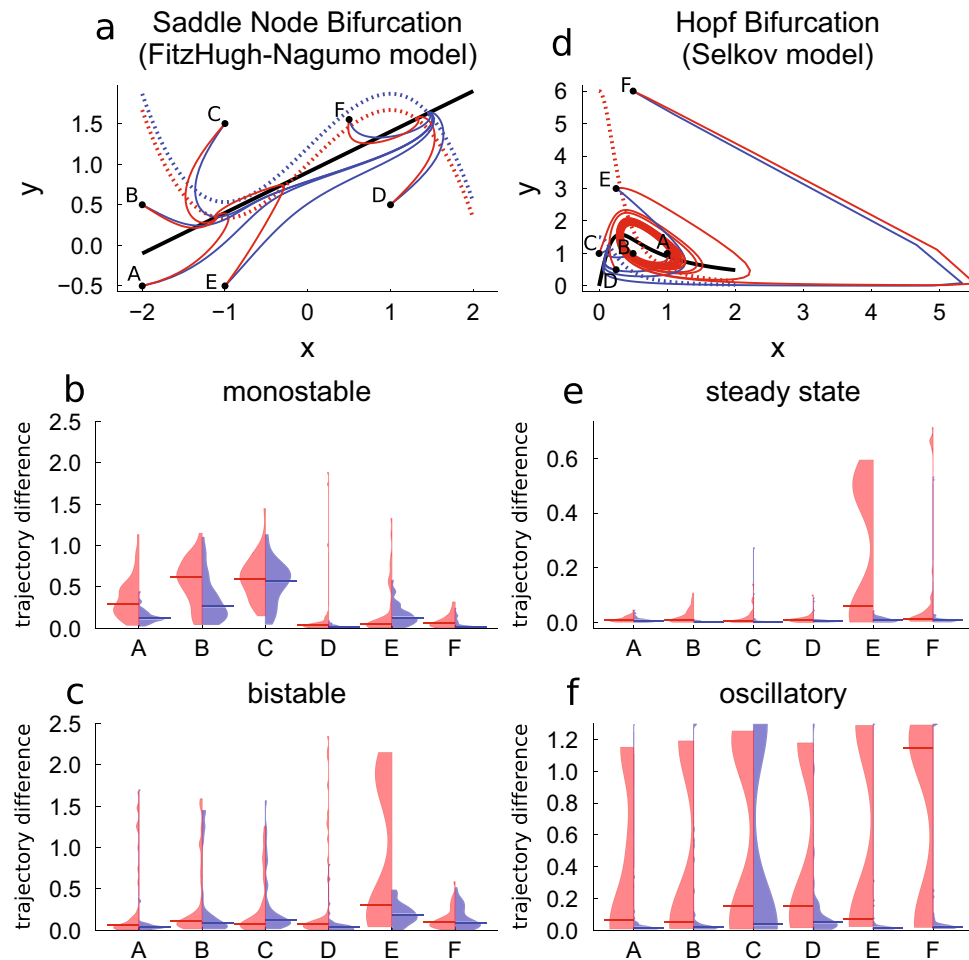


Figure 5. Distribution of the trajectory difference (TD) for the longterm prediction of the FitzHugh–Nagumo model (b, c) and the Selkov model (e, f) over 50 neural network initializations trained using the MSE loss function (red) or LDA loss function (blue) for starting position A to F of the trajectories (a, d) using two different parametrizations. The horizontal bar indicates the median of the respective distribution. The training data contain additive normal distributed noise with the noise level set to 0.0001. The red trajectories correspond to the bistable case (a) resp. oscillatory case (d), the blue one to the monostable case (a) resp. steady state (d). The black curve is the nullcline given by the known part of the differential equation, the dashed red (blue) curve is the second nullcline corresponding to the bistable resp. oscillatory state (to the monostable resp. steady state) in (a) resp. (b). This figure is plotted using Julia package Plots (Version v1.16.6, <https://github.com/JuliaPlots/Plots.jl>) and color-corrected and converted to eps using Adobe Illustrator (Version 25.4.1).

of the oscillatory parametrization (cf. Fig. 5f), whereas a change of k_1 and k_2 does not impact the results for C (cf. Fig. 5f).

To sum up, the starting position of the trajectory and its resulting properties can impact the success of the training. This is especially the case for the MSE, where there are no further hyperparameters to tune the solution.

SInDy's answer: algebraic form of the missing term. Finally, to get an idea of the algebraic form of the missing term, an approach for sparse identification of nonlinear dynamical systems (SInDy)¹² by Brunton et al. was applied to examples trained with the LDA loss function. This approach was chosen to show by way of example that a successful trained UDE system can be translated into an algebraic form. The examples are chosen from the statistical analysis with the lowest noise level such that the trajectory difference between the approximated solution of the training data and the training data itself is smallest. We investigated one example for each parametrization and each model. To improve the performance of SInDy, outliers in the beginning and the end of the approximated guess of the dynamics of the missing term are removed before SInDy is applied. The optimizer used by SInDy is STRRidge¹². The basis functions from which SInDy can select its guess are polynomials up to order of 5, except for the case of the Gardner model where polynomials up to order of 3 are combined with terms of the form $\frac{1}{1+z^n}$ with z being either x or y and $n = 1, 2, 3$. This exception was chosen to identify the term more clearly and not hidden it in a polynomial expansion of the function.

The results are shown in Tables 1 and 2. All in all, the algebraic form of the missing terms as well as the system parameters are approximated well.

The approach is not limited to the examples shown. It is possible to learn other terms or several terms at the same time. A crucial precondition for a successful reconstruction of the missing term with SInDy is an accurate approximation of the dynamics of the missing term with the UDE approach, which is directly linked with the successful reconstruction of the dynamical behavior of the training data. Thus, the success of the SInDy approach is directly linked with the accuracy of the results of the UDE approach. Furthermore, an accurate reconstruction of the missing term with SInDy based only on the approximation of the training data might give better results for the longterm prediction of the dynamics than the UDE, because the sparsity criterion of SInDy suppresses numerical fluctuations present in the UDE that might interfere with the longterm prediction.

Discussion and conclusion

In summary, the LDA loss function leads to more stable and accurate approximation of the longterm dynamical behavior in systems with a changing number of fixed points or a change in oscillatory behavior due to a bifurcation.

Often, trained networks using MSE as loss function lead to a solution which is only slightly worse than the networks using the LDA loss function, but there is a risk of complete failure of the approximation depending on the network initialization. Local minima in the MSE loss function due to different dynamical regimes of the UDE can represent completely different dynamical solutions, which gradient-based learning cannot recover from by adapting the gradient-descent procedure with, for example, momentum³⁷ or adaptive learning rates³⁶.

We used the LDA loss, which optimizes two components, the length and angle of state space vectors, with different local minima. A weighted combination of both components allowed us to estimate the dynamics of bifurcating systems more accurately, and avoided the catastrophic scenario of local minima leading to different dynamic regimes. While some randomly chosen initial parameters for the neural network can still lead to a bad fit of the UDE, we found that in examples where the MSE loss often finds local minima in the wrong dynamic regime, the basin of attraction of the optimization process was indeed much larger when we used the LDA loss instead. In some cases the choice of the factors weighting the two components can improve the results of the training as shown in case of the study of different starting points. We demonstrated that in applications, the loss function has to be chosen with great care based on the dynamics of the measured trajectories. This is because there is no gradual change from the possibility of successful training to failure across the bifurcation.

A more general open question of UDE is the impact of the training data length and its temporal resolution on the successful approximation of the system. A preliminary survey of this question for examples of the LDA loss does not indicate a clear effect for the temporal resolution (cf. supplementary Fig. S7) but as expected the trajectory differences between the approximated and the true solution trajectory decreases with increasing training data length (cf. supplementary Fig. S8). A more comprehensive study should be the content of further research, because in any real data set of measured time series the length and temporal resolution is fixed.

In real-world applications, physics-informed neural network approaches have successfully been applied to, among others, fluid dynamical problems^{38,39}. SInDy has also shown good results in complex data-driven settings^{40–42}. Universal Differential Equations¹¹ elegantly bring together two important aspects of these approaches: the advantages of knowledge about the underlying physics in the form of a differential equation, and the statistical estimation of dynamics via machine learning. SInDy¹² can be used to open the black box of machine learning and reconstruct algebraic terms to complete a differential equation. This approach to knowledge discovery is powerful. It implies that some interactions in the system of interest are not yet discovered, which is particularly true for—for example—ecosystems that cannot completely be examined in the lab or for systems for which the underlying governing equation is unknown. In such systems tipping points and potentially unknown bifurcations are of particular interest.

We have demonstrated, that UDEs can be used to estimate and reconstruct missing terms even in bifurcating systems. However, all examples shown here are simple compared to real-world applications. But even at this basic level we only have a rough idea of the interaction of the learning process and the bifurcation properties of the system. The interaction of both the dynamics of gradient-descent optimizing a loss function and the dynamics of the system of interest itself is highly fascinating and not yet understood. At this stage more fundamental research is necessary.

Particular care has to be taken in how the loss function is constructed for the optimization process, because the optimization process itself can lead to bifurcations. Nonetheless, our results suggest that dynamics in real-world, not yet fully described systems can be discovered using the LDA loss function.

Data availability

All experiments have been implemented in the Julia programming language¹³ based on packages developed particularly to fit UDEs with neural networks (<https://github.com/SciML/DiffEqFlux.jl>) and SInDy (<https://github.com/SciML/DataDrivenDiffEq.jl>). The code to produce all data presented in the paper is available publicly at <https://github.com/pnieters/GeneralizedDynamicsFromData>.

Code availability

Accession codes The code to produce all data presented in the paper is available publicly at <https://github.com/pnieters/GeneralizedDynamicsFromData>.

Received: 28 May 2021; Accepted: 21 September 2021

Published online: 14 October 2021

References

1. Steffen, W. *et al.* Trajectories of the Earth system in the anthropocene. *Proc. Natl. Acad. Sci.* **115**, 8252–8259. <https://doi.org/10.1073/pnas.1810141115> (2018).
2. Simonnet, E., Dijkstra, H. A. & Ghil, M. Bifurcation analysis of ocean, atmosphere, and climate models. In *Handbook of Numerical Analysis*, 187–229. [https://doi.org/10.1016/s1570-8659\(08\)00203-2](https://doi.org/10.1016/s1570-8659(08)00203-2) (Elsevier, 2009).
3. van Nes, E. H., Rip, W. J. & Scheffer, M. A theory for cyclic shifts between alternative states in shallow lakes. *Ecosystems* **10**, 17–28. <https://doi.org/10.1007/s10021-006-0176-0> (2007).
4. Stommel, H. Thermohaline convection with two stable regimes of flow. *Tellus* **13**, 224–230. <https://doi.org/10.1111/j.2153-3490.1961.tb00079.x> (1961).
5. Feudel, U., Pisarchik, A. N. & Showalter, K. Multistability and tipping: From mathematics and physics to climate and brain—Mini-review and preface to the focus issue. *Chaos Interdiscip. J. Nonlinear Sci.* **28**, 033501. <https://doi.org/10.1063/1.5027718> (2018).
6. Hadjighasem, A., Farazmand, M., Blazeovski, D., Froyland, G. & Haller, G. A critical comparison of Lagrangian methods for coherent structure detection. *Chaos Interdiscip. J. Nonlinear Sci.* **27**, 053104. <https://doi.org/10.1063/1.4982720> (2017).
7. Mancho, A. M., Small, D. & Wiggins, S. A tutorial on dynamical systems concepts applied to Lagrangian transport in oceanic flows defined as finite time data sets: Theoretical and computational issues. *Phys. Rep.* **437**, 55–124. <https://doi.org/10.1016/j.physrep.2006.09.005> (2006).
8. Boers, N., Ghil, M. & Rousseau, D.-D. Ocean circulation, ice shelf, and sea ice interactions explain Dansgaard–Oeschger cycles. *Proc. Natl. Acad. Sci.* **115**, E11005–E11014. <https://doi.org/10.1073/pnas.1802573115> (2018).
9. Shimoda, Y. & Arhonditsis, G. B. Phytoplankton functional type modelling: Running before we can walk? A critical evaluation of the current state of knowledge. *Ecol. Model.* **320**, 29–43. <https://doi.org/10.1016/j.ecolmodel.2015.08.029> (2016).
10. Edwards, A. M. & Brindley, J. Oscillatory behaviour in a three-component plankton population model. *Dyn. Stab. Syst.* **11**, 347–370. <https://doi.org/10.1080/0268119608806231> (1996).
11. Rackauckas, C. *et al.* Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385v1*, v2, v3 (2020).
12. Brunton, S. L., Proctor, J. L. & Kutz, J. N. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proc. Natl. Acad. Sci.* **113**, 3932–3937. <https://doi.org/10.1073/pnas.1517384113> (2016).
13. Bezanson, J., Edelman, A., Karpinski, S. & Shah, V. B. Julia: A fresh approach to numerical computing. *SIAM Rev.* **59**, 65–98. <https://doi.org/10.1137/141000671> (2017).
14. Innes, M. Don't unroll adjoint: Differentiating ssa-form programs. *arXiv preprint arXiv:1810.07951* (2018).
15. Sel'kov, E. E. Self-oscillations in glycolysis. 1. A simple kinetic model. *Eur. J. Biochem.* **4**, 79–86. <https://doi.org/10.1111/j.1432-1033.1968.tb00175.x> (1968).
16. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444. <https://doi.org/10.1038/nature14539> (2015).
17. Lagaris, I. E., Likas, A. & Fotiadis, D. I. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Trans. Neural Netw.* **9**, 987–1000. <https://doi.org/10.1109/72.712178> (1998).
18. Chen, R. T. Q., Rubanova, Y., Bettencourt, J. & Duvenaud, D. Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366* (2018). (Accessed 26 April 2021).
19. Raissi, M., Perdikaris, P. & Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **378**, 686–707. <https://doi.org/10.1016/j.jcp.2018.10.045> (2019).
20. Raissi, M., Perdikaris, P. & Karniadakis, G. E. Multistep neural networks for data-driven discovery of nonlinear dynamical systems. *arXiv preprint arXiv:1801.01236* (2018). (Accessed 26 April 2021).
21. Psychgiou, D. C. & Ungar, L. H. A hybrid neural network–first principles approach to process modeling. *AIChE J.* **38**, 1499–1511. <https://doi.org/10.1002/aic.690381003> (1992).
22. Cybenko, G. Approximation by superpositions of a sigmoidal function. *Math. Control Signals Syst.* **2**, 303–314. <https://doi.org/10.1007/BF02551274> (1989).
23. Hornik, K. Approximation capabilities of multilayer feedforward networks. *Neural Netw.* **4**, 251–257. [https://doi.org/10.1016/0893-6080\(91\)90009-T](https://doi.org/10.1016/0893-6080(91)90009-T) (1991).
24. Bishop, C. M. *Pattern Recognition and Machine Learning* (Springer, 2006).
25. Tibshirani, R. Regression shrinkage and selection via the lasso. *J. R. Stat. Soc. Series B (Methodol.)* **58**, 267–288. <https://doi.org/10.1111/j.2517-6161.1996.tb02080.x> (1996).
26. Goodfellow, I., Bengio, Y., Courville, A. & Bengio, Y. *Deep Learning* Vol. 1 (MIT Press Cambridge, 2016).
27. Christoffersen, P. & Jacobs, K. The importance of the loss function in option valuation. *J. Finan. Econ.* **72**, 291–318. <https://doi.org/10.1016/j.jfineco.2003.02.001> (2004).
28. Xiao, T. & Frank, M. Using neural networks to accelerate the solution of the Boltzmann equation. *arXiv preprint arXiv:2010.13649* (2020). (Accessed 26 April 2021).
29. FitzHugh, R. Mathematical models of threshold phenomena in the nerve membrane. *Bull. Math. Biophys.* **17**, 257–278. <https://doi.org/10.1007/bf0247753> (1955).
30. FitzHugh, R. Impulses and physiological states in theoretical models of nerve membrane. *Biophys. J.* **1**, 445–466. [https://doi.org/10.1016/s0006-3495\(61\)86902-6](https://doi.org/10.1016/s0006-3495(61)86902-6) (1961).
31. Gardner, T. S., Cantor, C. R. & Collins, J. J. Construction of a genetic toggle switch in *Escherichia coli*. *Nature* **403**, 339–342. <https://doi.org/10.1038/35002131> (2000).
32. Bose, I. & Ghosh, S. Bifurcation and criticality. *J. Stat. Mech. Theory Exp.* **2019**, 043403. <https://doi.org/10.1088/1742-5468/ab11d8> (2019).
33. Rössler, O. E. An equation for continuous chaos. *Phys. Lett. A* **57**, 397–398. [https://doi.org/10.1016/0375-9601\(76\)90101-8](https://doi.org/10.1016/0375-9601(76)90101-8) (1976).
34. Rössler, O. E. Chaotic behavior in simple reaction systems. *Zeitschrift für Naturforschung A* **31**, 259–264. <https://doi.org/10.1515/zna-1976-3-408> (1976).
35. Glorot, X. & Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 249–256 (JMLR Workshop and Conference Proceedings, 2010).
36. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)* (2015).
37. Nesterov, Y. E. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. In *Dokl. akad. nauk Sssr* **269**, 543–547 (1983).
38. Raissi, M., Yazdani, A. & Karniadakis, G. E. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science* **367**, 1026–1030. <https://doi.org/10.1126/science.aaw4741> (2020).
39. Raissi, M. & Karniadakis, G. E. Hidden physics models: Machine learning of nonlinear partial differential equations. *J. Comput. Phys.* **357**, 125–141. <https://doi.org/10.1016/j.jcp.2017.11.039> (2018).
40. Champion, K., Zheng, P., Aravkin, A. Y., Brunton, S. L. & Kutz, J. N. A unified sparse optimization framework to learn parsimonious physics-informed models from data. *IEEE Access* **8**, 169259–169271. <https://doi.org/10.1109/ACCESS.2020.3023625> (2020).
41. Champion, K., Lusch, B., Kutz, J. N. & Brunton, S. L. Data-driven discovery of coordinates and governing equations. *Proc. Natl. Acad. Sci.* **116**, 22445–22451. <https://doi.org/10.1073/pnas.1906995116> (2019).

42. Rudy, S. H., Brunton, S. L., Proctor, J. L. & Kutz, J. N. Data-driven discovery of partial differential equations. *Sci. Adv.* **3**, e1602614. <https://doi.org/10.1126/sciadv.1602614> (2017).

Acknowledgements

R.V.-K. and P.N. thank Georg Schröter for intensive discussions.

Author contributions

R.V.-K. developed the trajectory-based loss function. P.N. and R.V.-K. designed the numerical experiments. P.N. implemented the experiments. Overall supervision was done by G.P. All authors contributed in preparing this manuscript.

Funding

Open Access funding enabled and organized by Projekt DEAL.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at <https://doi.org/10.1038/s41598-021-99609-x>.

Correspondence and requests for materials should be addressed to R.V.-K.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

© The Author(s) 2021

B.8 CONFERENCE ABSTRACT: A TRAJECTORIES' GUIDE TO THE STATE SPACE - LEARNING MISSING TERMS IN BIFURCATING ECOLOGICAL SYSTEMS

Rahel Vortmeyer-Kley, Pascal Nieters and Gordon Pipa. 'A trajectories' guide to the state space-learning missing terms in bifurcating ecological systems'. In: *EGU General Assembly Conference Abstracts*. 2021, EGU21-16159. DOI: [10.5194/egusphere-egu21-16159](https://doi.org/10.5194/egusphere-egu21-16159). URL: <https://doi.org/10.5194/egusphere-egu21-16159>

Abstract

Ecological systems typically can exhibit various states ranging from extinction to coexistence of different species in oscillatory states. The switch from one state to another is called bifurcation. All these behaviours of a specific system are hidden in a set of describing differential equations (DE) depending on different parametrisations. To model such a system as DE requires full knowledge of all possible interactions of the system components. In practise, modellers can end up with terms in the DE that do not fully describe the interactions or in the worst case with missing terms.

The framework of universal differential equations (UDE) for scientific machine learning (SciML) [198] allows to reconstruct the incomplete or missing term from an idea of the DE and a short term timeseries of the system and make long term predictions of the system's behaviour. However, the approach in has difficulties to reconstruct the incomplete or missing term in systems with bifurcations. We developed a trajectory-based loss metric for UDE and SciML to tackle the problem and tested it successfully on a system mimicking algal blooms in the ocean.

BIBLIOGRAPHY

- [1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard et al. 'Tensorflow: A system for large-scale machine learning'. In: *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*. 2016, pp. 265–283.
- [2] Omar J Ahmed and Mayank R Mehta. 'Running speed alters the frequency of hippocampal gamma oscillations'. en. In: *J. Neurosci.* 32.21 (May 2012), pp. 7373–7383.
- [3] Rajeev Alur and David L Dill. 'A theory of timed automata'. In: *Theoretical computer science* 126.2 (1994), pp. 183–235.
- [4] Srdjan D Antic, Wen-Liang Zhou, Anna R Moore, Shaina M Short and Katerina D Ikonomu. 'The decade of the dendritic NMDA spike'. en. In: *J. Neurosci. Res.* 88.14 (Nov. 2010), pp. 2991–3001.
- [5] Stefan Apostel, Nicholas D Haynes, Eckehard Schöll, Otti D'Huys and Daniel J Gauthier. 'Reservoir Computing Using Autonomous Boolean Networks Realized on Field-Programmable Gate Arrays'. In: *Reservoir Computing*. Springer, 2021, pp. 239–271.
- [6] Lennert Appeltant, Miguel Cornelles Soriano, Guy Van der Sande, Jan Danckaert, Serge Massar, Joni Dambre, Benjamin Schrauwen, Claudio R Mirasso and Ingo Fischer. 'Information processing using a single dynamical node as complex system'. In: *Nature communications* 2.1 (2011), pp. 1–6.
- [7] Anish Athalye, Logan Engstrom, Andrew Ilyas and Kevin Kwok. 'Synthesizing robust adversarial examples'. In: *International conference on machine learning*. PMLR. 2018, pp. 284–293.
- [8] Mostafa Rahimi Azghadi, Bernabe Linares-Barranco, Derek Abbott and Philip HW Leong. 'A hybrid CMOS-memristor neuromorphic synapse'. In: *IEEE transactions on biomedical circuits and systems* 11.2 (2016), pp. 434–445.
- [9] 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'. en. In: *J. Comp. Neurol.* 231.2 (Jan. 1985), pp. 180–189.
- [10] Yoshua Bengio, Tristan Deleu, Edward J Hu, Salem Lahlou, Mo Tiwari and Emmanuel Bengio. 'GFlowNet Foundations'. In: *arXiv preprint arXiv:2111.09266* (2021).

- [11] David Beniaguev, Idan Segev and Michael London. ‘Single cortical neurons as deep artificial neural networks’. en. In: *Neuron* 109.17 (Sept. 2021), 2727–2739.e3.
- [12] O Bernander, R J Douglas, K A Martin and C Koch. ‘Synaptic background activity influences spatiotemporal integration in single pyramidal cells’. en. In: *Proc. Natl. Acad. Sci. U. S. A.* 88.24 (Dec. 1991), pp. 11569–11573.
- [13] Jeff Bezanson, Alan Edelman, Stefan Karpinski and Viral B Shah. ‘Julia: A fresh approach to numerical computing’. In: *SIAM review* 59.1 (2017), pp. 65–98.
- [14] Jacopo Bono and Claudia Clopath. ‘Modeling somatic and dendritic spike mediated plasticity at the single neuron and network level’. en. In: *Nat. Commun.* 8.1 (Sept. 2017), p. 706.
- [15] Valentino Braitenberg and Almut Schüz. *Cortex: statistics and geometry of neuronal connectivity*. Springer Science & Business Media, 2013.
- [16] Tiago Branco, Beverley A Clark and Michael Häusser. ‘Dendritic discrimination of temporal input sequences in cortical neurons’. In: *Science* 329.5999 (2010), pp. 1671–1675.
- [17] Tiago Branco and Michael Häusser. ‘The single dendritic branch as a fundamental functional unit in the nervous system’. en. In: *Curr. Opin. Neurobiol.* 20.4 (Aug. 2010), pp. 494–502.
- [18] Tiago Branco, Kevin Staras, Kevin J Darcy and Yukiko Goda. ‘Local dendritic activity sets release probability at hippocampal synapses’. en. In: *Neuron* 59.3 (Aug. 2008), pp. 475–485.
- [19] Federico Brandalise, Stefano Carta, Fritjof Helmchen, John Lisman and Urs Gerber. ‘Dendritic NMDA spikes are necessary for timing-dependent associative LTP in CA3 pyramidal cells’. In: *Nature communications* 7.1 (2016), pp. 1–9.
- [20] Johanni Brea, Alexisz Tamás Gaál, Robert Urbanczik and Walter Senn. ‘Prospective Coding by Spiking Neurons’. en. In: *PLoS Comput. Biol.* 12.6 (June 2016), e1005003.
- [21] Romain Brette and Wulfram Gerstner. ‘Adaptive Exponential Integrate-and-Fire Model as an Effective Description of Neuronal Activity’. In: *Journal of Neurophysiology* 94.5 (2005). PMID: 16014787, pp. 3637–3642. DOI: [10.1152/jn.00686.2005](https://doi.org/10.1152/jn.00686.2005). eprint: <https://doi.org/10.1152/jn.00686.2005>. URL: <https://doi.org/10.1152/jn.00686.2005>.
- [22] Bede M Broome, Vivek Jayaraman and Gilles Laurent. ‘Encoding and decoding of overlapping odor sequences’. en. In: *Neuron* 51.4 (Aug. 2006), pp. 467–482.

- [23] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell et al. 'Language models are few-shot learners'. In: *arXiv preprint arXiv:2005.14165* (2020).
- [24] Steven L Brunton, Joshua L Proctor and J Nathan Kutz. 'Discovering governing equations from data by sparse identification of nonlinear dynamical systems'. In: *Proceedings of the national academy of sciences* 113.15 (2016), pp. 3932–3937.
- [25] Julián Bueno, Daniel Brunner, Miguel C Soriano and Ingo Fischer. 'Conditions for reservoir computing performance using semiconductor lasers with delayed optical feedback'. In: *Optics express* 25.3 (2017), pp. 2401–2412.
- [26] Dean V Buonomano and Wolfgang Maass. 'State-dependent computations: spatiotemporal processing in cortical networks'. In: *Nature Reviews Neuroscience* 10.2 (2009), pp. 113–125.
- [27] A N Burkitt. 'A review of the integrate-and-fire neuron model: I. Homogeneous synaptic input'. en. In: *Biol. Cybern.* 95.1 (July 2006), pp. 1–19.
- [28] Samuel P Burns, Dajun Xing and Robert M Shapley. 'Is gamma-band activity in the local field potential of V1 cortex a "clock" or filtered noise?' In: *Journal of Neuroscience* 31.26 (2011), pp. 9658–9664.
- [29] Markus Butz, Florentin Wörgötter and Arjen van Ooyen. 'Activity-dependent structural plasticity'. en. In: *Brain Res. Rev.* 60.2 (May 2009), pp. 287–305.
- [30] György Buzsáki, Nikos Logothetis and Wolf Singer. 'Scaling brain size, keeping timing: evolutionary preservation of brain rhythms'. In: *Neuron* 80.3 (2013), pp. 751–764.
- [31] György Buzsáki and David Tingley. 'Space and time: The hippocampus as a sequence generator'. In: *Trends in cognitive sciences* 22.10 (2018), pp. 853–869.
- [32] J del Castillo and B Katz. 'Quantal components of the end-plate potential'. en. In: *J. Physiol.* 124.3 (June 1954), pp. 560–573.
- [33] Maria Chait, Steven Greenberg, Takayuki Arai, Jonathan Z Simon and David Poeppel. 'Multi-time resolution analysis of speech: evidence from psychophysics'. In: *Frontiers in neuroscience* 9 (2015), p. 214.
- [34] David J Chalmers. 'Connectionism and compositionality: Why Fodor and Pylyshyn were wrong'. In: (1993).

- [35] Elisabetta Chicca, Fabio Stefanini, Chiara Bartolozzi and Giacomo Indiveri. 'Neuromorphic electronic circuits for building autonomous cognitive systems'. In: *Proceedings of the IEEE* 102.9 (2014), pp. 1367–1388.
- [36] John Cohen. 'Psychological time'. In: *Scientific American* 211.5 (1964), pp. 116–125.
- [37] Matteo Cucchi, Christopher Gruener, Lautaro Petruskas, Peter Steiner, Hsin Tseng, Axel Fischer, Bogdan Penkovsky, Christian Matthus, Peter Birkholz, Hans Kleemann et al. 'Reservoir computing with biocompatible organic electrochemical networks for brain-inspired biosignal classification'. In: *Science Advances* 7.34 (2021), eabh0693.
- [38] George Cybenko. 'Approximation by superpositions of a sigmoidal function'. In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.
- [39] Matthew Dale, Julian F Miller, Susan Stepney and Martin A Trefzer. 'Evolving carbon nanotube reservoir computers'. In: *International Conference on Unconventional Computation and Natural Computation*. Springer. 2016, pp. 49–61.
- [40] Matthew Dale, Julian F Miller, Susan Stepney and Martin A Trefzer. 'Reservoir Computing in Material Substrates'. In: *Reservoir Computing*. Springer, 2021, pp. 141–166.
- [41] Keith Davids, Paul Glazier, Duarte Araujo and Roger Bartlett. 'Movement systems as dynamical systems'. In: *Sports medicine* 33.4 (2003), pp. 245–260.
- [42] Mike Davies et al. 'Loihi: A Neuromorphic Manycore Processor with On-Chip Learning'. In: *IEEE Micro* 38.1 (Jan. 2018), pp. 82–99.
- [43] Peter Dayan and Laurence F Abbott. *Theoretical neuroscience: computational and mathematical modeling of neural systems*. Computational Neuroscience Series, 2001.
- [44] Bert De Vries and Jose C Principe. 'The gamma model—A new neural model for temporal processing'. In: *Neural networks* 5.4 (1992), pp. 565–576.
- [45] Bert De Vries and José Príncipe. 'A theory for neural networks with time delays'. In: *Advances in neural information processing systems* 3 (1990).
- [46] Bryce S DeWitt. 'Quantum theory of gravity. I. The canonical theory'. In: *Physical Review* 160.5 (1967), p. 1113.

- [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'. en. In: *Cell Rep.* 21.6 (Nov. 2017), pp. 1550–1561.
- [48] Kenji Doya, Shin Ishii, Alexandre Pouget and Rajesh PN Rao. *Bayesian brain: Probabilistic approaches to neural coding.* 2007.
- [49] George Dragoi and György Buzsáki. 'Temporal encoding of place sequences by hippocampal cell assemblies'. In: *Neuron* 50.1 (2006), pp. 145–157.
- [50] George Dragoi and Susumu Tonegawa. 'Preplay of future place cell sequences by hippocampal cellular assemblies'. In: *Nature* 469.7330 (2011), pp. 397–401.
- [51] Naomi R Driesen, Gregory McCarthy, Zubin Bhagwagar, Michael H Bloch, Vincent D Calhoun, Deepak C D'souza, Ralitza Gueorguieva, George He, Hoi-Chung Leung, Ramachandran Ramani et al. 'The impact of NMDA receptor blockade on human working memory-related prefrontal function and connectivity'. In: *Neuropsychopharmacology* 38.13 (2013), pp. 2613–2622.
- [52] Céline Drieu and Michaël Zugaro. 'Hippocampal sequences during exploration: mechanisms and functions'. In: *Frontiers in cellular neuroscience* 13 (2019), p. 232.
- [53] K Du, Y W Wu, R Lindroos, Y Liu and others. 'Cell-type-specific inhibition of the dendritic plateau potential in striatal spiny projection neurons'. In: *Proceedings of the* (2017).
- [54] Simon Du, Jason Lee, Haochuan Li, Liwei Wang and Xiyu Zhai. 'Gradient descent finds global minima of deep neural networks'. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 1675–1685.
- [55] Daniel Durstewitz, Jeremy K Seamans and Terrence J Sejnowski. 'Neurocomputational models of working memory'. In: *Nature neuroscience* 3.11 (2000), pp. 1184–1191.
- [56] Sarah L Eagleman and Valentin Dragoi. 'Image sequence reactivation in awake V4 networks'. In: *Proceedings of the National Academy of Sciences* 109.47 (2012), pp. 19450–19455.
- [57] Howard Eichenbaum. 'Time cells in the hippocampus: a new dimension for mapping memories'. In: *Nature Reviews Neuroscience* 15.11 (2014), pp. 732–744.
- [58] Howard Eichenbaum. 'On the Integration of Space, Time, and Memory'. en. In: *Neuron* 95.5 (Aug. 2017), pp. 1007–1018.

- [59] Matthias Ekman, Peter Kok and Floris P de Lange. 'Time-compressed preplay of anticipated events in human primary visual cortex'. In: *Nature Communications* 8.1 (2017), pp. 1–9.
- [60] A K Engel, P Fries, P König, M Brecht and W Singer. 'Temporal binding, binocular rivalry, and consciousness'. en. In: *Conscious. Cogn.* 8.2 (June 1999), pp. 128–151.
- [61] RC Evans and KT Blackwell. 'Calcium: amplitude, duration, or location?' In: *The Biological Bulletin* 228.1 (2015), pp. 75–83.
- [62] Chrisantha Fernando and Sampa Sojakka. 'Pattern recognition in a bucket'. In: *European conference on artificial life*. Springer. 2003, pp. 588–597.
- [63] Ulrike Feudel, Alexander N Pisarchik and Kenneth Showalter. 'Multistability and tipping: From mathematics and physics to climate and brain—Minireview and preface to the focus issue'. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 28.3 (2018), p. 033501.
- [64] R Douglas Fields. 'A new mechanism of nervous system plasticity: activity-dependent myelination'. en. In: *Nat. Rev. Neurosci.* 16.12 (Dec. 2015), pp. 756–767.
- [65] Peter SB Finnie, Robert W Komorowski and Mark F Bear. 'The spatiotemporal organization of experience dictates hippocampal involvement in primary visual cortical plasticity'. In: *bioRxiv* (2021).
- [66] J A Fodor and Z W Pylyshyn. 'Connectionism and cognitive architecture: a critical analysis'. en. In: *Cognition* 28.1-2 (Mar. 1988), pp. 3–71.
- [67] Jerry A Fodor. *The language of thought*. Vol. 5. Harvard university press, 1975.
- [68] Jerry A Fodor. 'The mind-body problem'. In: *Scientific american* 244.1 (1981), pp. 114–123.
- [69] Norbert J Fortin, Kara L Agster and Howard B Eichenbaum. 'Critical role of the hippocampus in memory for sequences of events'. In: *Nature neuroscience* 5.5 (2002), pp. 458–462.
- [70] Stan Franklin and Max Garzon. 'Neural computability'. In: *Progress in neural networks* 1.128,144 (1990).
- [71] Rainer W Friedrich and Gilles Laurent. 'Dynamic optimization of odor representations by slow temporal patterning of mitral cell activity'. In: *Science* 291.5505 (2001), pp. 889–894.

- [72] Takaichi Fukuda and Toshio Kosaka. 'Gap Junctions Linking the Dendritic Network of GABAergic Interneurons in the Hippocampus'. In: *Journal of Neuroscience* 20.4 (2000), pp. 1519–1528. ISSN: 0270-6474. DOI: [10.1523/JNEUROSCI.20-04-01519.2000](https://doi.org/10.1523/JNEUROSCI.20-04-01519.2000). eprint: <http://www.jneurosci.org/content/20/4/1519.full.pdf>. URL: <http://www.jneurosci.org/content/20/4/1519>.
- [73] Takaichi Fukuda, Toshio Kosaka, Wolf Singer and Ralf AW Galuske. 'Gap junctions among dendrites of cortical GABAergic neurons establish a dense and widespread intercolumnar network'. In: *Journal of Neuroscience* 26.13 (2006), pp. 3434–3443.
- [74] Steve B Furber, Francesco Galluppi, Steve Temple and Luis A Plana. 'The spinnaker project'. In: *Proceedings of the IEEE* 102.5 (2014), pp. 652–665.
- [75] Peng P Gao, Joseph W Graham, Wen-Liang Zhou, Jinyoung Jang, Sergio Angulo, Salvador Dura-Bernal, Michael Hines, William W Lytton and Srdjan D Antic. 'Local glutamate-mediated dendritic plateau potentials change the state of the cortical pyramidal neuron'. In: *Journal of Neurophysiology* 125.12 (2021), pp. 23–42.
- [76] Sonia Gasparini and Jeffrey C Magee. 'State-dependent dendritic computation in hippocampal CA1 pyramidal neurons'. en. In: *J. Neurosci.* 26.7 (Feb. 2006), pp. 2088–2100.
- [77] Sonia Gasparini, Michele Migliore and Jeffrey C Magee. 'On the initiation and propagation of dendritic spikes in CA1 pyramidal neurons'. en. In: *J. Neurosci.* 24.49 (Dec. 2004), pp. 11046–11056.
- [78] Jeffrey P Gavornik and Mark F Bear. 'Learned spatiotemporal sequence recognition and prediction in primary visual cortex'. en. In: *Nat. Neurosci.* 17.5 (May 2014), pp. 732–737.
- [79] Dileep George and Jeff Hawkins. 'Towards a mathematical theory of cortical micro-circuits'. en. In: *PLoS Comput. Biol.* 5.10 (Oct. 2009), e1000532.
- [80] W. Gerstner. 'Spike-response model'. In: *Scholarpedia* 3.12 (2008). revision #91800, p. 1343. DOI: [10.4249/scholarpedia.1343](https://doi.org/10.4249/scholarpedia.1343).
- [81] MOHAMED M Ghoneim, James V Hinrichs, STEVEN P Mewaldt and Ronald C Petersen. 'Ketamine: behavioral effects of subanesthetic doses.' In: *Journal of clinical psychopharmacology* 5.2 (1985), pp. 70–77.
- [82] Albert Gidon and Idan Segev. 'Principles governing the operation of synaptic inhibition in dendrites'. en. In: *Neuron* 75.2 (July 2012), pp. 330–341.

- [83] Ben Gold, Nelson Morgan and Dan Ellis. *Speech and audio signal processing: processing and perception of speech and music*. John Wiley & Sons, 2011.
- [84] Patricia S Goldman-Rakic. 'Cellular basis of working memory'. In: *Neuron* 14.3 (1995), pp. 477–485.
- [85] 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'. en. In: *J. Neurosci.* 17.1 (Jan. 1997), pp. 204–215.
- [86] Michael Graupner and Nicolas Brunel. 'Calcium-based plasticity model explains sensitivity of synaptic changes to spike pattern, rate, and dendritic location'. en. In: *Proc. Natl. Acad. Sci. U. S. A.* 109.10 (Mar. 2012), pp. 3991–3996.
- [87] Jordan Guerguiev, Timothy P Lillicrap and Blake A Richards. 'Towards deep learning with segregated dendrites'. en. In: *Elife* 6 (Dec. 2017).
- [88] Robert Gütig and Haim Sompolinsky. 'The tempotron: a neuron that learns spike timing-based decisions'. In: *Nature neuroscience* 9.3 (2006), pp. 420–428.
- [89] Jiang Hao, Xu-Dong Wang, Yang Dan, Mu-Ming Poo and Xiao-Hui Zhang. 'An arithmetic rule for spatial summation of excitatory and inhibitory inputs in pyramidal neurons'. en. In: *Proc. Natl. Acad. Sci. U. S. A.* 106.51 (Dec. 2009), pp. 21906–21911.
- [90] 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'. en. In: *J. Neurosci.* 29.10 (Mar. 2009), pp. 3233–3241.
- [91] Michael Häusser. 'Synaptic function: dendritic democracy'. In: *Current Biology* 11.1 (2001), R10–R12.
- [92] Jeff Hawkins and Subutai Ahmad. 'Why neurons have thousands of synapses, a theory of sequence memory in neocortex'. In: *Frontiers in neural circuits* 10 (2016), p. 23.
- [93] Nicholas D Haynes, Miguel C Soriano, David P Rosin, Ingo Fischer and Daniel J Gauthier. 'Reservoir computing with a single time-delay autonomous Boolean node'. In: *Physical Review E* 91.2 (2015), p. 020801.
- [94] Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Psychology Press, 1949.
- [95] Richard NA Henson and N Burgess. 'Representations of serial order'. In: *4th Neural Computation and Psychology Workshop, London, 9–11 April 1997*. Springer. 1998, pp. 283–300.

- [96] Andreas V M Herz, Tim Gollisch, Christian K Machens and Dieter Jaeger. 'Modeling single-neuron dynamics and computations: a balance of detail and abstraction'. en. In: *Science* 314.5796 (Oct. 2006), pp. 80–85.
- [97] Konstantin Hicke, Miguel Angel Escalona-Morán, Daniel Brunner, Miguel Cornelles Soriano, Ingo Fischer and Claudio R Mirasso. 'Information processing using transient dynamics of semiconductor lasers subject to delayed feedback'. In: *IEEE Journal of Selected Topics in Quantum Electronics* 19.4 (2013), pp. 1501610–1501610.
- [98] Ira J Hirsh. 'Auditory Perception of Temporal Order'. In: *J. Acoust. Soc. Am.* 31.6 (June 1959), pp. 759–767.
- [99] Alan L Hodgkin and Andrew F Huxley. 'Resting and action potentials in single nerve fibres'. In: *The Journal of physiology* 104.2 (1945), pp. 176–195.
- [100] Christoph Hoerl and Teresa McCormack. 'Thinking in and about time: A dual systems perspective on temporal cognition'. In: *Behavioral and Brain Sciences* 42 (2019).
- [101] Michael Hollmann and Stephen Heinemann. 'Cloned Glutamate Receptors'. en. In: *Annual review of neuroscience* (Nov. 2003).
- [102] Tage Honoré, Jørn Lauridsen and Povl Krogsgaard-Larsen. 'The binding of [3H] AMPA, a structural analogue of glutamic acid, to rat brain membranes'. In: *Journal of neurochemistry* 38.1 (1982), pp. 173–178.
- [103] Kurt Hornik. 'Some new results on neural network approximation'. In: *Neural networks* 6.8 (1993), pp. 1069–1072.
- [104] Auke Jan Ijspeert, Jun Nakanishi and Stefan Schaal. 'Movement imitation with nonlinear dynamical systems in humanoid robots'. In: *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*. Vol. 2. IEEE. 2002, pp. 1398–1403.
- [105] Bernd Illing, Jean Robin Ventura, Guillaume Bellec and Wulfram Gerstner. 'Local plasticity rules can learn deep representations using self-supervised contrastive predictions'. In: *Thirty-Fifth Conference on Neural Information Processing Systems*. 2021.
- [106] Giacomo Indiveri, Bernabé Linares-Barranco, Tara Julia Hamilton, André Van Schaik, Ralph Etienne-Cummings, Tobi Delbruck, Shih-Chii Liu, Piotr Dudek, Philipp Häfliger, Sylvie Renaud et al. 'Neuromorphic silicon neuron circuits'. In: *Frontiers in neuroscience* 5 (2011), p. 73.
- [107] Michael Innes. 'Don't unroll adjoint: Differentiating ssa-form programs'. In: *arXiv preprint arXiv:1810.07951* (2018).

- [108] Mike Innes. 'Flux: Elegant machine learning with Julia'. In: *Journal of Open Source Software* 3.25 (2018), p. 602.
- [109] Junji Ito, Pedro Maldonado, Wolf Singer and Sonja Grün. 'Saccade-related modulations of neuronal excitability support synchrony of visually elicited spikes'. In: *Cerebral cortex* 21.11 (2011), pp. 2482–2497.
- [110] Eugene M Izhikevich. 'Simple model of spiking neurons'. In: *IEEE Transactions on neural networks* 14.6 (2003), pp. 1569–1572.
- [111] Herbert Jaeger. 'The "echo state" approach to analysing and training recurrent neural networks-with an erratum note'. In: ().
- [112] Tim Jarsky, Alex Roxin, William L Kath and Nelson Spruston. 'Conditional dendritic spike propagation following distal synaptic activation of hippocampal CA1 pyramidal neurons'. en. In: *Nat. Neurosci.* 8.12 (Dec. 2005), pp. 1667–1676.
- [113] Edwin T Jaynes. *Probability theory: The logic of science*. Cambridge university press, 2003.
- [114] H Shawn Je, Feng Yang, Yuanyuan Ji, Guhan Nagappan, Barbara L Hempstead and Bai Lu. 'Role of pro-brain-derived neurotrophic factor (proBDNF) to mature BDNF conversion in activity-dependent competition at developing neuromuscular synapses'. en. In: *Proc. Natl. Acad. Sci. U. S. A.* 109.39 (Sept. 2012), pp. 15924–15929.
- [115] Ole Jensen and John E Lisman. 'Hippocampal CA3 region predicts memory sequences: accounting for the phase precession of place cells.' In: *Learning & memory* 3.2-3 (1996), pp. 279–287.
- [116] Hongbo Jia, Nathalie L Rochefort, Xiaowei Chen and Arthur Konnerth. 'Dendritic organization of sensory input to cortical neurons in vivo'. en. In: *Nature* 464.7293 (Apr. 2010), pp. 1307–1312.
- [117] Renaud Jolivet, Timothy J Lewis and Wulfram Gerstner. 'Generalized integrate-and-fire models of neuronal activity approximate spike trains of a detailed model to a high degree of accuracy'. en. In: *J. Neurophysiol.* 92.2 (Aug. 2004), pp. 959–976.
- [118] Eric R Kandel, James H Schwartz, Thomas M Jessell, Steven Siegelbaum, A James Hudspeth and Sarah Mack. *Principles of neural science*. Vol. 4. McGraw-hill New York, 2000.
- [119] 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'. In: *Elife* 8 (2019), e46966.

- [120] Farshad Khadivar, Ilaria Lauzana and Aude Billard. 'Learning dynamical systems with bifurcations'. In: *Robotics and Autonomous Systems* 136 (2021), p. 103700.
- [121] H G Kim, M Beierlein and B W Connors. 'Inhibitory control of excitable dendrites in neocortex'. en. In: *J. Neurophysiol.* 74.4 (Oct. 1995), pp. 1810–1814.
- [122] David C Knill and Alexandre Pouget. 'The Bayesian brain: the role of uncertainty in neural coding and computation'. In: *TRENDS in Neurosciences* 27.12 (2004), pp. 712–719.
- [123] C Koch, T Poggio and V Torre. 'Retinal ganglion cells: a functional interpretation of dendritic morphology'. en. In: *Philos. Trans. R. Soc. Lond. B Biol. Sci.* 298.1090 (July 1982), pp. 227–263.
- [124] Christof Koch. *Biophysics of computation: information processing in single neurons*. Oxford university press, 2004.
- [125] Peter König, Andreas K Engel and Wolf Singer. 'Integrator or coincidence detector? The role of the cortical neuron revisited'. In: *Trends in neurosciences* 19.4 (1996), pp. 130–137.
- [126] Clemens Korndörfer, Ekkehard Ullner, Jordi García-Ojalvo and Gordon Pipa. 'Cortical spike synchrony as a measure of input familiarity'. In: *Neural computation* 29.9 (2017), pp. 2491–2510.
- [127] Nikolaus Kriegeskorte and Pamela K Douglas. 'Cognitive computational neuroscience'. In: *Nature neuroscience* 21.9 (2018), pp. 1148–1160.
- [128] John H Krystal, Laurence P Karper, John P Seibyl, Glenna K Freeman, Richard Delaney, J Douglas Bremner, George R Heninger, Malcolm B Bowers and Dennis S Charney. 'Subanesthetic effects of the noncompetitive NMDA antagonist, ketamine, in humans: psychotomimetic, perceptual, cognitive, and neuroendocrine responses'. In: *Archives of general psychiatry* 51.3 (1994), pp. 199–214.
- [129] Matthew E Larkum, Thomas Nevian, Maya Sandler, Alon Polisky and Jackie Schiller. 'Synaptic integration in tuft dendrites of layer 5 pyramidal neurons: a new unifying principle'. In: *Science* 325.5941 (2009), pp. 756–760.
- [130] Karl Spencer Lashley. *The problem of serial order in behavior*. Vol. 21. Bobbs-Merrill, 1951.
- [131] Yann LeCun, Yoshua Bengio and Geoffrey Hinton. 'Deep learning'. In: *nature* 521.7553 (2015), pp. 436–444.
- [132] Albert K Lee and Matthew A Wilson. 'Memory of sequential experience in the hippocampus during slow wave sleep'. In: *Neuron* 36.6 (2002), pp. 1183–1194.

- [133] Robert Legenstein and Wolfgang Maass. 'Edge of chaos and prediction of computational performance for neural circuit models'. In: *Neural networks* 20.3 (2007), pp. 323–334.
- [134] R A Lester, J D Clements, G L Westbrook and C E Jahr. 'Channel kinetics determine the time course of NMDA receptor-mediated synaptic currents'. en. In: *Nature* 346.6284 (Aug. 1990), pp. 565–567.
- [135] Johannes Leugering, Pascal Nieters and Gordon Pipa. 'Computational Elements of Circuits'. In: *The neocortex*. Ed. by Wolf Singer, Terrence J Sejnowski and Pasko Rakic. MIT Press, 2019, pp. 195–209.
- [136] Johannes Leugering, Pascal Nieters and Gordon Pipa. 'A minimal model of neural computation with dendritic plateau potentials.' In: *bioRxiv* (2021), p. 690792.
- [137] Johannes Leugering, Pascal Nieters and Gordon Pipa. 'Neuromorphic Pattern Detector and Neuromorphic Circuitry Here-with'. Pat. DE:102019134044:A1. June 2021.
- [138] Johannes Leugering and Gordon Pipa. 'A unifying framework of synaptic and intrinsic plasticity in neural populations'. In: *Neural computation* 30.4 (2018), pp. 945–986.
- [139] Songting Li, Nan Liu, Xiaohui Zhang, David W McLaughlin, Douglas Zhou and David Cai. 'Dendritic computations captured by an effective point neuron model'. en. In: *Proc. Natl. Acad. Sci. U. S. A.* 116.30 (July 2019), pp. 15244–15252.
- [140] Shiyu Liang, Ruoyu Sun, Jason D Lee and Rayadurgam Srikant. 'Adding one neuron can eliminate all bad local minima'. In: *Advances in Neural Information Processing Systems* 31 (2018), pp. 4350–4360.
- [141] Timothy P Lillicrap, Adam Santoro, Luke Marris, Colin J Akerman and Geoffrey Hinton. 'Backpropagation and the brain'. en. In: *Nat. Rev. Neurosci.* 21.6 (June 2020), pp. 335–346.
- [142] John E Lisman, Jean-Marc Fellous and Xiao-Jing Wang. 'A role for NMDA-receptor channels in working memory'. In: *Nature neuroscience* 1.4 (1998), pp. 273–275.
- [143] John Lisman and Nelson Spruston. 'Postsynaptic depolarization requirements for LTP and LTD: a critique of spike timing-dependent plasticity'. en. In: *Nat. Neurosci.* 8.7 (July 2005), pp. 839–841.
- [144] Michael London and Michael Häusser. 'Dendritic computation'. en. In: *Annu. Rev. Neurosci.* 28 (2005), pp. 503–532.
- [145] Attila Losonczy and Jeffrey C Magee. 'Integrative properties of radial oblique dendrites in hippocampal CA1 pyramidal neurons'. en. In: *Neuron* 50.2 (Apr. 2006), pp. 291–307.

- [146] Attila Losonczy, Judit K Makara and Jeffrey C Magee. 'Compartmentalized dendritic plasticity and input feature storage in neurons'. In: *Nature* 452.7186 (2008), pp. 436–441.
- [147] Xuelin Lou, Fan Fan, Mirko Messa, Andrea Raimondi, Yumei Wu, Loren L Looger, Shawn M Ferguson and Pietro De Camilli. 'Reduced release probability prevents vesicle depletion and transmission failure at dynamin mutant synapses'. In: *Proceedings of the National Academy of Sciences* 109.8 (2012), E515–E523.
- [148] Junshi Lu, Lu Luo, Qian Wang, Fang Fang and Nihong Chen. 'Cue-triggered activity replay in human early visual cortex'. In: *Science China Life Sciences* 64.1 (2021), pp. 144–151.
- [149] Huan Luo and David Poeppel. 'Phase patterns of neuronal responses reliably discriminate speech in human auditory cortex'. en. In: *Neuron* 54.6 (June 2007), pp. 1001–1010.
- [150] Huan Luo and David Poeppel. 'Cortical oscillations in auditory perception and speech: evidence for two temporal windows in human auditory cortex'. In: *Frontiers in psychology* 3 (2012), p. 170.
- [151] Wolfgang Maass, Prashant Joshi and Eduardo D Sontag. 'Computational aspects of feedback in neural circuits'. In: *PLoS computational biology* 3.1 (2007), e165.
- [152] Wolfgang Maass, Thomas Natschläger and Henry Markram. 'Real-time computing without stable states: A new framework for neural computation based on perturbations'. In: *Neural computation* 14.11 (2002), pp. 2531–2560.
- [153] Michael C Mackey and Leon Glass. 'Oscillation and chaos in physiological control systems'. In: *Science* 197.4300 (1977), pp. 287–289.
- [154] Jeffrey C Magee and Erik P Cook. 'Somatic EPSP amplitude is independent of synapse location in hippocampal pyramidal neurons'. In: *Nature neuroscience* 3.9 (2000), pp. 895–903.
- [155] Guy Major, Matthew E Larkum and Jackie Schiller. 'Active properties of neocortical pyramidal neuron dendrites'. en. In: *Annu. Rev. Neurosci.* 36 (July 2013), pp. 1–24.
- [156] 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'. en. In: *J. Neurophysiol.* 99.5 (May 2008), pp. 2584–2601.
- [157] Pedro Maldonado, Cecilia Babul, Wolf Singer, Eugenio Rodriguez, Denise Berger and Sonja Grun. 'Synchronization of neuronal responses in primary visual cortex of monkeys viewing natural images'. In: *Journal of neurophysiology* 100.3 (2008), pp. 1523–1532.

- [158] Luca Manneschi, Matthew OA Ellis, Guido Gigante, Andrew C Lin, Paolo Del Giudice and Eleni Vasilaki. 'Exploiting multiple timescales in hierarchical echo state networks'. In: *Frontiers in Applied Mathematics and Statistics* 6 (2021), p. 76.
- [159] Gary F Marcus. *The algebraic mind: Integrating connectionism and cognitive science*. MIT press, 2003.
- [160] Gary Marcus. 'Deep learning: A critical appraisal'. In: *arXiv preprint arXiv:1801.00631* (2018).
- [161] Warren S McCulloch and Walter Pitts. 'A logical calculus of the ideas immanent in nervous activity'. In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.
- [162] Josh H McDermott, Michael Schemitsch and Eero P Simoncelli. 'Summary statistics in auditory perception'. en. In: *Nat. Neurosci.* 16.4 (Apr. 2013), pp. 493–498.
- [163] Carver Mead. 'Introduction to VLSI systems'. In: *IEEE Proceedings I-Solid-State and Electron Devices* 128.1 (1980), p. 18.
- [164] B W Mel. 'Synaptic integration in an excitable dendritic tree'. en. In: *J. Neurophysiol.* 70.3 (Sept. 1993), pp. 1086–1101.
- [165] Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura et al. 'A million spiking-neuron integrated circuit with a scalable communication network and interface'. In: *Science* 345.6197 (2014), pp. 668–673.
- [166] Marvin Minsky and Seymour Papert. *Perceptrons*. MIT Press, 1969.
- [167] Melanie Mitchell. 'Why AI is harder than we think'. In: *arXiv preprint arXiv:2104.12871* (2021).
- [168] Gianluigi Mongillo, Omri Barak and Misha Tsodyks. 'Synaptic theory of working memory'. In: *Science* 319.5869 (2008), pp. 1543–1546.
- [169] 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'. en. In: *Neuron* 12.3 (Mar. 1994), pp. 529–540.
- [170] Rubén Moreno-Bote. 'Poisson-like spiking in circuits with probabilistic synapses'. In: *PLoS computational biology* 10.7 (2014), e1003522.
- [171] Mark A Motter and Jose C Principe. 'A gamma memory neural network for system identification'. In: *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*. Vol. 5. IEEE. 1994, pp. 3232–3237.

- [172] William Muñoz, Robin Tremblay, Daniel Levenstein and Bernardo Rudy. 'Layer-specific modulation of neocortical dendritic inhibition during active wakefulness'. en. In: *Science* 355.6328 (Mar. 2017), pp. 954–959.
- [173] Alexander Neckar, Sam Fok, Ben V Benjamin, Terrence C Stewart, Nick N Oza, Aaron R Voelker, Chris Eliasmith, Rajit Manohar and Kwabena Boahen. 'Braindrop: A mixed-signal neuromorphic architecture with a dynamical systems-based programming model'. In: *Proceedings of the IEEE* 107.1 (2018), pp. 144–164.
- [174] Emre O Neftci, Bruno U Pedroni, Siddharth Joshi, Maruan Al-Shedivat and Gert Cauwenberghs. 'Stochastic synapses enable efficient brain-inspired learning machines'. In: *Frontiers in neuroscience* 10 (2016), p. 241.
- [175] John Ashworth Nelder and Robert WM Wedderburn. 'Generalized linear models'. In: *Journal of the Royal Statistical Society: Series A (General)* 135.3 (1972), pp. 370–384.
- [176] Daniel A Nicholson, Rachel Trana, Yael Katz, William L Kath, Nelson Spruston and Yuri Geinisman. 'Distance-dependent differences in synapse number and AMPA receptor expression in hippocampal CA1 pyramidal neurons'. In: *Neuron* 50.3 (2006), pp. 431–442.
- [177] Pascal Nieters, Johannes Leugering and Gordon Pipa. 'Neuromorphic computation in multi-delay coupled models'. In: *IBM Journal of Research and Development* 61.2/3 (2017), pp. 8–7.
- [178] Pascal Nieters, Johannes Leugering and Gordon Pipa. 'Neuromorphic Adaptive Filters for event detection, trained with a gradient free online learning rule'. In: *Cognitive Computing – Merging Concepts with Hardware*. 2018.
- [179] Pascal Nieters, Johannes Leugering and Gordon Pipa. 'Active dendrites implement probabilistic temporal logic gates.' In: *Proc. of the Computational Cognition Workshop Osnabrück*. 2019.
- [180] L Nowak, P Bregestovski, P Ascher, A Herbert and A Prochiantz. 'Magnesium gates glutamate-activated channels in mouse central neurones'. en. In: *Nature* 307.5950 (1984), pp. 462–465.
- [181] J O'Keefe and J Dostrovsky. 'The hippocampus as a spatial map. Preliminary evidence from unit activity in the freely-moving rat'. en. In: *Brain Res.* 34.1 (Nov. 1971), pp. 171–175.
- [182] J O'Keefe and M L Recce. 'Phase relationship between hippocampal place units and the EEG theta rhythm'. en. In: *Hippocampus* 3.3 (July 1993), pp. 317–330.

- [183] 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'. en. In: *Front. Cell. Neurosci.* 8 (Sept. 2014), p. 292.
- [184] S Ortín, Miguel C Soriano, L Pesquera, Daniel Brunner, D San-Martín, Ingo Fischer, CR Mirasso and JM Gutiérrez. 'A unified framework for reservoir computing and extreme learning machines based on a single time-delayed neuron'. In: *Scientific reports* 5.1 (2015), pp. 1–11.
- [185] Norman H Packard, James P Crutchfield, J Doyne Farmer and Robert S Shaw. 'Geometry from a time series'. In: *Physical review letters* 45.9 (1980), p. 712.
- [186] Agostina Palmigiano, Theo Geisel, Fred Wolf and Demian Battaglia. 'Flexible information routing by transient synchrony'. In: *Nature neuroscience* 20.7 (2017), pp. 1014–1022.
- [187] Bogdan Penkovsky, Xavier Porte, Maxime Jacquot, Laurent Larger and Daniel Brunner. 'Coupled nonlinear delay systems as deep convolutional neural networks'. In: *Physical review letters* 123.5 (2019), p. 054101.
- [188] Carl Adam Petri. 'Communication with automata'. In: (1966).
- [189] Brad E Pfeiffer and David J Foster. 'Hippocampal place-cell sequences depict future paths to remembered goals'. In: *Nature* 497.7447 (2013), pp. 74–79.
- [190] Gordon Pipa, Sonja Grün and Carl Van Vreeswijk. 'Impact of spike train autostructure on probability distribution of joint spike events'. In: *Neural Computation* 25.5 (2013), pp. 1123–1163.
- [191] Panayiota Poirazi, Terrence Brannon and Bartlett W Mel. 'Pyramidal neuron as two-layer neural network'. In: *Neuron* 37.6 (2003), pp. 989–999.
- [192] Alon Polsky, Bartlett W Mel and Jackie Schiller. 'Computational subunits in thin dendrites of pyramidal cells'. en. In: *Nat. Neurosci.* 7.6 (June 2004), pp. 621–627.
- [193] Alexandre Pouget, Jeffrey M Beck, Wei Ji Ma and Peter E Latham. 'Probabilistic brains: knowns and unknowns'. In: *Nature neuroscience* 16.9 (2013), pp. 1170–1178.
- [194] Alexandre Pouget, Jan Drugowitsch and Adam Kepecs. 'Confidence and certainty: distinct probabilistic quantities for different goals'. In: *Nature neuroscience* 19.3 (2016), pp. 366–374.
- [195] Jose C Principe, Bert De Vries and Pedro Guedes De Oliveira. 'The gamma-filter—a new class of adaptive IIR filters with restricted feedback'. In: *IEEE transactions on signal processing* 41.2 (1993), pp. 649–656.

- [196] Zenon W Pylyshyn. *Computation and cognition: Toward a foundation for cognitive science*. The MIT Press, 1986.
- [197] HuiXin Qin, Jun Ma, WuYin Jin and ChunNi Wang. 'Dynamics of electric activities in neuron and neurons of network induced by autapses'. In: *Science China Technological Sciences* 57.5 (2014), pp. 936–946.
- [198] Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, Ali Ramadhan and Alan Edelman. 'Universal differential equations for scientific machine learning'. In: *arXiv preprint arXiv:2001.04385* (2020).
- [199] W Rall. 'Experimental monosynaptic input-output relations in the mammalian spinal cord'. en. In: *J. Cell. Comp. Physiol.* 46.3 (Dec. 1955), pp. 413–437.
- [200] W Rall. 'Electrophysiology of a dendritic neuron model'. en. In: *Biophys. J.* 2.2 Pt 2 (Mar. 1962), pp. 145–167.
- [201] Paul Rhodes. 'The properties and implications of NMDA spikes in neocortical pyramidal cells'. en. In: *J. Neurosci.* 26.25 (June 2006), pp. 6704–6715.
- [202] Blake A Richards and Timothy P Lillicrap. 'Dendritic solutions to the credit assignment problem'. en. In: *Curr. Opin. Neurobiol.* 54 (Feb. 2019), pp. 28–36.
- [203] Pieter R Roelfsema and Anthony Holtmaat. 'Control of synaptic plasticity in deep cortical networks'. In: *Nature Reviews Neuroscience* 19.3 (2018), pp. 166–180.
- [204] S Rosen. 'Temporal information in speech: acoustic, auditory and linguistic aspects'. en. In: *Philos. Trans. R. Soc. Lond. B Biol. Sci.* 336.1278 (June 1992), pp. 367–373.
- [205] Frank Rosenblatt. 'The perceptron: a probabilistic model for information storage and organization in the brain.' In: *Psychological review* 65.6 (1958), p. 386.
- [206] Carlo Rovelli. *The order of time*. Riverhead books, 2019.
- [207] David E Rumelhart, Geoffrey E Hinton and Ronald J Williams. *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science, 1985.
- [208] Sara Sabour, Nicholas Frosst and Geoffrey E Hinton. 'Dynamic routing between capsules'. In: *arXiv preprint arXiv:1710.09829* (2017).
- [209] S Schaal, S Kotosaka and D Sternad. 'Non-linear dynamical systems as movement primitives'. In: *Proc. IEEE International Conference on Humanoid Robotics, 2000*. 2000.

- [210] Sebastian Schmitt, Johann Klähn, Guillaume Bellec, Andreas Grübl, Maurice Guettler, Andreas Hartel, Stephan Hartmann, Dan Husmann, Kai Husmann, Sebastian Jeltsch et al. 'Neuromorphic hardware in the loop: Training a deep spiking network on the brainscales wafer-scale system'. In: *2017 international joint conference on neural networks (IJCNN)*. IEEE. 2017, pp. 2227–2234.
- [211] Nicolas W Schuck and Yael Niv. 'Sequential replay of non-spatial task states in the human hippocampus'. en. In: *Science* 364.6447 (June 2019).
- [212] Johannes Schumacher, Hazem Toutounji and Gordon Pipa. 'An analytical approach to single node delay-coupled reservoir computing'. In: *International Conference on Artificial Neural Networks*. Springer. 2013, pp. 26–33.
- [213] Johannes Schumacher, Thomas Wunderle, Pascal Fries, Frank Jäkel and Gordon Pipa. 'A statistical framework to infer delay and direction of information flow from measurements of complex systems'. In: *Neural computation* 27.8 (2015), pp. 1555–1608.
- [214] Terrence J Sejnowski, Christof Koch and Patricia Smith Churchland. 'Computational neuroscience'. In: *Science* 241.4871 (1988), pp. 1299–1306.
- [215] EE Sel'Kov. 'Self-Oscillations in Glycolysis 1. A Simple Kinetic Model'. In: *European Journal of Biochemistry* 4.1 (1968), pp. 79–86.
- [216] H Sebastian Seung, Daniel D Lee, Ben Y Reis and David W Tank. 'The autapse: a simple illustration of short-term analog memory storage by tuned synaptic feedback'. In: *Journal of computational neuroscience* 9.2 (2000), pp. 171–185.
- [217] Glenn Shafer, Peter R Gillett and Richard Scherl. 'The logic of events'. In: *Ann. Math. Artif. Intell.* 28.1 (Oct. 2000), pp. 315–389.
- [218] Mina Shahi, Carl van Vreeswijk and Gordon Pipa. 'Serial Spike Time Correlations Affect Probability Distribution of Joint Spike Events'. In: *Frontiers in computational neuroscience* 10 (2016), p. 139.
- [219] Yuko Shimoda and George B Arhonditsis. 'Phytoplankton functional type modelling: Running before we can walk? A critical evaluation of the current state of knowledge'. In: *Ecological modelling* 320 (2016), pp. 29–43.
- [220] Seth L Shipman, Bruce E Herring, Young Ho Suh, Katherine W Roche and Roger A Nicoll. 'Distance-dependent scaling of AMPARs is cell-autonomous and GluA2 dependent'. In: *Journal of Neuroscience* 33.33 (2013), pp. 13312–13319.

- [221] Hava T Siegelmann and Eduardo D Sontag. 'On the computational power of neural nets'. In: *Journal of computer and system sciences* 50.1 (1995), pp. 132–150.
- [222] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dhharshan Kumaran, Thore Graepel et al. 'A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play'. In: *Science* 362.6419 (2018), pp. 1140–1144.
- [223] Eric Simonnet, Henk A Dijkstra and Michael Ghil. 'Bifurcation analysis of ocean, atmosphere, and climate models'. In: *Handbook of numerical analysis*. Vol. 14. Elsevier, 2009, pp. 187–229.
- [224] W Singer. 'Cortical dynamics'. In: *The neocortex*. MIT Press, 2019, pp. 167–194.
- [225] Wolf Singer, Andreas K Engel, Andreas K Kreiter, Matthias HJ Munk, Sergio Neuenschwander and Pieter R Roelfsema. 'Neuronal assemblies: necessity, signature and detectability'. In: *Trends in cognitive sciences* 1.7 (1997), pp. 252–261.
- [226] Paul Smolensky. 'On the proper treatment of connectionism'. en. In: *Behav. Brain Sci.* 11.1 (Mar. 1988), pp. 1–23.
- [227] Paul Smolensky. 'Tensor product variable binding and the representation of symbolic structures in connectionist systems'. In: *Artificial intelligence* 46.1-2 (1990), pp. 159–216.
- [228] William R Softky and Christof Koch. 'The highly irregular firing of cortical cells is inconsistent with temporal integration of random EPSPs'. In: *Journal of neuroscience* 13.1 (1993), pp. 334–350.
- [229] Min Song, Minseok Kang, Hyeonsu Lee, Yong Jeong and Se-Bum Paik. 'Classification of Spatiotemporal Neural Activity Patterns in Brain Imaging Data'. en. In: *Sci. Rep.* 8.1 (May 2018), p. 8231.
- [230] Nelson Spruston. 'Pyramidal neurons: dendritic structure and synaptic integration'. en. In: *Nat. Rev. Neurosci.* 9.3 (Mar. 2008), pp. 206–221.
- [231] Nelson Spruston, Greg Stuart and Michael Häusser. 'Principles of dendritic integration'. In: *Dendrites* 351.597 (2016), p. 1.
- [232] Will Steffen, Johan Rockström, Katherine Richardson, Timothy M Lenton, Carl Folke, Diana Liverman, Colin P Summerhayes, Anthony D Barnosky, Sarah E Cornell, Michel Crucifix et al. 'Trajectories of the Earth System in the Anthropocene'. In: *Proceedings of the National Academy of Sciences* 115.33 (2018), pp. 8252–8259.

- [233] Florian Stelzer, André Röhm, Raul Vicente, Ingo Fischer and Serhiy Yanchuk. 'Deep neural networks using a single neuron: folded-in-time architecture using feedback-modulated delay loops'. In: *Nature communications* 12.1 (2021), pp. 1–10.
- [234] C F Stevens. 'Quantal release of neurotransmitter and long-term potentiation'. en. In: *Cell* 72 Suppl (Jan. 1993), pp. 55–63.
- [235] Katherine R Storrs, Tim C Kietzmann, Alexander Walther, Johannes Mehrer and Nikolaus Kriegeskorte. 'Diverse deep neural networks all predict human IT well, after training and fitting'. In: *bioRxiv* (2020).
- [236] Christoph Stosiek, Olga Garaschuk, Knut Holthoff and Arthur Konnerth. 'In vivo two-photon calcium imaging of neuronal networks'. en. In: *Proc. Natl. Acad. Sci. U. S. A.* 100.12 (June 2003), pp. 7319–7324.
- [237] Steven H Strogatz. *Nonlinear dynamics and chaos with student solutions manual: With applications to physics, biology, chemistry, and engineering*. CRC press, 2018.
- [238] Greg Stuart and Nelson Spruston. 'Determinants of voltage attenuation in neocortical pyramidal neuron dendrites'. In: *Journal of Neuroscience* 18.10 (1998), pp. 3501–3510.
- [239] Mototaka Suzuki and Matthew E Larkum. 'Dendritic calcium spikes are clearly detectable at the cortical surface'. en. In: *Nat. Commun.* 8.1 (Aug. 2017), p. 276.
- [240] Naoya Takahashi, Christian Ebner, Johanna Sigl-Glöckner, Sara Moberg, Svenja Nierwetberg and Matthew E Larkum. 'Active dendritic currents gate descending cortical outputs in perception'. en. In: *Nat. Neurosci.* (Aug. 2020).
- [241] Naoya Takahashi, Kazuo Kitamura, Naoki Matsuo, Mark Mayford, Masanobu Kano, Norio Matsuki and Yuji Ikegaya. 'Locally synchronized synaptic inputs'. en. In: *Science* 335.6066 (Jan. 2012), pp. 353–356.
- [242] Naoya Takahashi, Thomas G Oertner, Peter Hegemann and Matthew E Larkum. 'Active cortical dendrites modulate perception'. In: *Science* 354.6319 (2016), pp. 1587–1590.
- [243] Floris Takens. 'Detecting strange attractors in turbulence'. In: *Dynamical systems and turbulence, Warwick 1980*. Springer, 1981, pp. 366–381.
- [244] Gouhei Tanaka, Tadayoshi Matsumori, Hiroaki Yoshida and Kazuyuki Aihara. 'Reservoir Computing with Diverse Timescales for Prediction of Multiscale Dynamics'. In: *arXiv preprint arXiv:2108.09446* (2021).

- [245] Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothée Masquelier and Anthony Maida. 'Deep learning in spiking neural networks'. In: *Neural Networks* 111 (2019), pp. 47–63.
- [246] Xiangbin Teng, Yue Sun and David Poeppel. 'Temporal order judgment reveals local-global auditory processes'. In: *Acta Acustica united with Acustica* 104.5 (2018), pp. 817–820.
- [247] Xiangbin Teng, Xing Tian and David Poeppel. 'Testing multi-scale processing in the auditory system'. In: *Scientific reports* 6.1 (2016), pp. 1–13.
- [248] Timo Teräsvirta. 'Specification, estimation, and evaluation of smooth transition autoregressive models'. In: *Journal of the American Statistical Association* 89.425 (1994), pp. 208–218.
- [249] Rory G Townsend and Pulin Gong. 'Detection and analysis of spatiotemporal patterns in brain activity'. en. In: *PLoS Comput. Biol.* 14.12 (Dec. 2018), e1006643.
- [250] Misha V Tsodyks, William E Skaggs, Terrence J Sejnowski and Bruce L McNaughton. 'Population dynamics and theta rhythm phase precession of hippocampal place cell firing: a spiking neuron model'. In: *Hippocampus* 6.3 (1996), pp. 271–280.
- [251] Misha Tsodyks, Klaus Pawelzik and Henry Markram. 'Neural networks with dynamic synapses'. In: *Neural computation* 10.4 (1998), pp. 821–835.
- [252] Alan M Turing. 'Computing machinery and intelligence'. In: *Parsing the turing test*. 1950.
- [253] Balázs B Ujfalussy, Judit K Makara, Máté Lengyel and Tiago Branco. 'Global and Multiplexed Dendritic Computations under In Vivo-like Conditions'. en. In: *Neuron* 100.3 (Nov. 2018), 579–592.e5.
- [254] Robert Urbanczik and Walter Senn. 'Learning by the dendritic prediction of somatic spiking'. en. In: *Neuron* 81.3 (Feb. 2014), pp. 521–528.
- [255] Hendrik Van Der Loos and Edmund M Glaser. 'Autapses in neocortex cerebri: synapses between a pyramidal cell's axon and its own dendrites'. In: *Brain research* 48 (1972), pp. 355–360.
- [256] Christoph Von Der Malsburg. 'The correlation theory of brain function'. In: *Models of neural networks*. Springer, 1994, pp. 95–119.
- [257] John Von Neumann. 'The computer and the brain'. In: *New Haven, Conn.: Yale Uni* (1958).

- [258] John Von Neumann. 'First Draft of a Report on the EDVAC'. In: *IEEE Annals of the History of Computing* 15.4 (1993), pp. 27–75.
- [259] Rahel Vortmeyer-Kley, Pascal Nieters and Gordon Pipa. 'A trajectories' guide to the state space-learning missing terms in bifurcating ecological systems'. In: *EGU General Assembly Conference Abstracts*. 2021, EGU21–16159. DOI: [10.5194/egusphere-egu21-16159](https://doi.org/10.5194/egusphere-egu21-16159). URL: <https://doi.org/10.5194/egusphere-egu21-16159>.
- [260] Rahel Vortmeyer-Kley, Pascal Nieters and Gordon Pipa. 'A trajectory-based loss function to learn missing terms in bifurcating dynamical systems'. In: *Scientific reports* 11.1 (2021), pp. 1–13.
- [261] Chunni Wang, Shengli Guo, Ying Xu, Jun Ma, Jun Tang, Faris Alzahrani and Aatef Hobiny. 'Formation of autapse connected to neuron and its biological function'. In: *Complexity* 2017 (2017).
- [262] Min Wang, Yang Yang, Ching-Jung Wang, Nao J Gamo, Lu E Jin, James A Mazer, John H Morrison, Xiao-Jing Wang and Amy FT Arnsten. 'NMDA receptors subserve persistent neuronal firing during working memory in dorsolateral prefrontal cortex'. In: *Neuron* 77.4 (2013), pp. 736–749.
- [263] Ruopeng Wang, Jia-Qin Yang, Jing-Yu Mao, Zhan-Peng Wang, Shuang Wu, Maojie Zhou, Tianyi Chen, Ye Zhou and Su-Ting Han. 'Recent advances of volatile memristors: Devices, mechanisms, and applications'. In: *Advanced Intelligent Systems* 2.9 (2020), p. 2000055.
- [264] Pete Warden. 'Speech commands: A dataset for limited-vocabulary speech recognition'. In: *arXiv preprint arXiv:1804.03209* (2018).
- [265] Jack Waters, Andreas Schaefer and Bert Sakmann. 'Backpropagating action potentials in neurones: measurement, mechanisms and potential functions'. In: *Progress in biophysics and molecular biology* 87.1 (2005), pp. 145–170.
- [266] Andrew M Wikenheiser and A David Redish. 'Hippocampal sequences and the cognitive map'. In: *Analysis and Modeling of Coordinated Multi-neuronal Activity*. Springer, 2015, pp. 105–129.
- [267] Willem A M Wybo, Benjamin Torben-Nielsen, Thomas Nevian and Marc-Oliver Gewaltig. 'Electrical Compartmentalization in Neurons'. en. In: *Cell Rep.* 26.7 (Feb. 2019), 1759–1773.e7.

- [268] Ning-Long Xu, Mark T Harnett, Stephen R Williams, Daniel Huber, Daniel H O'Connor, Karel Svoboda and Jeffrey C Magee. 'Nonlinear dendritic integration of sensory and motor input during an active sensing task'. en. In: *Nature* 492.7428 (Dec. 2012), pp. 247–251.
- [269] Shengjin Xu, Wanchen Jiang, Mu-ming Poo and Yang Dan. 'Activity recall in a visual cortical ensemble'. In: *Nature neuroscience* 15.3 (2012), pp. 449–455.
- [270] Andrew I Yang, Gulce N Dikecligil, Heidi Jiang, Sandhitsu R Das, Joel M Stein, Stephan U Schuele, Joshua M Rosenow, Kathryn A Davis, Timothy H Lucas and Jay A Gottfried. 'The what and when of olfactory working memory in humans'. en. In: *Curr. Biol.* (Aug. 2021).
- [271] Jianmin Yang et al. 'proBDNF negatively regulates neuronal remodeling, synaptic transmission, and synaptic plasticity in hippocampus'. en. In: *Cell Rep.* 7.3 (May 2014), pp. 796–806.
- [272] Friedemann Zenke and Surya Ganguli. 'Superspike: Supervised learning in multilayer spiking neural networks'. In: *Neural computation* 30.6 (2018), pp. 1514–1541.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both \LaTeX and \LyX :

<https://bitbucket.org/amiede/classicthesis/>

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

<http://postcards.miede.de/>

Final Version as of 25th May 2022 (`classicthesis`).