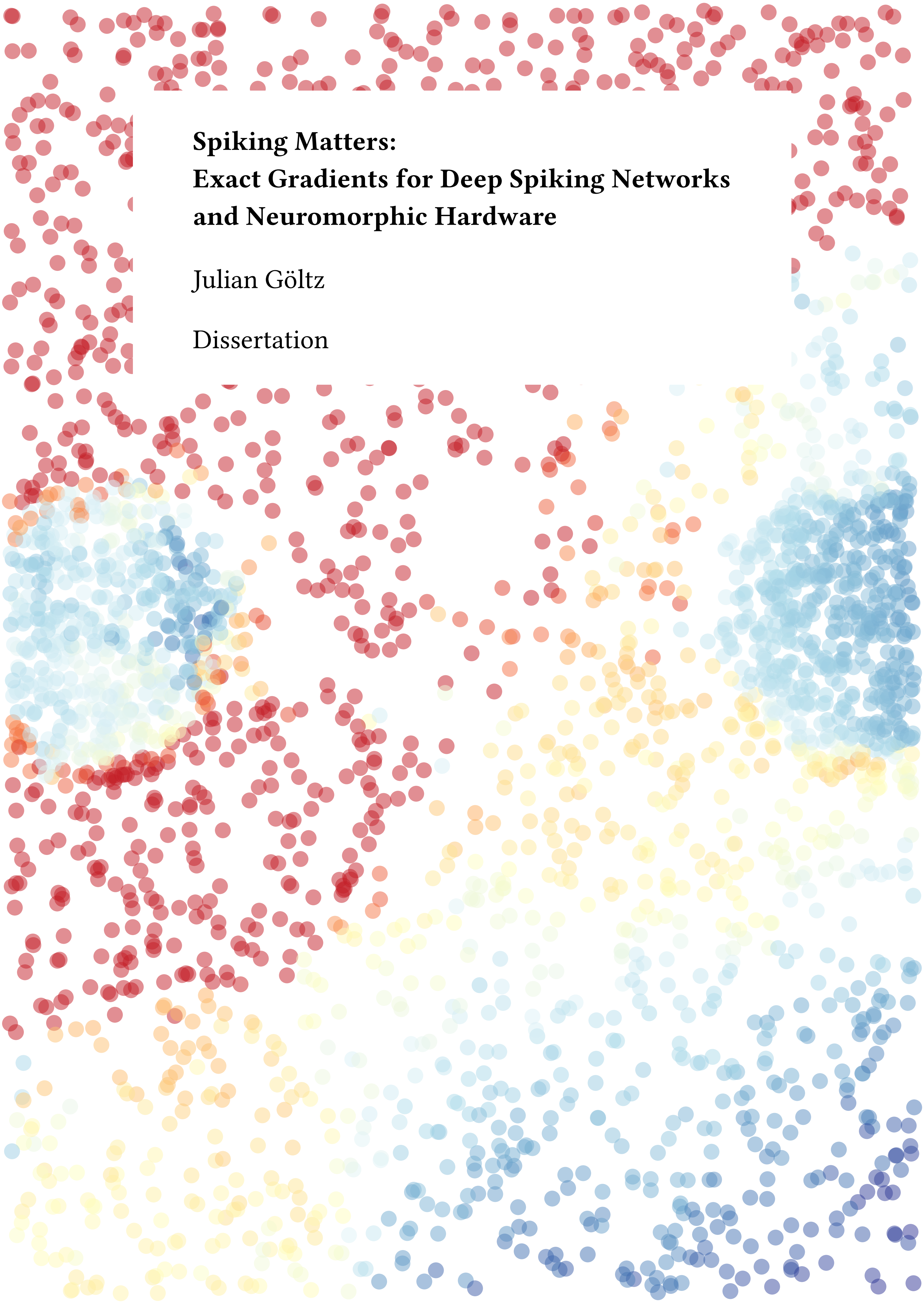

Spiking Matters: Exact Gradients for Deep Spiking Networks and Neuromorphic Hardware

Julian Göltz

Dissertation

Dissertation

submitted to the

Combined Faculty of Mathematics, Engineering and Natural Sciences

of Heidelberg University, Germany

for the degree of

Doctor of Natural Sciences

Put forward by

Julian Göltz

born in Crailsheim, Germany

Oral examination: 9 February 2026

Spiking Matters:

Exact Gradients for Deep Spiking Networks

and Neuromorphic Hardware

Supervisor	Dr. Mihai A. Petrovici
Referees	Prof. Dr. Walter Senn
	Prof. Dr. Manfred Salmhofer

Spiking Matters:
Exact Gradients for Deep Spiking Networks
and Neuromorphic Hardware

Inspired by the brain's unparalleled capacity for intelligent and efficient processing, spiking neural networks offer a transformative path towards energy-efficient computation for machine intelligence, and inspire a new class of neuromorphic, or brain-inspired, hardware. However, training spiking neurons accurately and efficiently has previously posed a major obstacle in this endeavour. This thesis pioneers a solution to the exact training of deep spiking networks.

Our approach is inherently sparse: it is based on analytical equations that enable truly event-based computation, relying only on spike times. Their derivation enables, for the first time, exact error backpropagation in hierarchical networks of leaky integrate-and-fire neurons. To validate our method, we design the Yin-Yang dataset with the specific purpose of isolating true difficulty from mere scale. Using this benchmark we highlight the performance and robustness of our approach. With a time-to-first-spike coding, inspired by information processing in the brain, we demonstrate fast and energy-efficient classification on the neuromorphic system BrainScaleS-2. Because our method harmonises ideally with neuronal transmission delays, it enables the natural co-training of weights and delays in both software and hardware. This allows the exploration of the computational power of delays. Jointly analysing the structure and dynamics of deep networks, we find striking connections to computational models of neuronal sensory integration.

The present work connects studies on the computational properties of networks with experiments on analogue hardware as well as the comparison to the biological archetype. The algorithm's fundamental spiking nature pushes neuromorphic hardware to maximum efficiency. Bridging neuromorphic engineering and neuroscience, our integrated approach not only advances performance and efficiency of machine intelligence, but also drives forward the understanding of fundamental mechanisms behind spatio-temporal processing in the brain.

Lernen mit Spikes:
Exakte Gradienten für Tiefe Spikende Netzwerke
und Neuromorphe Hardware

Inspiriert von der beispiellosen Effizienz des Gehirns in intelligenter Informationsverarbeitung bieten spikende neuronale Netze einen revolutionären Weg zu energieeffizienter, maschineller Intelligenz und inspirieren somit neuartige, neuromorphe Hardware. Allerdings stellte das präzise und effiziente Training spikender Neuronen bislang ein großes Hindernis dar. Die vorliegende Arbeit beschreibt nun eine neue Lösung zum exakten Trainieren tiefer spikender Netzwerke.

Unser Ansatz ist intrinsisch effizient, weil er auf analytischen Gleichungen basiert, die es erlauben, Berechnung auf die kritischen Zeitpunkte zu beschränken. Wir ermöglichen erstmals eine exakte Fehlerrückpropagation in hierarchischen Netzwerken aus leaky Integrate-and-Fire-Neuronen. Um unsere Methode zu validieren, haben wir den Yin-Yang-Datensatz entwickelt, mit dem spezifischen Ziel, wahre Problematik von reiner Größe zu trennen. Anhand dieses Benchmarks zeigen wir die Leistungsfähigkeit und Robustheit unseres Ansatzes auf. Unter Verwendung einer Informationskodierung basierend auf der Zeit bis zum ersten Spike, wie sie im Gehirn auch für zeitkritische Aufgaben verwendet wird, demonstrieren wir eine schnelle und energieeffiziente Klassifizierung auf dem neuromorphen System BrainScaleS-2. Da unsere Methode ideal mit neuronalen Signallaufzeiten harmoniert, ermöglicht es auf natürliche Weise das gleichzeitige Trainieren von Gewichten und Verzögerungen sowohl in Software als auch in Hardware, was die Erforschung der rechnerischen Vorteile von Verzögerungen ermöglicht. In einer Analyse, die Struktur und Dynamik der tiefen Netzwerke kombiniert, finden wir auffällige Verbindungen zu Modellen der sensorischen Integration im Gehirn.

Die vorliegende Arbeit verbindet die Untersuchung der theoretischen Eigenschaften von tiefen spikenden Netzwerken mit Experimenten auf analoger Hardware sowie dem Vergleich mit dem biologischen Vorbild. Fundamental auf Spikes basierend treibt der Algorithmus neuromorphe Hardware zu maximaler Effizienz an. Durch die Verbindung von neuromorpher Technik mit Neurowissenschaften verbessert unser Ansatz nicht nur die Leistung und Effizienz maschineller Intelligenz, sondern fördert auch das Verständnis der grundlegenden Mechanismen hinter der räumlich-zeitlichen Informationsverarbeitung im Gehirn.

Contents

1. Introduction

Die Natur hat schon alle Designfragen beantwortet.[1]

Luigi Colani

As humans, we have always investigated nature, ourselves, and our likeness. In this endeavour, the brain takes a special place as the object of our curiosity, for it is the only system universally acknowledged to be capable of intelligence. Scientists and engineers have a dual motivation for their interest, obtaining *insights into* the brain to understand its functions, limits, and defects, but also gaining *insights from* brains for our technological advancement.

Not just the brain but biology in general has long served as a source of inspiration for engineering. One of the arguably most drastic recent revolutions in science is the CRISPR/Cas system that originates in the bacterial immune system and has expedited progress in life sciences (Ishino et al. 1987; Gasiunas et al. 2012; Jinek et al. 2012). More than 70 years ago, Otto Schmitt coined the term biomimetics for this technique of mimicking biological solutions for challenges we face as humans (Harkness 2002). Incidentally, his studies of signal propagation along nerve fibres lead to his famous invention, the Schmitt trigger (Schmitt 1938), a device that continues to be used in microelectronics to prevent metastable states. Lastly the field of Artificial Intelligence was founded to study the biological, natural intelligence.

Getting insights into the brain is deeply connected to understanding our human nature. In addition to the philosophical pursuit of our identity, it is the question to understand our limitations and potentially even break free of them. More concretely, the understanding of our brain is evidently instrumental for medical treatments of diseases.

Many efforts are pursued in parallel: wet lab neuroscience conducts experiments that lay the factual foundation on which other disciplines build. Computational neuroscience targets the study of models of the brain, including specifically how neurons interact to create cognitive abilities and how learning shapes neuronal dynamics and connections. As already established, research into Artificial Intelligence is directly influenced by our understanding of the brain. Irrespective of the different approach taken, this is even more so for neuromorphic engineering, the endeavour to create brainlike hardware. This thesis is an attempt to create new connections between these directions and study the efficient computation in brain-inspired systems.

It is evident that we are living in a time of ubiquitous Artificial Intelligence (AI). The first incident that brought this to the attention of the general public was the broadly covered match between AlphaGo and Go grandmaster Lee Sedol (Silver et al. 2017). Since then, we have seen increasingly elaborate models in conjunction with more compute power as well as more training data: for example, mastering a variety of games (Silver et al. 2018), achieving unprecedented success in predicting protein folding (Jumper et al. 2021), and significantly advancing natural

[1] 'Nature has already answered all design questions'. Displayed as part of the permanent exhibition in the Pinakothek der Moderne, Munich.

language processing (Devlin et al. 2019). Nowadays, various kinds of AI are readily available, including for face recognition, speech transcription, or chatbots, conducting tasks not deemed feasible mere years ago.

These advances fall into the field of machine learning: traditionally, this field comprised intricate models targeting specific tasks using minimal resources. Machine learning models such as decision trees often allow for the immediate interpretation of the solutions. However, the breakthrough came when these handcrafted models were replaced by large networks, assembled from simple units and structured in sequences of layers. An essential step was using these networks with many layers as generic models that are specialised through efficient training: while the method, called error backpropagation, is conceptionally simple (Linnainmaa 1970; Werbos 1982; Rumelhart et al. 1986), the combination with vast amounts of compute and training data enabled the exploitation of the computational complexity of large networks (Oh and Jung 2004; Krizhevsky et al. 2012). Because of the structure of these deep neural networks (DNNs), this field is often called deep learning (DL).

DNNs consist of artificial neurons that clearly inherit their name from their biological primitives: the units receive inputs, add them up internally, and generate an output based on a function of the internal state (McCulloch and Pitts 1943). In their earliest form, these units relied on plain thresholding and thus communicated binarised values from layer to layer. Today's networks rely on more complex – but crucially still nonlinear – activation functions to increase computational capabilities and facilitate training. The perceptron (Rosenblatt 1958), the first network of such neurons, was meant as a model of perception in the brain: based on known neurophysiology, Rosenblatt tried to find a description of sensing. Later, this was generalised to multilayer perceptrons by Minsky and Papert (1969), a specific form of DNNs still in use today. The discovery of receptive fields in the visual system (Hubel and Wiesel 1959; Hubel and Wiesel 1962) inspired the generalisation to filters for computation (Fukushima 1980) and eventually convolutional neural networks (CNNs) (LeCun et al. 1989; LeCun et al. 1998; Krizhevsky et al. 2012).

In contemporary artificial neural networks (ANNs), biological inspiration is a lot less direct. For example, residual networks are a part of modern architectures (He et al. 2016), featuring so-called skip connections that through bypassing the traditional information flow dramatically facilitate training. Skip connections have been found in, e.g., insect brains (Winding et al. 2023), but only after their addition to DNNs in machine learning. In other cases, the connection to biology is superficial and predominantly present in name: for example, the attention mechanism (Bahdanau et al. 2016) borrows its name from psychology but lacks a direct correspondence to the biological phenomenon. In the form of self-attention this mechanism enabled the transformer architecture (Vaswani et al. 2017) and that, in turn, large language models (LLMs) which have since gained broad popularity (Project Apertus et al. 2025; OpenAI et al. 2024; Gemini Team et al. 2024; DeepSeek-AI et al. 2025).

Similar to how – compared to an average human – a calculator is more efficient at multiplying two large integers, or a search engine more effective at finding a website, an LLM can be more efficient on certain tasks – at least when only considering inference and disregarding the training. This includes peculiar tasks like rewriting a given text in the style of a famous author, or coming up with code for inverting a binary tree (Luccioni et al. 2024). The set of examples for which this is true seems to be increasing. However, for more complex assignments LLMs seem to intrinsically suffer from the problem of hallucinating wrong answers (Ji et al. 2023). Moreover, there is doubt when it comes to actual creativity[2] as well as solid reasoning (Mirzadeh et al. 2025): tasked on definitely new maths problems not seen in the training data, they seem to have difficulties (Petrov et al. 2025; Mahdavi et al. 2025).

What is more, the computational overhead of training these billion-parameter models in the first place by far exceeds the cost of inference. In fact, technological development was a prerequisite for the DL epoch, starting with GPU implementations to keep pace with scaled training demands (Oh and Jung 2004). Most of the recent advances have been brought forward by increasingly staggering computational resources: caused by AI, the International Energy Agency projects the energy consumption of data centres to more than double within the next five years and, in the United States of America in 2030 to consume 'more electricity [than] the production of [all] energy-intensive goods combined' (International Energy Agency 2025). The associated emission of greenhouse gases (Luccioni et al. 2022; Stiefel and Coggan 2023) and the power hunger have made it a topic in mainstream media (Cohen 2024). Especially in light of the approaching climate crisis (IPCC 2023), it seems imperative to reduce this vast energy consumption. Through the need for change and the associated funding, the opportunity arises for more efficient use of computational resources, but crucially also for novel, more efficient hardware (Khan et al. 2018).

From early calculators like the abacus, humans have long used devices for assistance with cognitively challenging problems (Ifrah 2001). An outstanding example is the Antikythera mechanism from the second century BCE that was used to predict lunar and solar eclipses (de Solla Price 1974; Freeth et al. 2006) and had a remarkable complexity, higher than other recovered machines for more than a millennium. In the last century, mechanics has been replaced by electronic circuits enabling analogue computers used for simulation of star trajectories or spacecraft rendezvous (Reynen 1963; Simpson and Smith 1967).

More recently, due to their versatility, digital architectures have provided the backbone of technological development. For a long period, the rapidly increasing transistor density was captured by Moore's law (Moore 1965). Associated with shrinking transistor sizes was a rise in efficiency and speed of computing (Dennard et al. 1974) that has effectively come to a stop around 2006 (Sutter et al. 2005; Bohr 2007) and since then, dedicated methods had to be applied for further improvements. Additionally, Moore's law has at least decelerated due to diminishing returns as the transistor sizes approach atomic scales (Leiserson et al. 2020).

Another limitation of traditional hardware stems from its architecture. The von Neumann principle (von Neumann 1945) enables scalability by compartmentalising memory and processing units, but this design creates efficiency bottlenecks: data needs to be retrieved first. As mitigation, hierarchical caching has been adopted as a common strategy to keeping relevant data close to the processing units. However, memory access still consumes more energy than computing itself in modern architectures (Horowitz 2014; Jouppi et al. 2021). For AI applications, parallelisation on specialised hardware like GPUs or tensor processing units (TPUs) is helping for now, but the trend seems unsustainable (Thompson et al. 2022; Schwartz et al. 2020).

To overcome these challenges and build more powerful AI, we can turn to the brain to learn from its remarkable efficiency (Levy and Calvert 2021). There are multiple central concepts that we can exploit to construct neuromorphic, i.e., more brain-like, computers. In contrast to the von Neumann principle, the brain distributes information processing across nerve cells: a

[2]Definition pending.

neuron and its synapses constitute compute and memory units within one cell, thus implementing so called in-memory computing. Together with the more general principle of distributed computing, it is a central pillar of neuromorphic engineering.

Another concept concerns brainlike communication: neurons mainly communicate by all-or-none signals, fully determined by the time they occur, not unlike a Dirac delta distribution. These spikes or action potentials implement an event-based communication scheme. Furthermore, the temporal sparseness of the communication is understood to be responsible for some of the neuronal efficiency (Olshausen and Field 1996; Koch and Segev 2000; Roy et al. 2019).

As described above, neural tissue is the central location of computation in the brain. The neurons exhibit intricate dynamics and react in a complex fashion when receiving inputs (Beniaguev et al. 2021). An alternative to expensive simulation is to find a system that is governed by the same differential equations which is the crucial idea behind analogue, or physical, computing. In the case of neuromorphic computing, this amounts to substituting biochemistry with electronics, granting high levels of speed and control. Replacing time-discrete numerical integration with time-continuous emulation also promises gains in efficiency, especially when the system can tolerate some levels of imprecision (Boahen 2017; Buhler et al. 2017). A notable example of analogue electronics is the BrainScaleS-2 system developed in Heidelberg (Billaudelle et al. 2020; Pehle et al. 2022) that is employed in this thesis.

Whenever we appeal to inspiration by nature, as done by the leading quotation, it is essential to remember that clearly not all natural designs are efficient or important, that is, worth copying. As an engineering-specific example, the Wright brothers used birds as inspiration to reconstruct flight, but refrained from copying the flapping wings (McDonnell et al. 2014). There are numerous examples of inefficient design in biology, one of the most remarkable being the recurrent laryngeal nerves (Owen 1839): due to evolutionary development, rather than leading from the brain directly to the larynx, they divert around the heart before reaching their target. This detour constitutes clear evidence for gradual evolution (Dawkins 2009). Likewise, for intelligent artificial systems one should keep in mind that when the building of a working machine is the primary goal, copying biology is not necessary.

However, details that are disregarded for engineering can become relevant for understanding biological processes, to treat illness and injury. Nowadays, we have a detailed understanding of the brain at many different levels of abstraction, from the scale of molecules and individual ion channels to high-level descriptions like psychology, integrating the whole brain and body. A well known example is the excitation in nerve cells: Hodgkin and Huxley (1952) described in great mathematical detail a phenomenological model of the dynamics of the cell leading to the action potential. This model has since been continuously corroborated by further analysis of neurons. As another example, based on his basic anatomical and behavioural studies, Papez (1937) proposed a mechanism abstractly explaining a system that controls emotions. Extended to include memory formation and termed Limbic system (MacLean 1949), this construct has been helpful both to explain how emotions interact with memories and to predict behavioural consequences of physiological damage to included brain regions. This relevance arises despite or maybe because of the major abstraction of the model.

Despite the pervasive calls for ever-increasing realism, a useful model has to feature some abstractions. Abbott (2008) beautifully highlights this with a reference to Jorge Borges' map of an empire 'whose size was that of the Empire' that in its detailed realism is completely useless (Borges 1975). Stepping away from the elaborate Hodgkin-Huxley description of neurons grants us the mathematical tractability to analyse populations of neurons, in particular their dynamics when given stimuli but also their learning behaviour. In this thesis, we will mostly rely on the leaky integrate-and-fire (LIF) neuron model (Lapicque 1907) that is ubiquitous in computational neuroscience (Abbott 1999; Brunel and van Rossum 2007a). While it presents a clear abstraction, it captures the most essential properties of neurons like spatio-temporal integration of inputs, time-continuous computation as well as spiking communication. With few adaptations it even provides a good basis to model activity of pyramidal cells (Rauch et al. 2003; Gerstner and Naud 2009; Teeter et al. 2018).

It has been argued that the high performance of ANNs is not a sign of fundamental supremacy but a result of the fortunate fit of their algorithms to GPUs – which were originally developed for the entirely different purpose of processing video graphics. Hooker (2021) called this phenomenon the hardware lottery: the available hardware (and software) determines the success of an algorithm. To achieve further progress, Hooker (2021) is calling for algorithm-hardware co-design with the objective of optimising the hardware, software, and algorithm as a single, highly-connected issue instead of separately. For building efficient hardware for brain-like intelligence, this implies creating a device drawing inspiration from the nervous system.

This reflects calls for increased cooperation and interdisciplinary research from both sides, the DL and the neuroscience communities (Hassabis et al. 2017; Tsodyks 2008). Specifically, Pfeiffer and Pfeil (2018) called for DL methods for spiking neural networks (SNNs) with implementation on neuromorphic substrates, to fully exploit their promised benefits. Early realisations of neuromorphic DL often used a form of spike-count-based conversion of ANNs (Cao et al. 2014; Diehl et al. 2016; Schmitt et al. 2017; Wu et al. 2019). However, the temporal and spatial sparseness are essential for the brain's efficiency, and communication of the individual spikes is crucial for biological neurons (Gerstner 2001; Izhikevich 2004). To replicate the success of ANNs while maintaining efficiency, it seems integral to adopt the training method intrinsic to all DL architectures, namely error backpropagation, while at the same time transcending efficiency limitations of conventional ANNs. Discouraged by the discontinuous time dynamics of some state variables in typical neuron models, a gradient-based approach at first remained elusive. Zenke and Ganguli (2018) designed a method adopting surrogate gradients to tackle this challenge, later refined by Neftci et al. (2019). The training is based on an approximation of the information propagation in SNNs, effectively converting the SNN into a corresponding recurrent neural network (RNN). This approximation makes the network amenable to standard DL practices like backpropagation through time (Werbos 1990) as well as the abundant DL toolchain. Surrogate gradients have proven highly successful, making it the de facto standard for training SNNs. However, the method does not leverage the event-based description of neuronal communication thus creating high computational costs, and it suffers from being inherently based on an approximation (Gygax and Zenke 2025).

Ultimately, the question arises whether exact gradients for spiking neurons exist, whether they can be used to train efficiently, and whether they are robust enough to work with inhomogeneous neuromorphic devices, rendering it possible to train SNNs emulated on a spiking substrate.

Organisation of the thesis

The quest for temporally sparse neural computation and the corresponding exact gradients represents the common theme for the following chapters, and this thesis demonstrates a number of contributions in these directions. To motivate and lay out the foundations for our later findings, we start with a detailed background into neuroscience and computation in the nervous system, computational modelling, deep learning, and neuromorphic engineering (Chapter 2).

The first contribution (Chapter 3) is the design of a new neuromorphic device: Lu.i is a small-scale, tangible PCB that emulates one neuron in analogue electronics. While not directly addressing the research hypothesis, it has proven incredibly valuable when teaching central concepts of neuroscience like time-continuous computation, spatio-temporal integration of inputs, and event-based computation. It also shows the rationale behind physical computation, building a system that emulates our model of interest. The inherent interactivity makes it captivating, leading to worldwide use: more than 1 000 Lu.i PCBs have been shipped, they are used in America, at various universities in Europe, and as part of the TReND-CaMinA summer schools[3] in 2023 in Ghana, 2024 in Rwanda and 2025 in Zambia. In addition, it has been picked up by teachers, and it even found adoption by Mercedes-Benz for outreach on their own efforts towards neuromorphic engineering (see Fig. SI.A.1). The connectivity of the PCBs allows the reproduction of diverse neural circuits in educational miniature versions. One particularly appealing example is an implementation of the ring attractor, a neuronal circuit that tracks heading direction in insects (Pisokas et al. 2020). Lu.i serves as a helpful tool for academic as well as pedagogical teaching by deepening understanding. The manuscript has been published in *Trends in Neuroscience and Education* (Stradmann, Göltz et al. 2025).

The second contribution (Chapter 4) is addressing the necessity for good metrics (Davies 2019; Yik et al. 2025): when assessing the computational complexity of a neural network, the effectiveness of a training algorithm, or the fidelity of neuromorphic hardware, a comparable and expressive test is crucial. From prototype devices to exploratory studies, these use cases are connected by a limitation in either neuronal real estate or computational complexity available for studies, while maintaining the need for precise discrimination. In this context, we developed the Yin-Yang (YY) dataset, addressing these issues. It continues to fulfil its purpose as proven by its wide dissemination and continued use to benchmark biologically plausible learning schemes, neuromorphic chips, or training algorithms for SNNs.[4] The work is published with the *Association for Computing Machinery* (Kriener et al. 2022).

The third contribution (Chapter 5) studies the computation within SNNs in the efficient limit of at most one spike per neuron. In this biologically-relevant activity regime (Thorpe et al. 2001) we found an analytical relation for the LIF neuron model that makes forward propagation simple. Crucially, this also allows the derivation of a training algorithm based on exact

[3]Teaching and Research in Natural Sciences for Development in Africa – Computational Neuroscience and Machine Learning in Africa

[4]At the time of writing, a list of publications that demonstrate their results on the YY dataset includes Wunderlich and Pehle (2021), Müller et al. (2022), Taylor et al. (2022), Lee et al. (2023), Ma et al. (2023), Spilger et al. (2023), Keuninckx et al. (2023), Max et al. (2024), Yousuf et al. (2024), Fontanini et al. (2024), Mészáros et al. (2025), Dold and Petersen (2025), Granier et al. (2025), Neuman et al. (2025), Arnaud Yarga and Wood (2025), Arnold et al. (2025), Béna et al. (2025), Aswani and Jabari (2025), Saponati et al. (2025) and Boccato et al. (2025). This is just a selection of the 46 citations currently found on Google Scholar.
The dataset has been used in many theses, beyond Heidelberg and Bern also at universities in Aalto, Finland; Udine, Italy; Rochester, USA; Sherbrooke, Canada; Manchester, UK.

gradients using only the network's sparse spike times. Together, this enables efficient learning with error backpropagation through deep spiking networks. In a two-pronged strategy, we validate the robustness of our approach. On the one hand we study different modes of disturbances in software simulations, on the other hand we conduct the litmus test of robustness: successful implementation on analogue hardware with its intrinsic imperfections. The in-the-loop technique allows incorporation of hardware deviations by integrating information from the emulated forward pass into the updates, a helpful trick for physical computing. Harmonising with the accelerated, analogue nature of the BrainScaleS-2 system, the sparseness of the activity due to the chosen coding has an outstanding effect: the rapid information propagation, precise learning, and fast emulation work together ultimately enabling extremely fast and energy-efficient classification. This work is published in *Nature Machine Intelligence* (Göltz et al. 2021).

The fourth contribution (Chapter 6) originates from an investigation into the decision process in hierarchical spiking networks. While DNNs are often accepted as a black box, we analyse our neural networks with methods from classical neuroscience. For this, networks with deeper architectures (up to ten layers) and connectivity adhering to more biological constraints are trained, enabling the concrete study of differences between stimulating (excitatory) and suppressing (inhibitory) parts of the network. We find progressive synchronisation and semantisation through the layers, and activity reminiscent of synfire chains (Abeles 1982) that have been studied as a model of cortical computation (Abeles 1991; Diesmann et al. 1999; Tetzlaff et al. 2002; Kumar et al. 2008; Trengove et al. 2012). Furthermore, other concepts known from biology are recovered, such as target-specific excitation and broad inhibition patterns. Because of the biological plausibility of the network structure and activity, the approach even allows comparison to in vitro data of similar stimulations (Reyes 2003; Barral et al. 2019). The work has been submitted for publication in a peer-reviewed journal and at the time of writing is available as a preprint (Oberste-Frielinghaus et al. 2025).

The fifth contribution (Chapter 7) generalises the concept of training. The conventional approach concentrates on the connection strengths as trainable parameters, but there are several additional factors that determine the computation and consequently influence expressivity as well as performance of deep spiking networks. In biological neurons, transmission delays emerge due to the physical time required for signal propagation. While often deliberately excluded from SNN models, delays turn out to have a tremendous impact on network activity (Maass and Schmitt 1999; Izhikevich 2006). Yet, effective exploitation remained underexplored: instead of actively trained, delays were only used via inhomogeneous initialisations (Gerstner et al. 1996; Bohte et al. 2002; D'Agostino et al. 2024). Also, methods based on surrogate gradients lack a natural translation between their discretised activity and delays since those act on spike times themselves. Therefore, additional mechanisms are required that introduce further challenges (Shrestha and Orchard 2018; Hammouamri et al. 2024; Sun et al. 2023). Our approach, termed DelGrad, builds on the investigation of exact spike times and their gradients to directly operate on the time axis. This grants a flexibility that enabled the first systematic study of different types of delays, proving the benefit of co-optimising weights and delays and, in addition, allowed the successful on-chip implementation on BrainScaleS-2. This work is published in *Nature Communications* (Göltz et al. 2025).

2. Background

To provide the reader with the necessary information to follow the novel contributions presented in Chapters 3 to 7, in this section basic concepts of nervous systems, deep learning, and neuromorphic hardware are introduced.

2.1. Biological brains

We start with a high-level description of the structure of our nervous system before describing its fundamental units, the nerve cells. Beginning with their morphology we continue with their resting state and, subsequently, their excitation. The mechanism associated with the excitation is central to information processing in the brain, as it allows neurons to perform spatio-temporal integration of received inputs, and furthermore, sparse communications via spikes.

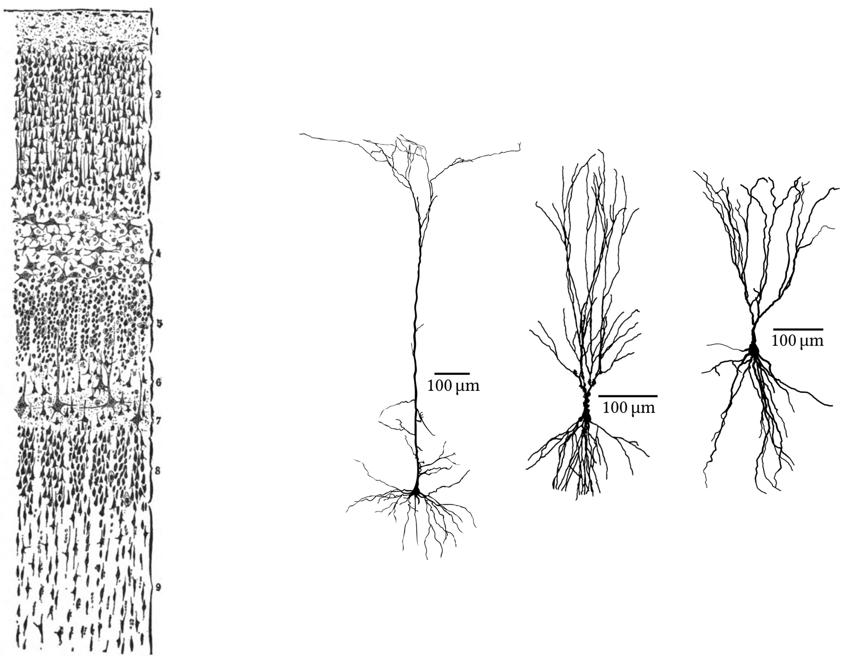

Figure 2.1.: Morphology of nerve cells. *Left:* Drawing of a stained section of human visual cortex. In this drawing by Ramón y Cajal, the layered structure as well as the neuron diversity is visible. Image taken from Ramón y Cajal (1899). *Right:* Examples of structured pyramidal cells of different rats and mice.[5]

[5]Reused from 'Pyramidal Neurons'. Bekkers, Elsevier (2011) with permission from Elsevier, as well as 'Distribution of thorny excrescences on CA3 pyramidal neurons in the rat hippocampus'. Gonzales et al., Wiley (2001) with permission from Wiley.

We elaborate on the propagation and interpretation of these spiking signals, and conclude with remarks on learning in the brain.

Contrary to ancient times when the mind was thought to reside in the centre of the body, specifically the heart (Lind 2006), we now understand the brain to be the central organ guiding our comprehension of the environment and our actions in this world. Together with the spinal cord, the brain forms the central nervous system (CNS) that in turn is connected to organs and muscles via the peripheral nervous system (PNS). The main function of the latter is to transport sensory stimuli across ganglia to the CNS and to relay motor commands to muscles. Due to their direct connection, retinal and olfactory sensory inputs are considered part of the CNS. The location of information integration and processing is the CNS, and as such it will be the focus in the following.

In the CNS, the cells responsible for the majority of computation are called nerve cells or neurons,[6] of which there are about 100 billion in the human brain (Williams and Herrup 1988). Perhaps unsurprisingly, the absolute size of brains, that is, their volume, correlates with cognitive ability (Deaner et al. 2007). However, this relation only holds when compared within a species (Herculano-Houzel 2009), and for a primate of our size, the estimated number of neurons of a human is not exceptional (Herculano-Houzel 2012).

While neurons are continually generated in mammalian brains in a limited fashion (Spalding et al. 2013; Varadarajan et al. 2022), the lifetime of an individual neuron is surprisingly long. In fact, when transplanting neurons across species into longer-lived animals, the lifespan of neurons can exceed that of the donor (Magrassi et al. 2013). Ageing processes in humans neither reduce the number of neurons nor change basic biophysical properties (Burke and Barnes 2006). In contrast, cognitive decline associated with ageing is hypothesised to be related to reduced functionality of synapses (Yankner et al. 2008).

Because only an overview can be given here, for more details the author refers to standard textbooks like Dayan and Abbott (2001) and Gerstner et al. (2014), or to the slightly more elaborate overview in Petrovici (2016).

Structure of a neuron Figure 2.1 shows examples of nerve cells to visualise their morphology. Like other types of cells, neurons have a cell body, called soma, that is separated from the extracellular fluid by a cell membrane made up of a lipid bilayer (Fig. 2.2). Connected to the soma are the dendrites which usually branch out in the proximity of the soma, forming a structure called dendritic tree. While there can be many dendrites, each neuron has only one axon which interfaces the soma at the axon hillock. Depending on brain area and neuron type, axon lengths and branching behaviour vary, but ultimately the axons grow into axon terminals that create connections to other neurons. The axons, therefore, are responsible for transmitting signals to other neurons, which collect these inputs in their dendritic tree. A large part of the computation on these inputs happens in the soma.

We will shortly introduce abstractions and generalisations of neurons, but it should be noted that there exists a vast diversity of neuron types (e.g., Billeh et al. 2020; BRAIN Initiative Cell Census Network (BICCN) et al. 2021; Zeng 2022). To what degree this is an artefact of biological evolution or whether morphological (and physiological) diversity serves a specific purpose remains under exploration, with theoretical models trying to evaluate function based on diversity (Maass 2016; Granier et al. 2025).

[6]Despite a common misconception, the correct spelling is neuron/neurons, as opposed to ‘neurone/neurones’, even in British English (McMenemey 1963; Mehta et al. 2019).

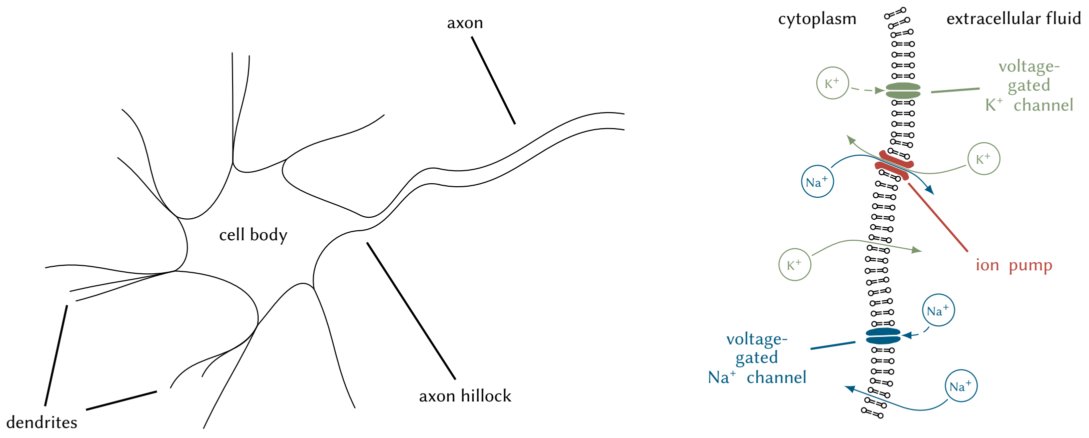

Figure 2.2.: Sketch of the neuronal structure. *Left:* Neurons comprise a cell body (soma), a dendritic tree, and an axon which is connected to the soma via the axon hillock. *Right:* The cell membrane consists of a lipid bilayer (black) that works as an electric insulator for the cell. Locally, however, it is permeable for ions via ion leak channels, voltage-gated ion channels (green/blue), and ion pumps (red). Adapted from Billaudelle (2022).

Resting state Crucial for the physiological state of a neuron is its electric potential, or more precisely, the ion concentrations in the cell: because the membrane is itself impermeable to ions, the imbalance in ion concentrations following the synaptic input is sustained, and a post-synaptic potential (PSP) decays only slowly over time. The eventual decay is driven by the leak channels, while the gradient of the ion concentrations between inside and outside of the cell is maintained by ion pumps. The most important ion pump uses energy in the form of adenosine triphosphate (ATP) to transport two potassium ions (K^+) into and three sodium ions (Na^+) out of the cell. The electric potential in equilibrium is called resting (or leak) potential. To quantify this electric potential, it is helpful to first look at the individual ions: the ion-specific reversal potential can be calculated with the Nernst equation. For example, the sodium potential E_{Na^+} depends on the external and internal Na^+ ion concentration $[Na^+]_{ext/int}$ like

$$E_{Na^+} = \frac{RT}{F z_{Na^+}} \ln \frac{[Na^+]_{ext}}{[Na^+]_{int}} \,, \tag{2.1}$$

with universal gas constant R, temperature T, Faraday constant F, and valency $z_{Na^+} = 1$. All ions permeating the membrane influence the overall leak potential E_ℓ jointly. This is described by the Goldman-Hodgkin-Katz equation, and the typical equilibrium state of a neuron is around $E_\ell = -65\,\mathrm{mV}$.

Synapses and the excitation of a neuron The main communication between neurons occurs in the shape of neurotransmitters exchanged at synapses, formed between an axonal bouton at the pre- and a dendritic spine at the postsynaptic neuron (Fig. 2.3). The neurotransmitters are essential for the communication and come with great diversity but generally either have a stimulative (excitatory) or suppressive (inhibitory) effect. Because a neuron uses the same set of neurotransmitters at all synapses, they can be used to identify a neuron: for example, the most widespread inhibiting (gamma-aminobutyric acid (GABA)) and exciting (glutamic acid) neurotransmitters are the source of the terms GABAergic and glutamatergic for the respective

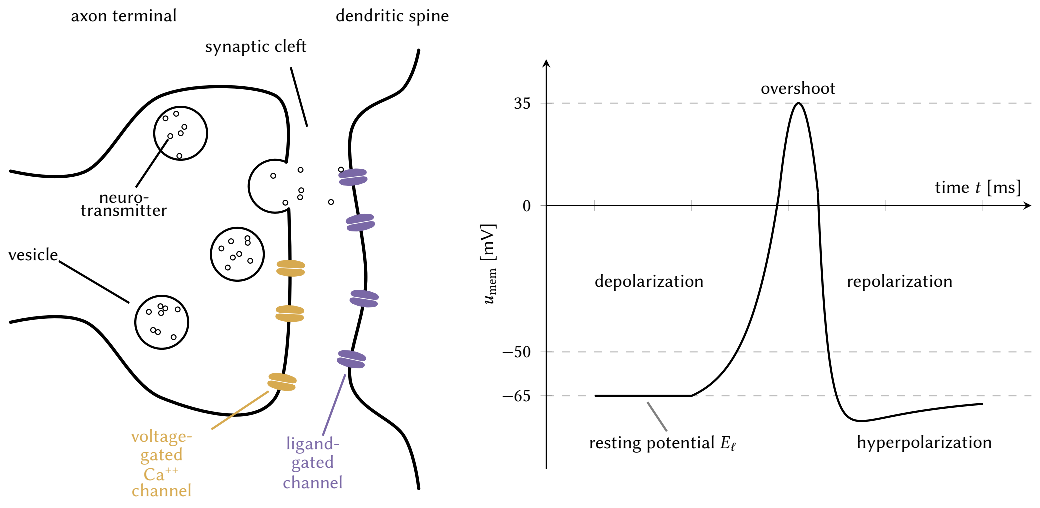

Figure 2.3.: Excitation in a neuron. *Left:* Sketch of a synapse, forming between an axon terminal and a spine on a dendrite. The former contains voltage-gated channels that upon arrival of a signal encourage the release of neurotransmitters which, during resting periods, are contained in vesicles. Release of neurotransmitters occurs when a vesicle merges with the cell membrane, resulting in their distribution into the extracellular medium. Diffusion through the synaptic cleft allows the activation of ligand-gated ion channels of the postsynaptic neuron, altering its membrane voltage. *Right:* Membrane dynamics following a sufficiently strong stimulus: first, the membrane rises rapidly through the depolarisation, even overshooting to reach a positive potential. Afterwards, the neuron repolarises, with a short period of hyperpolarisation that is associated with a temporary refractoriness.

neurons. The concept that a neuron uses the same neurotransmitters at all synapses is known as Dale's principle (Dale 1935), coined by John Eccles (1954).

The neurotransmitters are kept in vesicles in the synapse. When a signal from the soma arrives at the bouton, the ensuing influx of calcium ions due to the change of the local membrane potential leads to the release of neurotransmitters into the synaptic cleft, the space between the presynaptic and postsynaptic cell. Once the neurotransmitters are in the extracellular medium, diffusion brings them in reach of the postsynaptic membrane where they can bind to ligand-gated ion channels. These channels enable the flux of ions through the membrane, in turn leading to a change in the membrane voltage, called a PSP. This local potential spreads through the cell to unfold its effect on the whole neuron.

Origin of an action potential The two classes of neurotransmitters have opposite effects: excitatory inputs lead to a depolarisation, that is, the membrane potential u_{mem} increases. Conversely, inhibitory inputs cause hyperpolarisation, i.e., a decrease of the voltage. In the soma, inputs from all presynaptic neurons come together. When the activity of multiple presynaptic neurons occurs in close temporal proximity, a sufficient local excitation (exceeding $\Delta u_{\mathrm{mem}} \approx 15\,\mathrm{mV}$) triggers a spike or action potential (Fig. 2.3). The underlying mechanism starts when the excitation activates the voltage-gated ion channels: as the permeability of these channels depends on the membrane voltage, they begin to govern the dynamics. First, the sodium channels open, leading to an influx of Na^+ ions into the cell. The resulting depolarisation opens the Na^+ channels further, a self-reinforcing effect. Next, the potassium channels are activated, allowing the K^+ ions to leave the cell, countering the rise of the membrane potential. Simultaneously, the sodium channels are deactivated, that is, completely stopping the sodium flow, while the continued flow of potassium ions repolarises the cell, even leading to a hyperpolarisation. Afterwards, the neuron slowly returns to resting conditions while the sodium channels exit their state of inactivation. The change of states of the voltage-gated sodium channels is typically understood via the so-called ball-and-chain inactivation model.[7] While the neuron is hyperpolarised, it is harder or even impossible to elicit another spike regardless of the level of excitatory input. A neuron in this state is said to be in the refractory period.

During such a sodium spike, the rapid increase of the membrane potential by about $\Delta u_{\text{mem}} = 100\,\text{mV}$ within few milliseconds is quite stereotypical, i.e., nearly independent of the form or size of the input, across different neurons and brain regions. However, it relies on the correct operation of the mechanism as described: a manipulation via local anaesthetics, for example, works by blocking sodium channels, thereby preventing the formation of action potentials and, consequently, the communication of neuronal signals like pain. Apart from these stereotypical spikes, there are other types of action potentials like NMDA or calcium spikes (Larkum et al. 1999; Palmer et al. 2014; Antic et al. 2010) that are hypothesised to play a role in brain functions (Aru et al. 2020).

Signals between neurons The axon hillock (Fig. 2.2) is especially excitable, facilitating the generation of action potentials. Once an action potential appears in the axon hillock, some excitation is propagated backwards through the soma to the dendritic tree, but the largest part of the signal is transported through the axon. The axons have a distribution of diameters that range from $0.16\,\mu\text{m}$ to $0.9\,\mu\text{m}$ (Liu and Schumann 2014; Liewald et al. 2014) and can vary significantly in length: in humans, the longest axons comprising the sciatic nerve travelling from the lumbar vertebra to the foot (Schünke et al. 2012) can exceed 1 m. In the mammalian CNS, axonal lengths are distributed across a wide spectrum with axons of pyramidal neurons reaching up to 1 m length (Ropireddy et al. 2010) and those of interneurons being much shorter, on the order of dozens of millimetres (Hu et al. 2014).

Transmission delays The communication time between neurons is of utmost importance for the reaction time of animals. Considering the axon as a cable, reducing the resistance by expanding the diameter of the axon increases the speed of propagation of the signal. This mechanism evolved in invertebrates to enable quick communication across long distances, with the most famous example being the squid giant axon, investigated by Hodgkin and Huxley (1952).

Another mechanism found by evolution around 425 million years ago is called *myelination*. The resulting faster and more efficient signalling across myelinated axons[8] is considered necessary or at least advantageous to allow the formation of the complex structures of CNS and PNS

[7]The details of this model are not relevant for this thesis, but it might be interesting that the suggestiveness of the model name is a good description of the biophysical basis: while the original hypothesis by Armstrong and Bezanilla (1977) used other language, it already included the idea of a ball blocking the channel. The balls consist of amino acids, microscopic description of structure of gate including renderings of the proteins have been done (Zhou et al. 2001) as well as imaging experiments showing the structure of the ball (Bentrop et al. 2001). Aldrich (2001) includes a detailed sketch.

[8]To highlight the diversity in biology, it is noted that while myelination typically occurs on axons, there are exceptions: the pseudounipolar neurons can have dendrites of extraordinary length on the order of 1 m that are also myelinated (Schünke et al. 2012, page 283).

in vertebrates (Zalc et al. 2008; Zalc 2016). The layers of myelin sheath wrapped around the axon are constructed by oligodendrocytes (OLs)[9] (Philips and Rothstein 2017) that also provide the metabolic support to the neurons. A mature OL can create and maintain more than 100 myelin sheaths on different axons (Xin and Chan 2020). The exact process of the myelination has only recently been found (Snaidero et al. 2014) in an ambitious experiment on zebrafish: the sheath itself continues to wrap around the axon in a helical motion, subsequently creating the visible layers of myelin. In between the sheathed segments the axon is in contact with the extracellular fluid at places called Ranvier nodes. This allows the excitation to jump from node to node, enabling the so-called saltatory conduction.[10]

For myelinated axons, in addition to factors like temperature, morphological properties like the diameter, myelin thickness and distance between Ranvier nodes control the velocity of signals (Waxman 1980). In mice, blocking the Myelin regulatory factor (Myrf) to prevent new myelination via OLs inhibited the mice to learn a new motor skill, strongly suggesting the importance of changes in the myelination for learning (McKenzie et al. 2014). Furthermore, following the learning of a new motor skill, changes in parts of the CNS consisting mainly of myelinated axons (called the white matter) have been observed (Sampaio-Baptista et al. 2013). Likewise in humans, there is some evidence that complex visuomotor tasks like piano playing or juggling has an impact on the myelination in the white matter (Bengtsson et al. 2005; Scholz et al. 2009).

Neural coding Understanding the brain rests on understanding its processes, presumably on the level of neurons. We have established that neurons mainly communicate with spikes, but how these spikes are connected to sensory stimuli or even high-level concepts like thoughts is a focus of ongoing research. This connection between information and activity is called coding. Some general concepts have been described, like the efficient coding hypothesis, arguing that neurons tend to minimise the redundancy in their encoding (Attneave 1954; Barlow 1961).

The most prominent coding originates from the context of muscles (Adrian and Zotterman 1926; Adrian 1928) and is called rate coding: the idea is that the information is encoded in the *average* activity, either average over time, space (i.e., a population of neurons), or trials. When a rate code is formed over many trials, it introduces large stochasticity to an individual trial. When precision is required, averaging over time or space comes at the cost of long integration times or the need for large neuron populations, respectively. Despite these drawbacks, there is clear experimental evidence that average activity strongly correlate with some presented information, also beyond muscle tissue. In neuroscience experiments, spike rates or derived quantities of average activity are often the go-to measure.

An alternative approach is to view (relative) timing of spikes as the carrier of information: for

[9]The equally well-known Schwann cells fulfil the same function, however for the PNS and for us the focus lies on the CNS.

[10]Interestingly, Ranvier (1878) compared the myelinated axons to transatlantic cables:

> The myelin has potentially another role: it is probably an isolating sheath. Electrical wires immersed in a conductive medium need to be protected from this medium by a non-conductive sheath; it is on this principle that transatlantic cables are built.

As referenced by (Zalc 2016) from the original:

> La myéline a peut-être encore un autre rôle; elle est probablement une enveloppe isolatrice. Vous savez que les fils électriques qui sont plongés dans un milieu conducteur doivent être isolés de ce milieu par une enveloppe non conductrice; c'est sur ce principe que repose la construction des câbles sous-marins.

auditory localisation, barn owls depend on precise timing of spikes (Carr and Konishi 1990), and in auditory systems generally first spikes appear important (Heil 2004). Precise timing and synchronisation also play a role in olfaction (Hopfield 1995; Brody and Hopfield 2003; Vickers et al. 2001). Rat experiments show that for whisker localisation 'first post-stimulus spike' is decisive (Panzeri et al. 2001; Petersen et al. 2001). Berry et al. (1997) demonstrated that retinal cells in two animal species are better described with discrete spikes as opposed to rate codes. For rapid visual procession within 150 ms in humans (Antal et al. 2000; Thorpe et al. 1996), codes based on spike timings have been proposed (Thorpe et al. 2001; Reich et al. 2001). Similarly, there is evidence of spike time coding of tactile information from our fingertips where information propagates faster than possible with rate codes (Johansson and Birznieks 2004).

The only definite statement to be made is that in the brain, a lot of codes exist seemingly in parallel, clear rules are yet to be defined, and there is no undisputed interpretation. In this thesis, we deliberately design networks in order to train them on a task. For this approach, we must impose a coding both for encoding inputs and decoding the output spike train of a network to calculate a classification accuracy or loss function. Inspired by the timing and energy benefits associated with sparse spiking (Lennie 2003), a time-to-first-spike (TTFS) encoding as well as decoding is employed in our experiments.

Plasticity After a synaptic event, the neurotransmitters are resorbed by the presynaptic cell into vesicles to be used for further activity. As this is a delayed process, the momentary availability of vesicles influences the size of the PSP. This form of short-term plasticity plays a role on a range of milliseconds to a few minutes (Tsodyks and Markram 1997; Markram and Tsodyks 1996; Thomson and Deuchars 1994). The process of changing the effectiveness of connections between neurons is termed plasticity and has different aspects. In addition to the short-term plasticity described above, there is structural plasticity: a share of connections between dendrites and axons form and perish dynamically over time (Trachtenberg et al. 2002; Holtmaat et al. 2005). This has inspired algorithmic realisations where, at any given time, only a subset of connections between neurons are possible, and the subset changes over time (Bellec et al. 2018; Billaudelle et al. 2021).

The plasticity most important for us, however, is long-term changes of synaptic strengths, enabling meaningful neural computation and thought to be at the root of what we call learning. Physiologically, this can be implemented by physical changes of the geometry of the dendritic spine, in turn influencing the synaptic efficacy (Matsuzaki et al. 2001), as well as through how the postsynaptic cell reacts to inputs (Lüscher et al. 2000; Gaiarsa et al. 2002). In addition, the vesicle release is stochastic, and its probability can be influenced, adapting the effective strength of a synapse (del Castillo and Katz 1954; Branco and Staras 2009; Schug et al. 2021).

Hebb (1949) described a phenomenological theory of learning that is commonly summarised as 'What fires together, wires together'. The idea behind the theory is that pairs of neurons with axons and dendrites in physical proximity detect and react positively to causal, temporally close activity, thus deciphering the spatio-temporal correlations in the inputs received from the environment. Learning paradigms, typically called learning rules, that fall in this category include (Sejnowski 1977; Oja 1982; Bienenstock et al. 1982). While Hebb did not formulate his theory explicitly in terms of spike times, it clearly laid the groundwork for spike-timing-dependent plasticity (STDP). This form of plasticity has been corroborated by experiments, showing a clear dependence of synaptic changes on the time difference between spikes (Markram et al. 1997; Bi and Poo 1998). The initial experiment showed an exponential and causal behaviour (connections with pre-spikes before post-spikes are strengthened), since then a lot more diverse STDP curves have been observed (Feldman 2012; Andrade-Talavera et al. 2023). Hebb's theory has also been extended to reward-based reinforcement learning (Frémaux et al. 2010; Frémaux and Gerstner 2016).

2.2. Modelling neural networks

After this phenomenological description of biology, we can tackle the theoretical characterisation of the brain as it will be used in the following. Even single neurons show such complicated behaviour that, e.g., Beniaguev et al. (2021) argue that one needs a deep CNN to model a single pyramidal neuron. However, a more abstract description is sufficient for our studies, and we will mostly stick with point neuron models: these are models that discard the physical structure of the neuron and instead describe the entire neuron, including soma and dendritic tree, as a single point. When spikes or currents are given as inputs, the dynamics of one or more local variables, such as the membrane potential, unfold. Potentially occurring spikes are then registered and communicated to the surrounding neurons.

Hodgkin-Huxley model The arguable gold standard for physiological models of neurons describes the membrane dynamics on the level of ion channels, deduced in experiments on the squid giant axon by Hodgkin and Huxley (1952). Controlling the squid's escape mechanism with water jet propulsion, the axon requires fast signal transmission and since invertebrates lack the mechanism of myelination this explains the large axon diameter that was convenient for the experiments.

In this model, the temporal evolution of the membrane $u_{\mathrm{mem}}(t)$ is determined by the currents influencing it: current through the ion channels Na^{+} as well as K^{+}, current through the leak, and external current I_{ext}. Naming the capacitance of the cell membrane C_{mem}, the dynamics follow from $C_{\mathrm{mem}}\dot{u}_{\mathrm{mem}} = \sum I$, more specifically:

$$C_{\mathrm{mem}} \frac{\mathrm{d}u_{\mathrm{mem}}}{\mathrm{d}t} = -g_{\mathrm{Na}} m^3 h \left(u_{\mathrm{mem}} - E_{\mathrm{Na}}\right) - g_{\mathrm{K}} n^4 \left(u_{\mathrm{mem}} - E_{\mathrm{K}}\right) - g_{\ell} \left(u_{\mathrm{mem}} - E_{\ell}\right) + I_{\mathrm{ext}}(t) \,, \tag{2.2}$$

with conductances g_{Na}, g_{K}, and g_{ℓ}. The opening and closing of the channels is modelled by the variables m, n, and h, which each evolve as a function of u_{mem} between 0 and 1, representing fully closed or open, respectively. Based on phenomenological functions α_x and β_x, they evolve according to

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \alpha_x(u_{\mathrm{mem}}) \cdot (1 - x) - \beta_x(u_{\mathrm{mem}}) \cdot x \,, \quad \text{for } x \in \{m,\, n,\, h\} \,. \tag{2.3}$$

Even without knowing the microscopic details of the ion channels, Hodgkin and Huxley found a faithful description of the membrane evolution and the dynamics of an action potential: Equations (2.2) and (2.3) cover cases of small excitation, typically called *subthreshold dynamics*, the action potential with activation and inactivation of channels, as well as the repolarisation and subsequent refractoriness. To reach this fidelity, all four state variables u_{mem}, m, n, and h are required as dynamic variables. This, however, creates room for simplifications, starting with the action potentials. Just like in biology, the spikes of the model are mostly stereotypical, i.e., all-or-none phenomena. This invites abstracting them away, leaving the modelling of the subthreshold dynamics.

Leaky integrate-and-fire neuron One such approach predates the Hodgkin-Huxley model: Louis Lapicque (1907) is often attributed with outlining the LIF model (Abbott 1999).[11] Below a fixed threshold ϑ, the membrane can be excited or inhibited by a synaptic I_{syn} or an external current I_{ext} according to the following ordinary differential equation (ODE):

$$C_{\text{mem}} \frac{\mathrm{d}u_{\text{mem}}}{\mathrm{d}t} = g_\ell \left(E_\ell - u_{\text{mem}}\right) + I_{\text{syn}}(t) + I_{\text{ext}}(t) \ . \tag{2.4}$$

Without input, the leak $\dot{u}_{\text{mem}} \propto (E_\ell - u_{\text{mem}})$ results in an exponential return to the leak voltage E_ℓ with time constant $\tau_{\text{mem}} = C_{\text{mem}}/g_\ell$. Notably, while I_{syn} might be a function of the voltage, depending on the model of synaptic interaction, there is just one dynamical variable u_{mem}. When the membrane voltage reaches the threshold, the neuron is considered to have spiked at that time,

$$u_{\text{mem}}(t_{\text{sp}}) = \vartheta \ , \tag{2.5}$$

and the neuron enters the refractory state for a period τ_{ref}, where the voltage is clamped to the reset potential V_{reset}:

$$u_{\text{mem}}(t) = V_{\text{reset}} \quad \forall t \in (t_{\text{sp}}, t_{\text{sp}} + \tau_{\text{ref}}] \ . \tag{2.6}$$

The justification for this model is the stereotypical consistency of biological action potentials as short-acting, high-impact pulses. In this manner, they are represented by δ distributions since their temporal extension is shorter than the typical membrane dynamics. The collection of spikes of a neuron is called its spike train:

$$\rho(t) = \sum_{\text{spikes } s} \delta(t - t_s) \ . \tag{2.7}$$

Due to its relative simplicity, the LIF model lends itself to theoretical analysis: the ODE is solvable for constant external input in the absence of synaptic activity, yielding an exponential decay from the initial value to the shifted leak potential $E_\ell + I/g_\ell$. For a current exceeding the threshold, the dependence of the spike rate ν on the input (called f-I curve) is

$$\nu(I) = \frac{1}{\tau_{\text{ref}} + \tau_{\text{mem}} \log\left(\frac{V_{\text{reset}} - E_\ell - I/g_\ell}{\vartheta - E_\ell - I/g_\ell}\right)} \ . \tag{2.8}$$

For large inputs, this rate approaches the inverse of the refractory time τ_{ref} (or approximates a linear function for vanishing τ_{ref}). In the context of deep learning, the relation between input and output is known as activation function: there, rectifying functions (with vanishing output below a threshold) like the rectifying linear unit (ReLU), as well as boundedness are sometime motivated with a reference to firing rates in biology (Householder 1941; Hahnloser et al. 2000).

Synapse models Looking at synaptic inputs, each spike creates a change in the membrane potential, the so called PSP. So called current-based (CuBa) synapses model this change in u_{mem} as a current $I_{\text{syn}}(t) = w \cdot \kappa(t - t_{\text{sp}})$ depending on a synaptic strength w and an interaction

[11] In a paper accompanying the translation of the original paper (Brunel and van Rossum 2007b) to honour its 100th anniversary, Brunel and van Rossum (2007a) clarify that Lapicque was mainly interested in excitation, not in the spike mechanism. Due to the limited knowledge of his time, neither that mechanism nor any reset of the membrane were known. The LIF model including the mandatory reset was coined and studied much later.

kernel κ. Several choices are available, like a δ distribution (creating a sudden jump in the voltage), or a rectangle function $\kappa(t) = \Theta(t)\Theta(\tau - t)$ with the Heaviside step function Θ. The Heaviside function is 1 for arguments larger than 0, and 0 otherwise. In accordance with the finite temporal extent of usual synaptic interactions, a common choice is an exponential decay with synaptic time constant τ_syn:

$$\kappa(t) = \Theta(t) \exp\left(-\frac{t}{\tau_\mathrm{syn}}\right) . \tag{2.9}$$

The Heaviside theta function ensures causality: a spike can only affect a neuron after it has occurred, not before. Continuing this thought, a transmission delay d of an axon carrying a spike t_sp could be introduced as

$$I_\mathrm{syn}(t) = w \cdot \kappa(t - (t_\mathrm{sp} + d)) , \tag{2.10}$$

essentially shifting the spike time. The effect of multiple presynaptic neurons and all their spikes, respectively, add up to the total synaptic current

$$I_\mathrm{syn}(t) = \sum_{\text{neurons } i} \sum_{\text{spikes } s} w_i \cdot \kappa(t - t_s) . \tag{2.11}$$

Interestingly, for some synapse models, there is a closed-form solution for the subthreshold dynamics of the voltage, specifically for exponential synapses,

$$u_\mathrm{mem}(t) = \sum_{\substack{\text{neurons } i \\ \text{spikes } s}} \Theta(t - t_s) \frac{w_i}{g_\ell} \frac{\tau_\mathrm{syn}}{\tau_\mathrm{mem} - \tau_\mathrm{syn}} \left[\exp\left(-\frac{t - t_s}{\tau_\mathrm{mem}}\right) - \exp\left(-\frac{t - t_s}{\tau_\mathrm{syn}}\right) \right] . \tag{2.12}$$

For the special case of equal time constants $\tau_\mathrm{mem} = \tau_\mathrm{syn}$, the voltage is not a difference of exponential as above but a sum of α-kernels of the form $t \exp(-t)$. According to l'Hôpital's rule in the limit $\tau_\mathrm{mem} \to \tau_\mathrm{syn}$, the dynamics are described by

$$u_\mathrm{mem}(t) = \sum_{\substack{\text{neurons } i \\ \text{spikes } s}} \Theta(t - t_s) \frac{w_i}{g_\ell} \frac{t - t_s}{\tau_\mathrm{syn}} \exp\left(-\frac{t - t_s}{\tau_\mathrm{syn}}\right) . \tag{2.13}$$

The conductance-based (CoBa) model is slightly closer to biological dynamics. Here, an input spike does not lead to a current directly, but changes the conductance of a channel, itself causing a current dependent on the distance of the membrane voltage to the reversal potential of this channel. For one spike at t_sp the current amounts to

$$I_\mathrm{syn}(t) = w \cdot \kappa(t - t_\mathrm{sp})(E_\mathrm{rev} - u_\mathrm{mem}) . \tag{2.14}$$

Often, the cases of large positive and negative reversal potential are treated separately, then consequently called excitatory E_exc and inhibitory E_inh reversal potentials. The additional dependence prevents a closed-form solution for the membrane voltage, restricting a lot of the theoretical analyses.

Other models A further simplification is sometimes made: the leak term in Equation (2.4) causes the neuron to forget past input over time, without this term synaptic inputs are retained until a spike is elicited. The subthreshold dynamics of the corresponding non-leaky integrate-

and-fire (nLIF) model are defined by

$$C_{\text{mem}} \frac{\mathrm{d}u_{\text{mem}}}{\mathrm{d}t} = I_{\text{syn}}(t) + I_{\text{ext}}(t) \ , \tag{2.15}$$

and its simplicity makes it a welcome target for theoretical studies (e.g., Mostafa 2018; Dold and Petersen 2025).

Within theoretical neuroscience, many more neuron models have been developed. Each model comes with certain peculiarities like the quadratic integrate-and-fire (qLIF) model. As the name implies, its membrane dynamics are governed by a quadratic term $\tau \dot{u}_{\text{mem}} \propto a(u_{\text{mem}} - E_\ell)(u_{\text{mem}} - \vartheta_{\text{soft}})$ (Ermentrout 1996; Latham et al. 2000). Here, a soft threshold ϑ_{soft} is introduced, that does not immediately imply a spike, but creates strong amplification increasing the membrane potential.

The soft threshold also appears in a more detailed model, the adaptive exponential leaky integrate-and-fire (AdEx) neuron, addressing a crucial phenomenon in biological neurons that the LIF model lacks: when a neuron is exposed to a constant input, it will typically not show a stable firing rate. On the contrary, different responses exist like bursting (rapid firing of spikes close together in time), a changing rate, or single, possibly delayed, spikes. Brette and Gerstner (2005) proposed to add a new dynamical variable, the adaptation current I_a, itself a leaky integrator:

$$\tau_\text{a} \frac{\mathrm{d}I_\text{a}}{\mathrm{d}t} = -I_\text{a} + a(u_{\text{mem}} - E_\ell) \ , \tag{2.16}$$

parametrised with a time constant τ_a and scaling factor a. This variable is driven by deviations of the membrane voltage u_{mem} from the leak voltage E_ℓ, and operates as an additional current on the membrane voltage:

$$C_{\text{mem}} \frac{\mathrm{d}u_{\text{mem}}}{\mathrm{d}t} = -g_\ell(u_{\text{mem}} - E_\ell) + g_\ell \Delta_T \exp\left(\frac{u_{\text{mem}} - \vartheta_{\text{soft}}}{\Delta_T} \right) - I_\text{a}(t) + I_{\text{syn}}(t) \ . \tag{2.17}$$

Furthermore, there is a term similar to what we have seen above: as soon as the soft threshold ϑ_{soft} is crossed, the membrane is exponentially driven up. The spike happens when the hard threshold is crossed, followed by a reset phase like in the LIF model (Eq. (2.6)). At those times, the spike-triggered adaptation changes the value of I_a by an amount b,

$$I_\text{a} \to I_\text{a} + b \ . \tag{2.18}$$

The neuron parameters can be tuned allowing the new mechanisms to enable 'initial bursting, regularly bursting, tonic spiking, adapting, accelerating, irregular spiking, or show delayed initiation' (Naud et al. 2008). The flourishing diversity goes hand in hand with more limited theoretical treatment and more difficult simulations.

The two theoretical additions of exponential driving and adaptation can also be considered independently. Especially the latter, sometimes named adaptive leaky integrate-and-fire (aLIF), allows the access to the two different timescales of the membrane and of the adaptation. With this modification, the model can model activity of diverse neurons faithfully (Gerstner and Naud 2009; Kobayashi 2009; Teeter et al. 2018). The related mechanism of making the threshold adaptive has been used to give neurons longer memory (Bellec et al. 2020), and we will use adaptation in Chapter 7 for the implementation of a reliable delay mechanism.

2.3. Deep learning

After having covered the modelling of spiking neurons, we turn to the basics of ANNs since one of the central contributions of this thesis is the application of DL methods to SNNs. DL is a subfield of machine learning (ML), and due to DL's overwhelming success in the last decade the terms are now often used interchangeably. In general, ML includes all forms of data-driven algorithms. An algorithm's performance on some target metric is typically measured by a loss $\mathcal{L}$ or energy E, and optimising this quantity is associated with enhancing performance. Examples for this are the Gauss-Newton algorithm or the Levenberg-Marquardt algorithm to iteratively solve a nonlinear least-squares problem. As we will see, optimising a network of artificial neurons fits right into this description of iteratively minimising a loss. The branch of DL is characterised by the structure of the optimised objects: deep networks consisting of several layers, each of which comprises many simple, abstract neurons.

Artificial neurons As alluded to in the introduction, a lot of the concepts in DL are inspired by the brain, but differ in central aspects. The building block, the artificial neuron, is still similar to the original described by McCulloch and Pitts (1943): a neuron – deprived of intrinsic dynamics – instantaneously adds up inputs $\boldsymbol{x}$, graded by its weight parameters $\boldsymbol{w}$ to get an abstract membrane potential a, offset by a potential bias b,

$$a = \boldsymbol{w}\boldsymbol{x} + b \; . \tag{2.19}$$

Before this state serves as the input to another neuron, the activation function φ is applied, introducing a nonlinearity. The original model (McCulloch and Pitts 1943) used thresholding for binary output. Modern neuronal units employ a variety of activation functions, common examples being ReLU or sigmoid functions. When considering a layered setup (Fig. 2.4) with a componentwise applied activation function we get

$$\boldsymbol{a}^{(i)} = \boldsymbol{w}^{(i)}\boldsymbol{y}^{(i-1)} + \boldsymbol{b}^{(i)} \; , \tag{2.20}$$

$$\boldsymbol{y}^{(i)} = \varphi(\boldsymbol{a}^{(i)}) \; . \tag{2.21}$$

Conveniently, the parameters belonging to layer i like weights or biases are often generalised to $\boldsymbol{\theta}^{(i)}$, and $\boldsymbol{\theta}$ encompassing all parameters of the network.

Basic forms of neural networks The simplest case is the perceptron described by Rosenblatt (1958), consisting of just one such layer of neurons. This is still used as a baseline and termed linear classifier, often with a mean squared error (MSE) loss between outputs and the targets.

The nonlinearity, especially over many layers, can lead to sufficiently complex combinations of the input. This enables the *kernel trick*: simple linear regression on the expanded latent space can perform well as a classifier or in regression tasks. This amounts to using the network as a support vector machine (SVM), and applied to time sequence tasks it is called a liquid state machine (LSM).

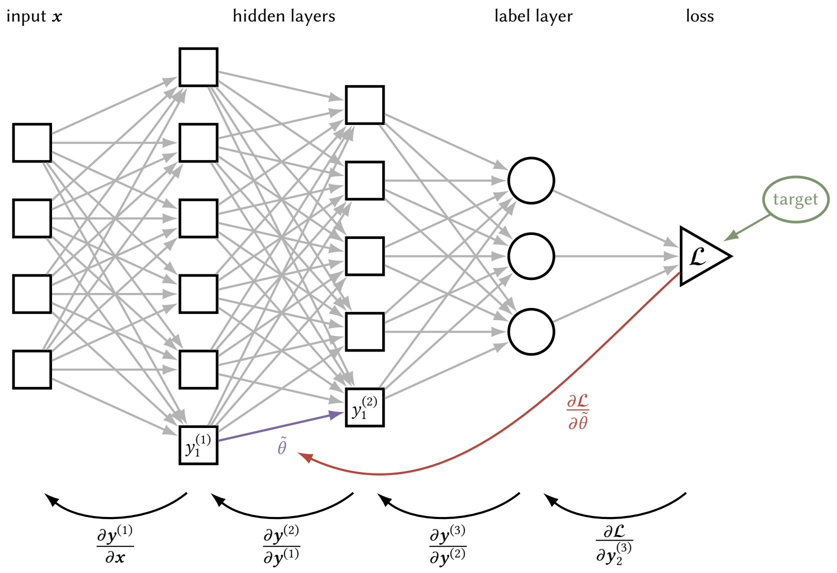
input $\boldsymbol{x}$
hidden layers
label layer
loss
target
$\mathcal{L}$
$y_1^{(1)}$
$y_1^{(2)}$
$\tilde{\theta}$
$\frac{\partial \mathcal{L}}{\partial \tilde{\theta}}$
$\frac{\partial y^{(1)}}{\partial x}$
$\frac{\partial y^{(2)}}{\partial y^{(1)}}$
$\frac{\partial y^{(3)}}{\partial y^{(2)}}$
$\frac{\partial \mathcal{L}}{\partial y_2^{(3)}}$

Figure 2.4.: Network structure. Example of a feedforward architecture, highlighting the forward (top) and backward propagation (bottom). Given inputs $\boldsymbol{x}$, activity subsequently propagates through the hidden layers until the activity of the label neurons $\boldsymbol{y}^{(L)}$ is used, together with the target, to calculate the value of the loss function $\mathcal{L}$. In each layer i, the activity $\boldsymbol{y}^{(i)}$ depends on the activity in the previous layer $\boldsymbol{y}^{(i-1)}$ and parameters $\boldsymbol{\theta}^{(i)}$, exemplified by $\tilde{\theta}$ between $y_1^{(1)}$ and $y_1^{(2)}$. The neurons in the last layer are often distinct, by using a different or no rectifying activation function (implied by the circular shape). To apply gradient descent (Eq. (2.22)) to a single parameter $\tilde{\theta}$, the chain rule can be used (red). An efficient algorithm iteratively propagates an error backward, aptly named error backpropagation (black arrows). Given the local error, the update is obtained by multiplication with $\partial y_2^{(1)}/\partial\tilde{\theta}$.

Training the parameters The most common training scheme is called gradient descent (GD), i.e., changing the parameters in the direction of steepest descent of the loss:

$$\Delta\tilde{\theta} \propto -\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \tilde{\theta}} \,. \tag{2.22}$$

This works if the loss function is a differentiable function of the output activation $\mathcal{L}(\boldsymbol{y})$, which itself depends according to Equation (2.20) on its parameters and input. When generalising to a deep network of L layers, a so called multilayer perceptron (MLP), this still holds as each layer depends continuously on its input and parameters, so

$$\mathcal{L}[\boldsymbol{\theta}, \boldsymbol{x}] = \mathcal{L}\left(\boldsymbol{y}^{(L)}\left(\boldsymbol{\theta}^{(L)}, \boldsymbol{y}^{(L-1)}\left(\boldsymbol{\theta}^{(L-1)}, (\ldots, \boldsymbol{y}^{(1)}(\boldsymbol{\theta}^{(1)}, \boldsymbol{x}))\right)\right)\right) \,. \tag{2.23}$$

This composition of functions can in principle be used together with Leibniz' chain rule[12] to

[12]The required differentiability is not necessarily fulfilled: the ReLU activation function is not differentiable at its kink. It has been investigated (Lee et al. 2020; Bolte and Pauwels 2020), and in practice does not impede training.

directly calculate gradients like

$$\Delta\boldsymbol{\theta}^{(i)} \propto \frac{\partial\mathcal{L}[\boldsymbol{\theta},\boldsymbol{x}]}{\partial\boldsymbol{\theta}^{(i)}} \tag{2.24}$$

$$= \frac{\partial\boldsymbol{y}^{(i)}}{\partial\boldsymbol{\theta}^{(i)}} \cdot \frac{\partial\mathcal{L}[\boldsymbol{\theta},\boldsymbol{x}]}{\partial\boldsymbol{y}^{(i)}} \tag{2.25}$$

$$= \frac{\partial\boldsymbol{y}^{(i)}}{\partial\boldsymbol{\theta}^{(i)}} \cdot \frac{\partial\boldsymbol{y}^{(i+1)}}{\partial\boldsymbol{y}^{(i)}} \cdots \frac{\partial\boldsymbol{y}^{(L)}}{\partial\boldsymbol{y}^{(L-1)}} \cdot \frac{\partial\mathcal{L}[\boldsymbol{\theta},\boldsymbol{x}]}{\partial\boldsymbol{y}^{(L)}} , \tag{2.26}$$

however with limited computationally efficiency. The efficient implementation is called error backpropagation, and iteratively propagates the error of each layer $\partial\mathcal{L}/\partial\boldsymbol{y}^{(i)}$ backwards,

$$\frac{\partial\mathcal{L}[\boldsymbol{\theta},\boldsymbol{x}]}{\partial\boldsymbol{y}^{(i)}} = \frac{\partial\boldsymbol{y}^{(i+1)}}{\partial\boldsymbol{y}^{(i)}} \cdot \frac{\partial\mathcal{L}[\boldsymbol{\theta},\boldsymbol{x}]}{\partial\boldsymbol{y}^{(i+1)}} . \tag{2.27}$$

With a shorthand for the error in the i^{th} layer $\boldsymbol{\delta}^{(i)} = \partial\mathcal{L}[\boldsymbol{\theta},\boldsymbol{x}]/\partial\boldsymbol{y}^{(i)}$ and the forward activation described above (Eqs. (2.20) and (2.21)), this equation is often written in the form

$$\boldsymbol{\delta}^{(i)} = \varphi'(\boldsymbol{a}^{(i)}) \cdot \boldsymbol{w}^{(i+1),T} \cdot \boldsymbol{\delta}^{(i+1)} . \tag{2.28}$$

Figure 2.4 highlights how this propagated error can be used together with the local derivative $\partial\boldsymbol{y}^{(i)}/\partial\boldsymbol{\theta}^{(i)}$, to calculate the gradient. The error backpropagation algorithm was independently found by Linnainmaa (1970) and Werbos (1982), the widespread use was started by Rumelhart et al. (1986).

Deep Learning in practice Above, the knowledge of the target corresponding to the input is implied, i.e., the case for supervised training, where the used dataset contains inputs and labels, different from unsupervised and reinforcement learning (RL). While most advances in DL include forms of the latter two, e.g., for transformers the self-supervised text prediction or their fine-tuning with reinforcement learning from human feedback (RLHF), here we focus on supervised settings. All these settings have the requirement of propagating the error through the network (Eq. (2.27)), independently of how the initial error arises, and therefore the supervised case serves as a good starting point.

Typically, training works like this: a given dataset is already separated in *test* and *train* parts. The latter can be fully utilised for training, whereas the network is only allowed to access the former after the end of the training, as a final trial of its performance. For practical purposes, from the training dataset a *validation* part can be split off for measuring performance on unseen data during training, while not tainting the test data. To guarantee comparability, it is best practice to provide these splits to users.

When performing GD based on an individual sample in a dataset, the parameter update is not guaranteed to decrease the loss for other samples. In the DL context, the method is thus more precisely called stochastic gradient descent (SGD) as first described in Robbins and Monro (1951) and Kiefer and Wolfowitz (1952). Averaging over the full dataset, batched GD, helps to smooth excessive gradients but given the large size of typical datasets would be slow. Additionally, instructive learning signals from some patterns might be averaged out in the large dataset, further obstructing the learning. This is countered by performing SGD on minibatches: small, random subsets of the training data that benefit the signal-to-noise ratio, while being efficiently computable through parallelisation.

This form of training is just one of many methods that make modern DL feasible. Some of these tricks can be directly carried over to training SNNs, like SGD or the use of optimisers like Adam (Kingma and Ba 2014), while others have no direct equivalents, e.g., because they are specific to ANN activation like batch norm (Ioffe and Szegedy 2015).

Training spiking neurons In contrast to these methods, training SNNs in a supervised fashion was established a lot later. With back reference to biology, typical approaches included the use of STDP to control firing times of neurons (Pfister et al. 2006), or to extract salient features in spike trains (Kempter et al. 1999; Song et al. 2000; Legenstein et al. 2005; Jordan et al. 2021).

A notable example of training a spiking neuron along the lines of DL is the *Tempotron* (Gütig and Sompolinsky 2006): it features an ad hoc learning rule for a single neuron. The learning rule depends on the analysis of the voltage trace. It has been used for separation of spike patterns, but is limited due to the single layer architecture. This restriction is not present in *SpikeProp* (Bohte et al. 2000; Bohte et al. 2002), that featured networks with a hidden layer and an approximative method of connecting an error in the label spike time with gradients of the parameters.[13] Because of the lack of known and challenging tasks to compete on, it did not permeate into the wider community. In addition, the use of time-stepped simulation for the forward pass to obtain the activity implies long simulation times for large networks.

One reason why the application of DL methods to SNNs happened relatively late could be connected to the following: in the neuromorphic and computational neuroscience literature, there are commonly found, though misleading or even wrong, statements that spikes are not differentiable, or that backpropagation is not possible because the derivatives are not defined. Probably, this statement arises because the voltage does indeed reset in a non-differentiable, even discontinuous way. However, the implied question of how a change in the membrane potential changes the spikes is ultimately not needed to calculate the gradients: information propagation in a network is driven only by spikes, and the dependence of each spike on its input and weights is differentiable.[14]

While not strictly necessary to find gradients with respect to some parameters, it is still instructive to look at the influence of a change in the membrane potential u_mem on the spike train $\partial\rho/\partial u_\mathrm{mem}$. In their work on *surrogate gradients*, Zenke and Ganguli (2018) and Neftci et al. (2019) investigated this by writing the spike train $\rho(t)$ (Eq. (2.7)) as a function of the voltage, $\rho(t) = \Theta(u_\mathrm{mem}(t) - \vartheta)$. The replacement works because the voltage is reset immediately after a spike. It is made primarily as a motivation for the gradient calculation but can be related to neurons with escape noise (Gygax and Zenke 2025). For the backward pass, the Heaviside step function Θ is approximated by a continuous function σ[15] which allows calculating the derivative

$$\frac{\partial\rho(t)}{\partial u_\mathrm{mem}(t)} \to \sigma'(u_\mathrm{mem}(t)) \,. \tag{2.29}$$

With this approximation, the whole machinery of DL is usable, and this ease of use is one

[13]As was later realised, the method is in fact not approximative but with the help of the implicit function theorem can be proven to be exact, see Yang et al. (2014) and Section SI.C.

[14]This is derived in Chapter 5 for individual spikes, in Chapter 7, Section SI.D for multiple spikes, and is by now also found in the literature, see e.g., Wunderlich and Pehle (2021). Interestingly, the handling of the Dirac delta distribution, including its derivative, is well-defined in functional analysis, more specifically distribution theory. We make use of that, e.g., in Chapter 7, Section SI.E, Equations (SI.31) to (SI.34).

[15]In the original publication, Zenke and Ganguli (2018) used the negative side of a fast sigmoid for performance reasons.

reason for the popularity of the surrogate gradient approach. In addition, the resulting computational graph is close to certain ANNs and thus the method can profit from existing software engineering. However, this computational graph is extremely removed from the event-based nature of neuronal computation: generally, surrogate gradients are computed for discretised neuron dynamics at finite time steps $0, \Delta t, 2\Delta t, \dots, T$. In this formalism, spikes are represented by 1 in the spiking tensor ρ that has dimensions of the number of neurons and time steps. A parameter update, according to error backpropagation, is then performed by the matrix product

$$\Delta\theta = \sum_t \frac{\partial u_{\mathrm{mem}}}{\partial \theta}[t] \cdot \sigma'(u_{\mathrm{mem}}[t]) \cdot \frac{\partial \mathcal{L}}{\partial \rho}[t] \,. \tag{2.30}$$

The computation requires a lot of memory as the voltage trace of all neurons needs to be retained. For neuromorphic hardware, this implies the need for vast data storage or transfer, making an implementation challenging, but not impossible (Cramer, Billaudelle et al. 2022).

With surrogate gradients, information propagates (backward) even without a spike, which allows the network to create activity from quiet states. This is certainly beneficial, for example when starting from a random initial state. All the more so, because in contrast to a blanket increase of weights, the parameters are changed in a guided manner to create spikes where they have the largest impact. Yet, the required flow of information through sub-threshold activations has a downside besides the increased computation: because there is an approximation at each neuron at each time step (Eq. (2.29) in Eq. (2.30)), the information is diluted, directing the resulting gradient away from the true gradient. In fact, it has been shown that the calculated surrogate gradient can even point in direction of steepest *ascent*, actively degrading performance (Gygax and Zenke 2025).

Mostafa (2018) put forward a radically different approach: for the nLIF neuron model, he found a way to express the spike time of a neuron as an explicit function of its weights and input spikes, permitting the application of error backpropagation in networks with up to two hidden layers. One point to stress here is that while the timing of spikes is differentiable, the appearing or disappearing spikes are not, so ample activity has to be ensured, e.g., by an ad hoc rule of increasing weights for sufficient spikes (Mostafa 2018). The approach was limited to the nLIF neuron model, which impedes the implementation on mixed-signal neuromorphic hardware. However, it allowed our later insight that even with more complicated neuron models, SNNs are trainable based on exact gradients: information is passed by spikes, and – except for a zero measure set – their timing shifts continuously when changing parameters.

2.4. Neuromorphic hardware

As discussed in the introduction, the goal of neuromorphic engineering, as the name suggests, is to build hardware that is inspired by the brain. This technology comes in many forms and with varying motivations. The initial development of neuromorphic technology started with sensors: Mead and Mahowald (1988) described an implementation of visual processing recreating a retina, shortly followed by an electronic cochlea for auditory perception (Lyon and Mead 1988). These early papers attempt to mimic models of biological processes in physical realisations (Mead 1989; Mead 1990). According to their own statement, the authors wanted to demonstrate and verify their understanding of biological systems by replicating them in silicon

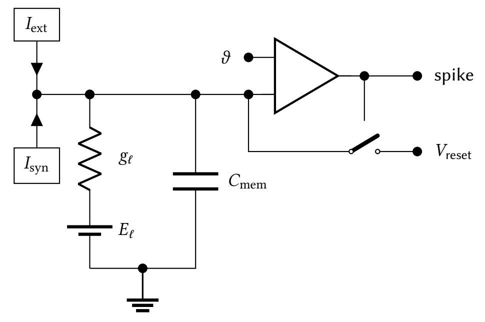

Figure 2.5.: Examples of neuromorphic hardware. *Left:* Basic circuit modelling a LIF neuron (Eq. (2.4)). Modified versions of this circuit serve as a building block in the design of analogue neuromorphic systems. *Right:* Photograph of the production setup of BrainScaleS-2 (taken with permission from Stradmann, Ilmberger et al. 2025).

circuits.[16] They pursued this approach not only for sensors but also for neurons (Mahowald and Douglas 1991). Finding a substrate that *emulates* desired traits of a model system is now often called physical computing. This is different from the more conventional approaches of either analytically investigating or numerically simulating a model. The latter usually implies the iterative integration of differential equations, which especially for nonlinear dynamics can require complicated integration schemes or fine time steps. For example, to capture neural effects like synchronisation, a high resolution in the integration is necessary (Hansel et al. 1998; Valadez-Godínez et al. 2020), raising the computational load. Physical computation comes with limited flexibility and control, but promises gains in energy efficiency and speed for specific systems (Indiveri et al. 2011; Boahen 2017; Buhler et al. 2017). An often used example here is the characteristic behaviour of a neuron as encoded by the LIF model: the integration of information over time, and event-based communication can be directly mapped to a circuit (Fig. 2.5), which serves as a building block for many neuromorphic systems. Analogue neuromorphic hardware continues to be successfully developed, this includes the *DYNAP-SE* family (Moradi et al. 2018; Richter et al. 2024) and the *Neurogrid* chip (Benjamin et al. 2014). Another prominent example, BrainScaleS-2, is introduced below. These mixed-signal approaches combine a lot of characteristics of neuromorphics: they compute in-memory, emulate their dynamics, and communicate with spikes.

Digital neuromorphic architectures like *SpiNNaker* are an alternative concept. *SpiNNaker*

[16]Mead and Mahowald (1988) end their paper with a paragraph which seems to be their interpretation of the famous quote by Richard Feynman, ‘What I cannot create, I do not understand’ (Feynman 1988):

> It is our conviction that our ability to realize simple neural functions is strictly limited by our understanding of their organizing principles, and not by difficulties in realization. If we *really* understand a system, we will be able to build it. Conversely, we can be sure that a system is not fully understood until a working model has been synthesized and successfully demonstrated.

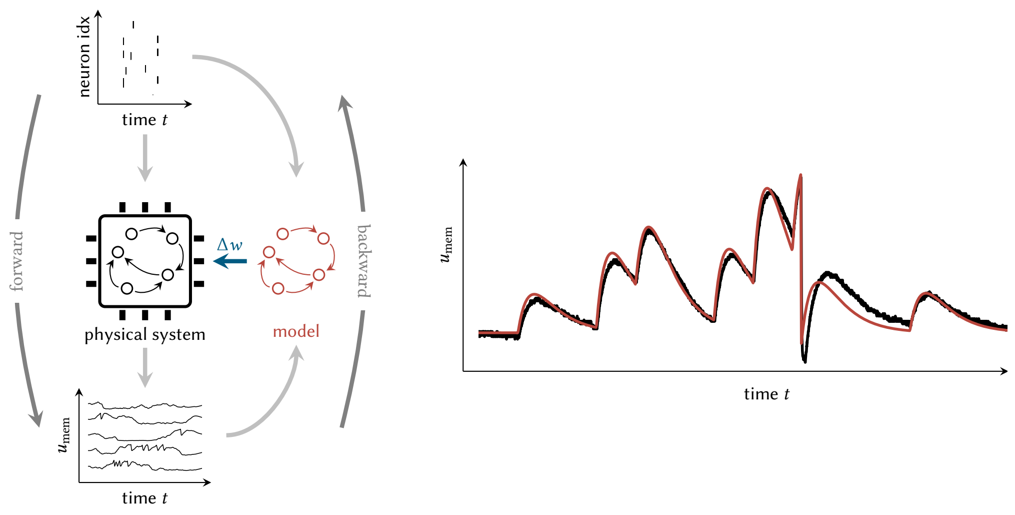

Figure 2.6.: In-the-loop usage of BrainScaleS-2. Adapted from Göltz et al. (2023). *Left:* Spikes (sketched in a raster plot on top) are given as input data to the physical system. The input determines the dynamics of the network on the substrate (black nodes connected with arrows), and observables are retrieved from the system, e.g., the voltage trace as depicted on the bottom. Note that training is also possible based solely on output spike times, requiring drastically less I/O bandwidth. The inputs and outputs inform a model of the dynamics (red) which is used to calculate updates Δw for the parameters of the system (blue). *Right:* Sketch of an exemplary trace from BrainScaleS-2 in black and a corresponding model in red. In addition to high frequency jitter with small amplitude, there are deviations the spike occurred. In practice, when PSPs bring the membrane close to, but not over, the threshold, small jitter can be sufficient to create erroneous spikes. When only using single spikes, the mismatch during the refractory period is unproblematic for a model, as it only considers the dynamics until the spike.

consists of distributed *ARM* cores, i.e., it employs traditional cores in a distributed setup with a focus on event-based communication (Furber et al. 2014; Mayr et al. 2019). Other examples are custom ASICs (Frenkel et al. 2019) that also include substrates by industry contributors like *TrueNorth* (DeBole et al. 2019) and *Loihi* (Davies et al. 2018). Intel's *Loihi* implements asynchronous cores and the resulting flexibility and parallelism makes it also useful for tasks not classically associated with neuromorphics like constraint satisfaction problems (Davies et al. 2021; Pierro et al. 2024). More typical DL applications, especially image classification, have been performed on these digital architectures (Esser et al. 2015; Stromatias et al. 2015; Renner et al. 2024). For recent reviews on the spectrum of available neuromorphic technology, we refer to Frenkel et al. (2023) and Basu et al. (2022).

BrainScaleS-2 In this thesis we primarily use BrainScaleS-2, an accelerated neuromorphic system (Fig. 2.5). This substrate is a prime example for mixed-signal neuromorphics: it uses CMOS technology to perform physical computation in analogue electronics (Pehle et al. 2022; Schemmel et al. 2021; Billaudelle et al. 2020). At its core, this system has 512 neuron circuits that each implement the AdEx neuron model introduced in Section 2.2. The circuits are set up in a modular way: with disabled adaptation and exponential components, for example, a circuit can faithfully reproduce LIF dynamics (Billaudelle 2022). Neurons on the chip can be connected with the two 256×256 synaptic crossbar arrays. To increase the fan-in, multi-compartment neurons can be created by connecting neighbouring circuits.

Deviations in the manufacturing process create mismatch between circuits that is called fixed-pattern noise. The noise causes each neuron to behave in a distinct manner. However, because the neuron and synapse parameters can be flexibly configured (Billaudelle et al. 2022), calibration partially compensates for fixed-pattern noise to ensure homogeneous dynamics close to the ideal model. The central contribution of this thesis is a training algorithm for SNNs. To use this algorithm in conjunction with neuromorphic hardware, we choose an in-the-loop (ITL) approach (Fig. 2.6): from the host computer, we set up a network on the chip, stimulate this network and record observables from the neurons. With the observed spike times, parameter updates are computed on the host computer to be used for subsequent experiments. In this manner, we can incorporate the inaccuracies that remain even after the calibration.

Apart from the faithful emulation and flexibility with parameters, BrainScaleS-2 offers a number of advantages: its dynamics are, compared to biological time scales, sped up by a factor of 10^3, enabling the study of learning behaviour that would otherwise take infeasibly long times (Wunderlich et al. 2019; Billaudelle et al. 2021). Nevertheless, BrainScaleS-2 can be interfaced with real-world inputs and can even control suitable motors (Schreiber 2021; Stradmann and Schemmel 2024). In combination with a compute cluster utilised for training, the neuromorphic systems form a research infrastructure that grants online access to a broad audience while minimising obstacles (Stradmann, Ilmberger et al. 2025). As a consequence, our experiments are in principle repeatable by anyone with this online access. Specifically, the hardware experiments in Chapter 5 were performed with *strobe*, a software developed by Cramer, Billaudelle et al. (2022), and in Chapter 7 with *PyNN* (Davison 2008; Müller et al. 2022).

3. Lu.i – A low-cost electronic neuron for education and outreach

The manuscript detailing our educational neuromorphic PCBs was published as

> Yannik Stradmann*, **Julian Göltz***, Mihai A. Petrovici, Johannes Schemmel and Sebastian Billaudelle (2025). ‘Lu.i – A low-cost electronic neuron for education and outreach’. In: *Trends in Neuroscience and Education* 38, p. 100248. DOI: `10.1016/j.tine.2025.100248`

Author contributions

YS and JG are equal contributors to the manuscript: JG came up with the original idea for the project, that was subsequently developed by YS, SB, and JG. The design of the PCB was performed by SB and YS with support from JG. YS and JG jointly wrote the manuscript, with feedback from SB, JS, and MAP.

Manuscript

In accordance with *Elsevier* policies,[17] and the CC BY-NC-ND licence, the manuscript is reprinted in the following.

[17] `https://www.elsevier.com/en-au/about/policies-and-standards/copyright`

Trends in Neuroscience and Education 38 (2025) 100248

Contents lists available at ScienceDirect

Trends in Neuroscience and Education

journal homepage: www.elsevier.com/locate/tine

Lu.i – A low-cost electronic neuron for education and outreach

Yannik Stradmann [*,1,a], Julian Göltz [1,a,b], Mihai A. Petrovici [b], Johannes Schemmel [a], Sebastian Billaudelle [a]

[a] *Kirchhoff-Institute for Physics, Heidelberg University, Heidelberg, Germany*
[b] *Department of Physiology, University of Bern, Bern, Switzerland*

A R T I C L E I N F O

Keywords:
Education
Outreach
Low-cost PCB
Open hardware
Leaky integrate-and-fire
Neuron

A B S T R A C T

With increasing presence of science throughout all parts of society, there are rising expectations for researchers to effectively communicate their work and for teachers to discuss contemporary findings in their classrooms. While the community can resort to established teaching aids for the fundamental concepts of most natural sciences, there is need for similarly illustrative demonstrators in neuroscience. We therefore introduce Lu.i: a parametrizable electronic implementation of the leaky integrate-and-fire neuron model in an engaging form factor. These palm-sized neurons can be used to visualize and experience the dynamics of individual cells and small networks. When stimulated with sensory input, Lu.i demonstrates brain-inspired information processing in the hands of a student. As such, it is actively used at workshops, in classrooms, and for science communication. As a versatile tool for teaching and outreach, Lu.i nurtures the comprehension of neuroscience research and neuromorphic engineering among future generations of scientists and the general public.

1. Introduction

Expanding our understanding of the brain is among the central frontiers of modern science and yet implies some of the longest standing questions humanity has posed to itself. Their fundamental nature induces an intrinsic curiosity about the progress of neuroscience, artificial intelligence, and brain-inspired technology. In contrast to this demand, the repertoire of tangible demonstrators to communicate principles and recent achievements in brain research is limited [1]. In comparison, other fields can build on many centuries of experience to convey their essential concepts through physical demonstrators and live experiments.

In our current understanding, the fundamental principles of information processing in nervous systems lie in neuronal dynamics and synaptic interactions. A strong intuition for these mechanisms is, therefore, the foundation for understanding and investigating more complex processes and emerging phenomena. In the following, we thus present Lu.i – an analog electronic implementation of the leaky integrate-and-fire (LIF) neuron model targeted for educational use as well as scientific outreach. Lu.i features current-based synaptic inputs that enable the formation of simple spiking neural networks (SNNs) and offers control over many parameters, including the time constants and the synaptic weights. The printed circuit board (PCB) visualizes the time-continuous dynamics of the emulated membrane potential and allows interfacing with digital and analog periphery for advanced experiments. It has been optimized for low-cost production, long battery life, and intuitive operation.

2. Neuron and synapse dynamics

Neurons are a family of electrically active cells that compute and communicate by exchanging action potentials. Each cell receives such signals via its synapses and integrates them over time. Once a neuron has accumulated enough input, it becomes active and communicates this by itself sending a spike to other neurons.

Lu.i implements the LIF neuron model, arguably the simplest abstraction that still captures these fundamental properties of neuronal information processing: time-continuous computation, spatio-temporal integration, and event-based communication. This model was originally put forward by Louis Lapicque (`/lu.i la'pik/`) in [2], after whom the PCB was fittingly named. In contrast to models based on specific ion channel dynamics like the one by Hodgkin and Huxley [3], LIF captures essential neuron dynamics in a single state variable: It describes the evolution of a neuron's membrane potential $V_{\mathrm{mem}}(t)$ by the differential equation

* Corresponding author.
E-mail address: yannik.stradmann@kip.uni-heidelberg.de (Y. Stradmann).
[1] These authors contributed equally, listed in reverse alphabetical order.

https://doi.org/10.1016/j.tine.2025.100248
Received 24 September 2024; Received in revised form 24 January 2025; Accepted 25 January 2025
Available online 31 January 2025

$$\tau_{\mathrm{mem}}\frac{\mathrm{d}V_{\mathrm{mem}}(t)}{\mathrm{d}t} = -[V_{\mathrm{mem}}(t) - V_{\mathrm{leak}}] + I_{\mathrm{syn}}(t)/g_{\mathrm{leak}}, \tag{1}$$

where τ_{mem} denotes the membrane time constant, g_{leak} the leak conductance, and V_{leak} its resting potential. $I_{\mathrm{syn}}(t)$ subsumes the time-dependent synaptic currents stimulating the neuron. This differential equation describes a membrane potential which continuously decays to the resting state. It is, however, augmented by a reset condition to mimic the hyperpolarization following the action potentials observed in biological neurons: Whenever the membrane potential crosses the threshold ϑ, the neuron emits a spike. This efferent signal is accompanied by a reset of the membrane potential, where the latter is simply clamped to V_{reset} for the refractory period. For an exemplary time evolution of these dynamics see Fig. 3A.

Lu.i implements current-based synapses with postsynaptic currents following exponential kernels with time constant τ_{syn}. This additional temporal filter mimics the kinetics of synaptic ion channels: Each presynaptic spike j, arriving at time t^j_{pre} at synapse i, triggers an exponentially decaying current

$$I^j_{\mathrm{syn}}(t) = w_i \cdot \exp\left(-\frac{t - t^j_{\mathrm{pre}}}{\tau_{\mathrm{syn}}}\right) \quad \text{for } t > t^j_{\mathrm{pre}}, \tag{2}$$

where w_i denotes the weight of the respective synapse i. This weight variable models the strength of synaptic interaction, its sign corresponds to the positive and negative effect of excitatory and inhibitory neurotransmitters, respectively. The total synaptic current then results as a sum over the individual contributions from all synapses i and spikes j.

In biological neurons, the membrane potential typically resides between $-80\,\mathrm{mV}$ and $0\,\mathrm{mV}$. The temporal dynamics are, however, independent of that absolute voltage scale, and are instead mainly governed by the time constants. We on the one hand chose to slow down the temporal dynamics to a scale that can be visually well perceived and interacted with by experimenters. On the other hand, we opted to reduce the full complexity of the parameter space by fixing two of the potentials, namely the threshold ϑ and reset potential V_{reset}, leaving the leak potential V_{leak} as a free, adjustable, parameter.

3. Electronic implementation

Lu.i realizes the LIF dynamics through a set of analog electronic circuits (Fig. 2) and thus forms a physical model thereof. It exposes the neuronal time constants τ_{mem} and τ_{syn}, the leak potential V_{leak} and all synaptic weights w_i as user-settable parameters (Fig. 3B). The membrane is accessible through a board-edge connector and its voltage – as well as spike events – are visualized by on-board LEDs.

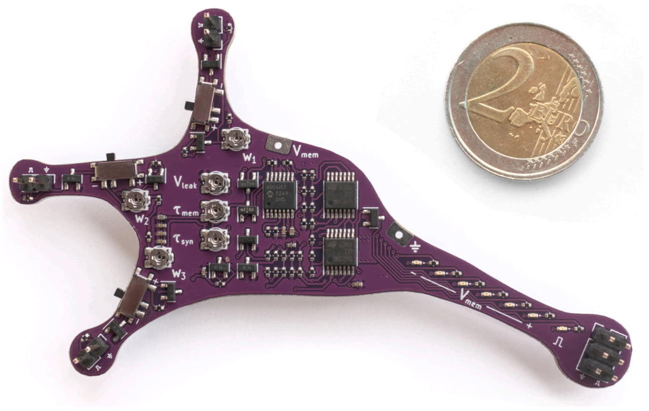

Fig. 1. A single Lu.i neuron PCB, with a 2-Euro coin for scale. To relay information from one neuron to the other, excitatory and inhibitory synapses can be formed by wiring the axonal output (right) to one of the three dendritic terminals (left).

Internally, Eq. (1) is rendered by the combination of capacitor C_{mem} and potentiometer g_{leak}, which form an RC integrator with adjustable time constant τ_{mem} (Fig. 2A). Without external stimuli, V_{mem} decays towards the resting potential V_{leak}, which we generate by the combination of an adjustable voltage divider with a subsequent unity gain buffer. The leak potential can thus be set between $0\,\mathrm{V}$ and the supply voltage V_{DD}. The spike mechanism is implemented by continuously comparing the membrane potential to the threshold, which was chosen as $\vartheta = V_{\mathrm{DD}}/2$ to guarantee sufficient voltage headroom for the comparator (Fig. 2D). Once the membrane reaches this threshold, the comparator trips, indicating a spike and causing a membrane reset. To avoid instabilities, it is fitted with a hysteresis circuit that temporarily reduces the comparator's reference potential to $V_{\mathrm{DD}}/4$ during the onset of a spike. At that point, the capacitor C_{ref} is discharged and the connected comparator trips, thus shorting the membrane to $V_{\mathrm{reset}} = 0\,\mathrm{V}$ via the transistor Q_{reset} to implement the refractory period. R_{ref} and C_{ref} determine the fixed refractory time of approximately 15 ms, which starts once V_{mem} is discharged below $V_{\mathrm{DD}}/4$, where the threshold comparator releases. The control signal for Q_{reset} is re-used as the neuron's axonal output, with a pulse width equivalent to the refractory time.

Lu.i features three synapses implementing the current-based model with an exponential kernel as introduced by Eq. (2). Each of them possesses a tunable weight and can be switched between excitation and inhibition. The synapses share a common synaptic time constant τ_{syn}, which is adjustable over a broad range. For an area- and cost-effective implementation, we minimize the amount of components per synaptic connection: Events from presynaptic neurons control the gate of the n-channel MOSFET Q_{low} (Fig. 2B). Depending on the selected polarity $S_{\mathrm{sign},i}$, this transistor either directly discharges the shared synaptic integrator or indirectly charges it via the p-channel MOSFET Q_{high}. For each event, this synaptic trace is in- or decremented by a fixed amount of charge proportional to the respective weight $g_{\mathrm{w},i}$ which can be configured through a potentiometer. The time constant $\tau_{\mathrm{syn}} = C_{\mathrm{syn}}/g_{\tau_{\mathrm{syn}}}$ of the integrator can be similarly tuned (Fig. 2C). Especially in light of the additional filter introduced by the membrane, the resulting temporal behavior closely approximates the instantaneous response of the original model. The synaptic current I_{syn} is derived from the integrator state through a V-I conversion stage. As such, it consists of two voltage-controlled current sources – each built from a resistor, a MOSFET, and an operational amplifier. Q_{push} and Q_{pull} operate in a push-pull configuration and generate two antagonistic currents. Their difference is proportional to the deflection of the integrator and corresponds to the total postsynaptic current I_{syn} that stimulates the membrane.

Lu.i displays its state through a set of LEDs. Six of them form a bar that visualizes the membrane potential, and a seventh LED indicates efferent spikes with a flash. This interface is sketched in Fig. 3A for various states of the neuron. The voltmeter is implemented through a set of comparators and a resistor ladder to generate the respective reference potentials. While these circuits take up significant area on the PCB, they have been omitted from the schematic for clarity. This intuitive on-board interface enables standalone operation and the visualization of network activity and signal propagation therein. Experimentation with external equipment is, however, encouraged and allows more detailed insights into the neuron dynamics. For that purpose, the emulated membrane is accessible through a pad at the board edge for interfacing with, e.g., current sources and oscilloscopes.

The PCB is powered from a single CR2032 coin cell, which we chose for its small form factor, wide availability, and comparably high capacity at low cost. All voltage references of the circuit are derived relative to this supply voltage of nominal 3 V. The temporal dynamics are thus, on first order, invariant to the battery voltage. This ensures mostly stable operation across the entire lifetime of the cell, which results in approximately 24 h of continuous use. Lu.i can be powered down completely through a switch on its back side.

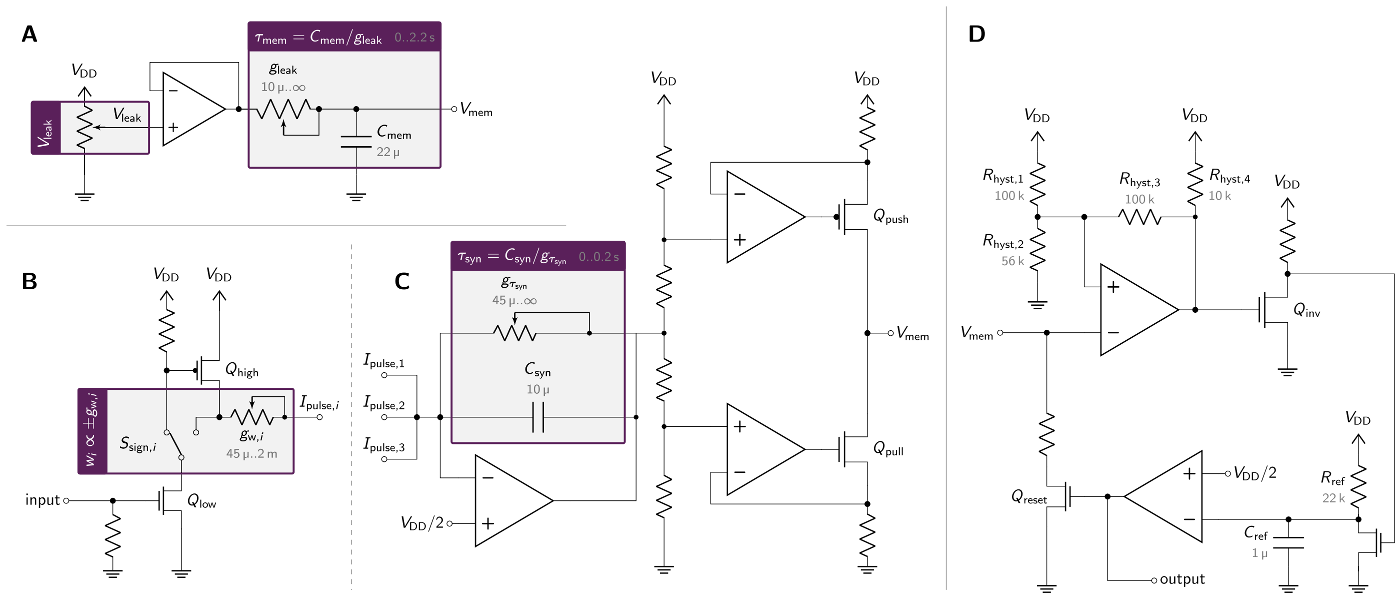

Fig. 2. Schematic of the LIF emulation circuit implemented in Lu.i. (A) Membrane capacitance and leak conductance. (B) Current-based synaptic input circuits. This circuit is instantiated three times, once per synapse. (C) Synaptic integrator and voltage-to-current conversion circuit. (D) Threshold, reset, and refractory circuit. The spike output pulse is derived from the neuron's reset signal and of equivalent duration. The purple boxes relate all user-settable parameters to their representation in the circuit.

While aiming for an intuitive and appealing form factor, the PCB has been strongly optimized for low-cost fabrication. This is reflected in the selection of components as well as the layout, which only relies on a simple two-layer PCB. As a result, we achieved a unit price of around US $3 (excluding the battery) already for batches below 1000 Lu.i neurons. As the backside only contains the battery holder and an optional power switch, fabrication costs can be further reduced by restricting automated assembly to the top layer.

4. Exploring neural computation with Lu.i

Lu.i was designed to illustrate two of the fundamental aspects of biological neurons: spatio-temporal accumulation of input and event-based communication, both of which are captured by the LIF model. These aspects can be demonstrated in a set of experiments of increasing complexity, some of them shown in Fig. 3.

The first property – leaky integration of input – can be seen in Fig. 3A: The membrane potential rises after weak excitatory stimuli and decays back to the resting potential, similarly with inhibitory input. The resulting trajectories are shaped by the adjustable time constants τ_{syn} and τ_{mem}. These determine the time scales on which consecutive inputs are integrated and stacked. Only when the threshold is reached, an efferent spike is triggered and visible externally. On Lu.i, these dynamics can be observed using an on-board LED strip visualizing the membrane state and spike output, as shown in Fig. 3A. Neurons compute through this combination of analog integration and thresholding, for example by performing spatio-temporal coincidence detection. Exploring the impact of the model parameters on this computation – in case of coincidence detection on the sensitivity or detection window – is a worthwhile educational exercise: When Lu.i receives two spikes with a certain time difference, the neuron's parametrization determines whether an output spike is emitted. With short time constants, it becomes active only for inputs close to each other, whereas longer time constants result in a larger detection window. These temporal dynamics fundamentally determine the timescale on which information is processed.

In contrast to the local computation on their membranes, neurons communicate through temporally sparse spike events. This signal propagation can be demonstrated in a simple two-neuron network (Fig. 3C), where a synaptic connection is formed by a cable between the presynaptic axon and a postsynaptic dendrite. By choosing a resting potential above the threshold, the first neuron can act as a regularly firing spike source to the second. As before, the stacking of excitatory stimuli and the reset upon threshold crossing can be observed on the membrane of the postsynaptic cell. The behavior of both neurons is clearly visible using the built-in LEDs without an external oscilloscope. Already in this simple setup, the influence of the synaptic parameters can be explored: For example, the combination of a short synaptic time constant and a strong excitatory weight can be used to trigger one spike for each incoming event. Increasing the synaptic time constant, while lowering the weight, can lead to a delayed propagation of single spikes. This can be used to build delay chains, which vividly illustrate the finite propagation speed of neural signals. Once these chains are closed (Fig. 3D), their activity becomes self-sustained.

Fig. 3 E shows a more complex example, where rate-based AND, OR and – in combination – XOR gates are implemented using three Lu.i neurons. In this case, the OR (AND) gate is implemented by a single neuron that has been tuned to fire for at least one (two) active presynaptic neurons. The output of the OR neuron excites the XOR cell, with the AND neuron acting inhibitorily.

While the inputs A and B can be presented using Lu.i neurons (e.g., in leak-over-threshold configuration), we have used an external microcontroller to stimulate the network in Fig. 3E. With a pulse duration of 15 ms and a signal level of approximately 2.5 V, Lu.i's event output signal can be detected by most 3.3 V and 5 V microcontrollers. The event inputs on Lu.i are compatible with signal levels from 1.8 V to 20 V, allowing to interface with a great variety of sensors and devices.

Due to its simplicity, the XOR network is attractive in educational and outreach environments. Inspired by existing literature, more complex networks have emerged from collaborations of researchers across all areas of neuroscience, including realtime sound localization [4], a balanced random network [5], a ring attractor model [6], an echo localization latch [7], and – with preprocessing of the analog signals – a brightness change detection circuit. Lu.i has been used repeatedly to teach a younger audience about fundamentals of neuroscience and

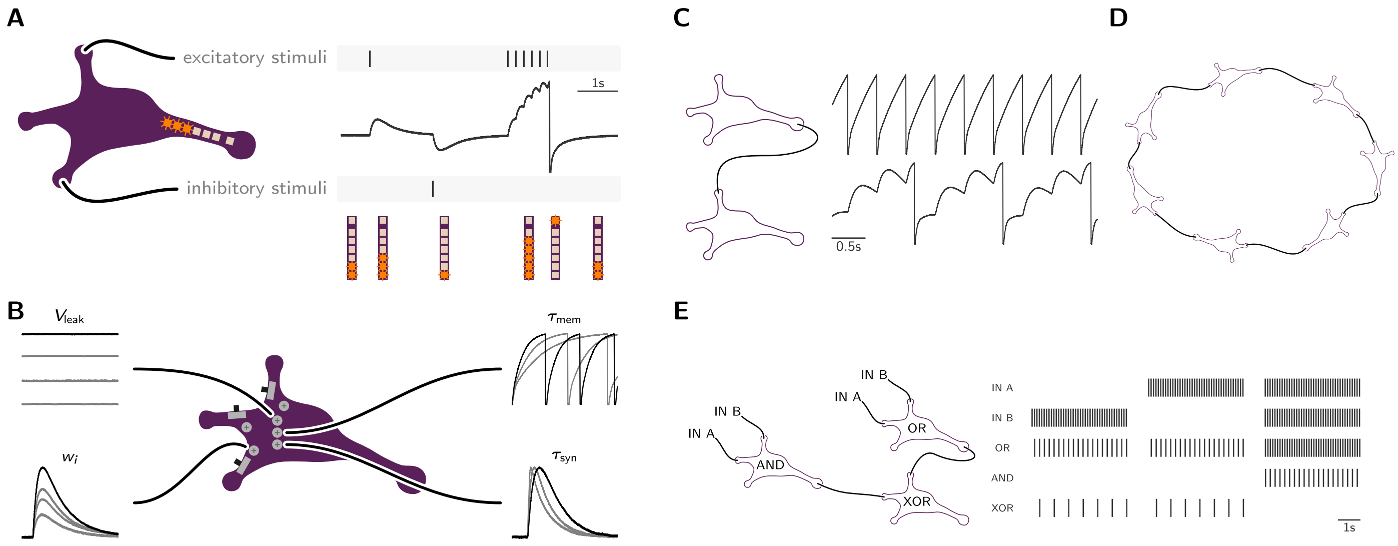

Fig. 3. Exploration of single-cell and network dynamics with Lu.i. (A) A single Lu.i neuron receiving multiple excitatory and a single inhibitory events. The depicted trace is an analog recording of V_{mem} on the board. It shows how stimuli are integrated on the membrane, which continuously leaks back to the resting potential. If the threshold ϑ is reached, the neuron sharply resets and emits a spike. For applications without an oscilloscope at hand, each Lu.i neuron features a bar of LEDs to display the current membrane potential as well as axonal spikes (top LED, flashing). (B) Tuneable neuron parameters on Lu.i. Each model parameter is represented by a small potentiometer (cf. Fig. 1), all three synaptic weights are individually configurable in sign and strength. The traces showing the influence of τ_{syn} have been normalized in amplitude. (C) Analog recording of the membrane potential V_{mem} of two Lu.i neurons. The top trace shows the dynamics of a circuit that is configured with $V_{\text{leak}} > \vartheta$ and emits spikes at regular intervals. This neuron projects onto a second one (bottom trace), which is excited by these events, integrates the post-synaptic current and – eventually – also spikes. (D) Wiring diagram of a closed, circular delay chain built from seven Lu.i neurons. (E) Spike recording of three Lu.i boards, configured to represent rate-based AND, OR and – combined – XOR gates. For panels (A) and (E), the input events are presented by an external microcontroller.

physical computing, also in combination with a subsequent transition to neuromorphic systems made accessible through the european research initiative EBRAINS. Within the first two years, Lu.i has been used at more than 20 workshops held by lecturers not affiliated with the group of authors. Across all described applications, it was used to compellingly illustrate fundamental topics across a wide range of research areas from robotics to systems neuroscience.

5. Discussion

This manuscript presents Lu.i, a palm-sized electronic neuron with versatile applications for teaching and scientific outreach. It can be used to illustrate the dynamics of individual neurons under different parametrizations and their interaction in small spiking neural networks (Fig. 3). Featuring various connectivity options as well as on-board visualization aids, Lu.i can be used stand-alone or in combination with external equipment, like oscilloscopes, current sources, or microcontrollers.

Lu.i complements a range of pedagogical tools spanning from experimental to computational neuroscience [8,9]. Among those are guided experiments on tissue and living animals, which are arguably the most natural way to convey biological concepts but always imply ethical and logistical challenges. Simulation-based curricula, on the contrary, trade immediacy with ease-of-use and simplicity, even when considering graphical user interfaces [10,11]. Reducing experimentation on living tissue in accordance to the 3R principles [12] while keeping the benefits of interactive teaching [13], the concept of tangible hardware has been put forward before [14–18]. As another effort in this direction, Lu.i combines an inviting interface with an analog yet accurate implementation of the LIF model. The latter is sufficiently complex and flexible to allow illustration of fundamental biological phenomena as well as the concept of physical computation. The PCB is optimized for cost-effective manufacturing to ease acquisition especially for educational institutions. With its engaging form factor, Lu.i has been welcomed at various conferences and workshops, leading to adoption by teachers and tutors in classrooms (Fig. 4). As such, the project received enthusiastic responses initiating collaborations across both different areas of expertise and from pupils to faculty.

The Lu.i project is available as open hardware[2] and undergoes active development. The circuits are continuously improved and future versions might be accompanied by additional extensions, such as sensory spike sources or actuators. In conjunction with the above-mentioned collaborations on courses and workshops using Lu.i, a curriculum of teaching material and feedback is being collected to nurture adoption among teaching personnel.

Funding

The presented work has received funding from the Manfred Stärk Foundation, the EC Horizon 2020 Framework Programme under grant agreement 945539 (HBP) and Horizon Europe grant agreement 101147319 (EBRAINS 2.0), the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy EXC 2181/1-390900948 (Heidelberg STRUCTURES Excellence Cluster), and the VolkswagenStiftung under grant number 9C840.

Ethical statement

Not applicable.

CRediT authorship contribution statement

Yannik Stradmann: Writing – review & editing, Writing – original draft, Visualization, Validation, Project administration, Methodology, Investigation, Conceptualization. **Julian Göltz:** Writing – review & editing, Writing – original draft, Visualization, Validation, Project administration, Methodology, Investigation, Conceptualization. **Mihai A. Petrovici:** Writing – review & editing, Supervision, Funding acquisition. **Johannes Schemmel:** Writing – review & editing, Supervision,

[2] https://github.com/giant-axon/lu.i-neuron-pcb.

Fig. 4. Lu.i has played an integral role at various events all over the world for teaching and outreach applications, including: Nacht der Forschung (Switzerland, 2022, ▦), CapoCaccia Workshop toward Neuromorphic Intelligence (Italy, 2023, ▦), TReND in Africa (Ghana, 2023, ▦), and Deutsche SchülerAkademie (Germany, 2023, ▦).

Funding acquisition. **Sebastian Billaudelle:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Project administration, Methodology, Investigation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors would like to thank the Manfred Stärk Foundation for sponsoring the first batch of Lui neurons in preparation for the Nacht der Forschung (Bern, 2022), Andreas Baumbach for his dedicated support during early development and prototyping, Laura Kriener for patiently tutoring Lu.i users and passionately building functional networks, Matthias Penselin and Nazarii Bogachuk for bringing the project into school classrooms, the Human Brain Project for sponsoring subsequent production batches, Björn Kindler for his administrative and public relations support, Barbara Webb, Ben Efron, Chenxi Wen, Robin Dietrich, Sarah Hamburg, Shreyan Banerjee, and Yulia Sandamirskaya for pioneering experiments during the CapoCaccia Workshop toward Neuromorphic Intelligence 2023, Saray Soldado-Magraner for taking Lu.i to Ghana and her photos thereof, the Deutsche SchülerAkademie for kindly authorizing the use of a group photo, Maxime Zimmermann for representing Lu.i at the FENS Forum 2024, and the anonymous reviewers for their comprehensive and constructive feedback that considerably improved this manuscript. Finally, we'd like to thank the NeuroTMA and ElectronicVisions groups for their continuous support and inspirational atmosphere.

References

[1] G.J. Gage, The case for neuroscience research in the classroom, Neuron 102 (5) (2019) 914–917, https://doi.org/10.1016/j.neuron.2019.04.007.

[2] L. Lapicque, Recherches quantitatives sur l'excitation electrique des nerfs traitee comme une polarization, J. Physiol. Pathophysiol. 9 (1907) 620–635.

[3] A.L. Hodgkin, A.F. Huxley, A quantitative description of membrane current and its application to conduction and excitation in nerve, J. Physiol. 117 (4) (1952) 500–544. http://view.ncbi.nlm.nih.gov/pubmed/12991237.

[4] L.A. Jeffress, A place theory of sound localization, J. Comp. Physiol. Psychol. 41 (1) (1948) 35.

[5] N. Brunel, Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons, J. Comput. Neurosci. 8 (3) (2000) 183–208, https://doi.org/10.1023/a:1008925309027.

[6] I. Pisokas, S. Heinze, B. Webb, The head direction circuit of two insect species, eLife 9 (2020) e53985, https://doi.org/10.7554/eLife.53985.

[7] C. Wen, T.K. Horiuchi, The curved openspace algorithm and a spike-latency model for sonar-based obstacle avoidance, Front. Neurorobot. 16 (2022), https://doi.org/10.3389/fnbot.2022.850013.

[8] T.C. Marzullo, G.J. Gage, The spikerbox: a low cost, open-source bioamplifier for increasing public participation in neuroscience inquiry, PLoS ONE 7 (3) (2012) 1–6, https://doi.org/10.1371/journal.pone.0030837.

[9] B. Latimer, D. Bergin, V. Guntu, D. Schulz, S. Nair, Open source software tools for teaching neuroscience, J. Undergrad. Neurosci. Educ. 16 (3) (2018) A197.

[10] T. Bekolay, J. Bergstra, E. Hunsberger, T. DeWolf, T.C. Stewart, D. Rasmussen, X. Choo, A.R. Voelker, C. Eliasmith, Nengo: a python tool for building large-scale functional brain models, Front. Neuroinform. 7 (2014), https://doi.org/10.3389/fninf.2013.00048.

[11] S. Spreizer, S. Johanna, S. Rotter, M. Diesmann, B. Weyers, Nest desktop, an educational application for neuroscience, eNeuro 8 (6) (2021), https://doi.org/10.1523/eneuro.0274-21.2021.

[12] W.M.S. Russell, R.L. Burch, The Principles of Humane Experimental Technique, Methuen, London, 1959.

[13] C. Crouch, A.P. Fagen, J.P. Callan, E. Mazur, Classroom demonstrations: learning tools or entertainment? Am. J. Phys. 72 (6) (2004) 835–838, https://doi.org/10.1119/1.1707018. https://doi.org/10.1119/1.1707018.

[14] K. Eng, G. Indiveri, V. Djambazova, P. Pyk, My first neuron: an educational tool for teaching neural computation. Frontiers in Neuroinformatics: 1st INCF Congress of Neuroinformatics 2008, Frontiers Media SA, 2008, https://doi.org/10.3389/conf.neuro.11.2008.01.136.

[15] P. Kvello, T. Sneltvedt, K.E. Haugstad, K. Feren, J.T. Malmo, J. Cyvin, From single neuron to brain function – a neural network building kit developed to fill in the missing link in school, 2017, (NTNU Nordic Research Symposium on Science Education, Trondheim).

[16] T. Baden, B. James, M.J.Y. Zimmermann, P. Bartel, D. Grijseels, T. Euler, L. Lagnado, M. Maravall, Spikeling: a low-cost hardware implementation of a spiking neuron for neuroscience teaching and outreach, PLoS Biol. 16 (10) (2018) e2006760, https://doi.org/10.1371/journal.pbio.2006760.

[17] J.R. Burdo, NeuroBytes electronic neuron simulators and the 2017 FUN summer workshop, J. Undergrad. Neurosci. Educ. 16 (3) (2018) A232–A235.

[18] R. Renault, Neurino, 2020. Available: https://hackaday.io/project/17060 3-neurino. [Online]. (visited on 03/14/2024).

4. The Yin-Yang dataset

The Yin-Yang benchmark for spatio-temporal algorithms was published as

> Laura Kriener, **Julian Göltz** and Mihai A. Petrovici (Mar. 2022). 'The Yin-Yang Dataset'. In: *Neuro-Inspired Computational Elements Conference.* NICE 2022. Virtual Event, USA: Association for Computing Machinery, pp. 107–111. DOI: `10.1145/3517343.3517380`.

Author contributions

LK is the first author of this publication. All three authors designed the project together, with LK writing the software and performing the experiments, except Figure 4 which was created by JG. The manuscript was written by LK, JG, and MAP.

Manuscript

In accordance with *ACM* policies,[18] in the following the manuscript is reprinted.

[18] `https://authors.acm.org/author-resources/author-rights`

The Yin-Yang dataset

Laura Kriener
laura.kriener@unibe.ch
Department of Physiology,
University of Bern
Switzerland

Julian Göltz
julian.goeltz@kip.uni-heidelberg.de
Kirchhoff-Institute for Physics,
Heidelberg University
Germany
Department of Physiology,
University of Bern
Switzerland

Mihai A. Petrovici
mihai.petrovici@unibe.ch
Department of Physiology,
University of Bern
Switzerland
Kirchhoff-Institute for Physics,
Heidelberg University
Germany

ABSTRACT

The Yin-Yang dataset was developed for research on biologically plausible error backpropagation and deep learning in spiking neural networks. It serves as an alternative to classic deep learning datasets, especially in early-stage prototyping scenarios for both network models and hardware platforms, for which it provides several advantages. First, it is smaller and therefore faster to learn, thereby being better suited for small-scale exploratory studies in both software simulations and hardware prototypes. Second, it exhibits a very clear gap between the accuracies achievable using shallow as compared to deep neural networks. Third, it is easily transferable between spatial and temporal input domains, making it interesting for different types of classification scenarios.

CCS CONCEPTS

• **Networks** → **Network performance evaluation**; • **Computing methodologies** → *Bio-inspired approaches*; • **Hardware** → **Functional verification**.

ACM Reference Format:
Laura Kriener, Julian Göltz, and Mihai A. Petrovici. 2022. The Yin-Yang dataset. In *Neuro-Inspired Computational Elements Conference (NICE 2022), March 28-April 1, 2022, Virtual Event, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3517343.3517380

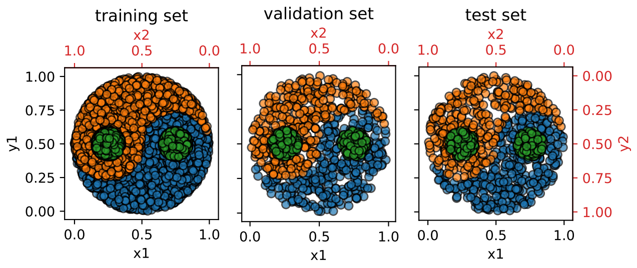

Figure 1: Training, validation and test dataset. Each dot in the yin-yang symbol represents one sample of the dataset. The color of the dot denotes its class (“Yin”, “Yang” or “Dot”). This figure was generated using the default settings for random seeds and dataset sizes (5000 samples for the training set and 1000 samples each for the validation and test set).

1 INTRODUCTION

We introduce the Yin-Yang dataset for learning in hierarchical networks [11]. It is tailored to the requirements of research on biologically plausible error backpropagation algorithms, learning in spiking neural networks and hierarchical networks on neuromorphic hardware. These fields typically require small but at the same time not trivially solvable datasets to prototype and test network architectures and learning algorithms. Setups commonly used for this purpose involve either elementary logic tasks such as XOR or small-scale datasets such as MNIST or fashion-MNIST [12, 22] and reduced versions thereof. However, these setups often do not adequately fulfill their purpose. Binary XOR only has a tiny number of input patterns and therefore a very limited, discrete set of reachable accuracies, making the evaluation and comparison of learning algorithms difficult. In turn, MNIST-type datasets have other drawbacks. For one, they require comparatively large networks, which might not be feasible during prototyping. But even more importantly, and despite this ostensible difficulty, they can nevertheless be classified with high accuracy even by shallow networks or networks without learning in the lower layers. This is problematic because training a deep network with an imperfect learning algorithm can result in performance indistinguishable from that of a shallow network or a network with plasticity only in the last layer. Conversely, a test on the MNIST dataset can fail to reveal the inability of the training algorithm to propagate error signals through the network, as the achieved high accuracies obscure the underlying problem.

The Yin-Yang dataset can provide an alternative for these testing and prototyping scenarios as it is solvable by smaller networks, contains fewer samples and most importantly exhibits a large gap between the accuracies reached by shallow or partly fixed networks on the one hand and correctly trained deep networks on the other. Note that here, we use “deep” in opposition to “shallow”, i.e., any network that has latent variables through which errors need to propagate. We consider a shallow network to be the equivalent of a single-layer perceptron, with only an input layer connected directly to a label layer.

NICE 2022, March 28-April 1, 2022, Virtual Event, USA
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9559-5/22/03.
https://doi.org/10.1145/3517343.3517380

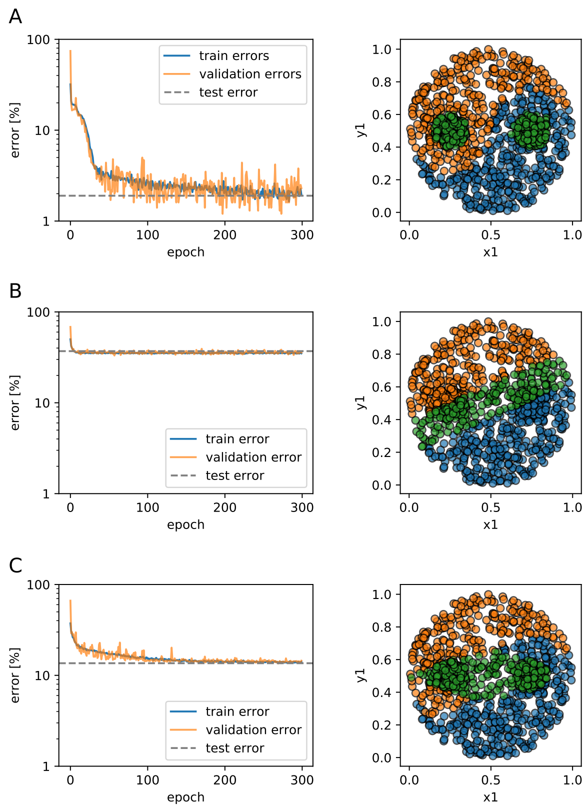

Figure 2: Comparison of exemplary training results for different network setups. Network parameters are given in Table 2. Left column: Evolution of the validation and training error during training. Right column: training result illustrated on the test set. (A) Network with one hidden layer and fully functional synaptic plasticity via classical error backpropagation. (B) Shallow network. (C) Network with one hidden layer and frozen lower weights.

2 DATASET

Each sample in the dataset represents a point in a two-dimensional representation of the yin-yang symbol. Depending on their location in the symbol the samples are classified into the "Yin", "Yang" or "Dot" class (Fig. 1). Even though the areas in the yin-yang symbol covered by the different classes have different sizes, the dataset is designed to be balanced, which means that all classes are represented by approximately the same amount of samples. Note that therefore the density of samples is higher in the "Dot"-class regions, as the combined area of these regions is smaller than that of the others.

The samples are randomly generated using rejection sampling. The exact version of a generated set of samples is therefore determined by the random seed and dataset size. This makes it possible to produce multiple dataset versions by providing different random seeds and dataset sizes. In the default configuration the training set has 5000 samples while the validation and test sets have 1000 samples respectively, each generated with a different random seed.

As can be seen in Fig. 1, values of all samples in the dataset are strictly positive. This is the case to accommodate network models which require positive input values only (common in the field of biologically-plausible networks, as firing rates as well as spike times are typically denoted by positive numbers). Because of that the yin-yang symbol is not centered around zero. This however complicates training in neuron models without intrinsic (learnable) bias. To facilitate training for these models, each sample in the dataset consists not only of the coordinates (x, y) determining the position in the yin-yang symbol but additionally also the values $(1 - x, 1 - y)$. This effectively symmetrizes the input and removes the need for a bias even though the yin-yang symbol is not centered around the origin of the coordinate system.

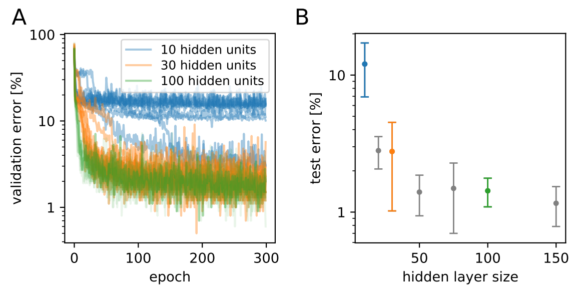

Figure 3: Impact of hidden layer size on network performance. (A) Validation errors during training for three different network architectures with different hidden layer sizes. For each architecture, ten training runs with different random weight initialization are overlaid. (B) Mean and standard deviation of the final test error depending on hidden layer size of the network. The colored data points correspond to the runs shown in A.

3 TRAINING RESULTS

As a baseline for further applications of this dataset we also provide some training results achieved with classical artificial neural networks. In particular, we compare network performance in three scenarios:

(1) a network with one hidden layer and fully functional error backpropagation;
(2) a shallow network with only an input and an output layer;
(3) a network with one hidden layer, but with frozen weights between the input and the hidden layer to emulate training with a faulty error backpropagation algorithm.

For all scenarios, we use very small network sizes to emulate a model or hardware prototyping environment. Incidentally, this is also helpful in highlighting another problem that is frequently overlooked when increasing the network size: because a large enough hidden layer can mask faulty error backpropagation, larger-scale networks are often inadequate for a quantitative verification of credit assignment (precise error propagation) within the studied network model. This is discussed below in more detail.

Table 1: Mean and standard deviation of the test accuracy for 20 training runs with different random initializations for different network configurations. Training parameters can be found in Table 2.

network	hidden layer with 20 neurons	hidden layer with 30 neurons
deep network	(97.0 ± 1.6) %	(97.6 ± 1.5) %
deep network (frozen lower weights)	(78.3 ± 7.8) %	(85.5 ± 5.8) %
shallow network	(63.8 ± 1.0) %	

The comparison between the three scenarios (Table 1 and Fig. 2) illustrates a manifest advantage of the Yin-Yang dataset compared to other commonly used datasets of comparable size: both the shallow network and the one with the frozen lower weights are clearly unable to learn the required features to successfully classify the dataset. This leads to a gap of more than 30 % between the accuracies achieved by a shallow and a deep network.

The failure of the partially frozen network highlights another important issue for various proposals of bio-plausible solutions to the credit assignment problem. In large enough networks, the large hidden layers project the input into a very high-dimensional space, which makes classification tasks more easily solvable by the linear classifier embodied by the top layer. This is commonly referred to as the "kernel trick" (see e.g. [19]). This can easily mask the inability of a network to correctly propagate errors and perform true gradient descent learning. While this issue would become observable when dealing with more complicated classification problems, it would require using large, deep networks that are not only difficult to debug but, more importantly, would lie beyond the capabilities of typical prototype devices or software simulations.

The Yin-Yang dataset addresses both problems simultaneously, by clearly highlighting faulty error backpropagation already within resource-efficient implementations with hidden layer sizes of around 20 to 30 neurons (see Table 1). Under these circumstances, the difference between the accuracy reached by a properly trained network and the network where only the top weights are trained lies around 20 % and 12 % respectively. This is a much higher gap than in a comparable example with the MNIST dataset, where networks need several hundred hidden neurons to show significant performance improvements beyond linear classifiers [12]. However, such sizes automatically introduce the kernel trick: a network with 500 hidden units reaches on average 98.3 % on MNIST, while the same network with only training in the top layer reaches 94.8 %. Unmasking these issues can become crucial in research on biologically plausible forms of credit assignment and (local) synaptic plasticity, where exact error backpropagation is notoriously difficult to realize, both for rate-based models and, even more pronouncedly, for spiking networks.

Another advantage of the Yin-Yang dataset over many other commonly used datasets is the dimensionality of its samples and the network sizes required to learn the task. Each sample consists of only four input values (compared to, e.g., the 784 input channels required by MNIST), which significantly reduces the required fan-in for hidden neurons. This can be especially beneficial on neuromorphic platforms, where the number of synaptic connections to a neuron is very often limited by the chip architecture, even more so for early-stage prototypes (e.g. [2, Section 3.3], [1, 5, 13, 14, 17]).

Also, this dataset can be learned with a single hidden layer of reasonably small size (Fig. 3). For consistently high final accuracies, a hidden layer of 20 to 30 neurons is required, but for a small proof-of-concept demonstration of a learning algorithm or hardware prototype, even 10 hidden units are enough to achieve results (around 88 % accuracy) that would be impossible with shallow networks, or with algorithms that cannot profit from a network's representational hierarchy. The full set of training parameters can be found in Table 2.

In addition to the results shown here, the dataset has already been used to showcase algorithms for error backpropagation in spiking neural networks in [7] and [21].

Table 2: Training parameters used to produce the results in Fig. 2. Fig. 3 uses the same parameters except for the size of the hidden layer.

parameter name	value
activation function	ReLU
size input	4
size hidden layer (for deep net)	30
size output layer	3
training epochs	300
batch size	20
optimizer	Adam, [10]
Adam parameter β	$(0.9, 0.999)$
Adam parameter ϵ	10^{-8}
learning rate	0.01

4 INPUT ENCODING

The Yin-Yang dataset can be adapted to suit the needs of very different network models. Depending on the used network architecture, neuron model and mode of communication between the neurons, different types of information encoding become necessary. In the following, we discuss several encoding methods that are well-suited for a variety of different network and neuron types.

4.1 Spatio-temporal input encoding

Using this dataset for spiking neural networks requires an explicit spatio-temporal input encoding. In [7] and [21], the four input features of the dataset were directly interpreted as the spike times of 4 input neurons (Fig. 4 A). This was done by choosing parameters t_{early} and t_{late} as the earliest and latest possible time the input neurons are allowed to spike. Then the dataset values $\vec{x} = (x, y, 1 -$

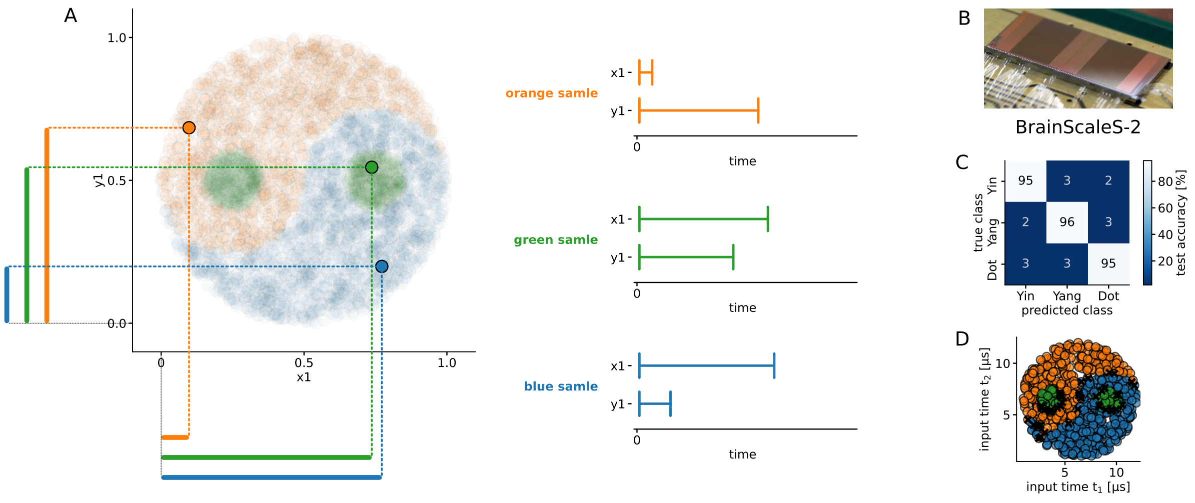

Figure 4: Spatio-temporal input encoding scheme and classification results on the neuromorphic chip BrainScaleS-2. Panels (B-D) adapted from [6]. (A) Encoding of the x, y-coordinates of the Yin-Yang pattern as input spike times t_1 and t_2 illustrated on one sample each for the three classes. (B) Image of the BrainScaleS-2 ASIC. (C) Confusion matrix after training the BrainScaleS-2 chip to classify the Yin-Yang dataset. (D) Classification result of the chip on the test set. For each input sample the color indicates the class determined by the trained network. Wrong classifications are marked with a black X. The wrongly classified samples all lie very close to the border between two classes.

$x, 1-y$) were translated into the four spike times $\vec{t} = (t_1, t_2, t_3, t_4)$ as follows:

$$\vec{t} = t_{\text{early}} + \vec{x} \cdot \left(t_{\text{late}} - t_{\text{early}}\right) \tag{1}$$

The choice of $t_{\text{early/late}}$ is dependent on the network architecture and employed learning algorithms. For [7] it has proven beneficial to choose t_{early} slightly after the start of the experiment and t_{late} as the sum of the two neuron time constants $t_{\text{late}} \approx \tau_{\text{m}} + \tau_{\text{syn}}$. The classification results achieved with the BrainScaleS-2 chip are shown in Fig. 4.

Alternatively, a different spike-based spatio-temporal encoding can be achieved implicitly by manipulating input currents, as proposed for example in [3]. Here, each input variable is interpreted as the strength of a constant input current into a leaky-integrate and fire neuron. The timing of the output spike of the input neurons depends on the strength of the input current I with

$$t_{\text{spike}} = \tau_m \log \frac{I}{I - \theta_{\text{I}}} \tag{2}$$

where τ_m denotes the membrane time constant and θ_{I} the minimal current necessary to evoke an output spike.

4.2 Rate-based input encoding

Many models for biologically plausible error backpropagation are built around rate-based neuron models (e.g. [9, 15, 16], for a review see also [20]). These approaches use continuous rates as an idealized version of rate coding in spiking neurons. Others build on the same approximations but explicitly use spike-based communication in their neural network implementations (e.g. [4, 8, 18]). For such rate-based models, a suitable encoding scheme can be easily realized by designating 4 input neurons and setting their output rates proportional to the values of the respective input feature.

In case of spiking neurons, these four input neurons can produce Poisson spike trains with the same rates as their rate-based counterparts, as, for example, in [18]. Alternatively, regular spike trains could also be used to represent firing rates; while more precise than the intrinsicly stochastic Poisson solution, this scheme has its own potential drawback of making the neuronal input-output function dependent on not just the rate, but also the phase of a neuron's afferents. Under certain circumstances, encoding an input as a single neuron may not be viable, for example when synaptic bandwidth or neuron firing rate are limited. In this case, one input can be represented by a population of neurons with a mean firing rate equal to the value of the input.

Code and data availability

Code for the Yin-Yang data set is available at https://github.com/lkriener/yin_yang_data_set. The example notebook in the repository includes the plotting of the data samples (Fig. 1) and the training of deep and shallow networks (Fig. 2). Additional data available on request from the authors.

REFERENCES

[1] Sebastian Billaudelle, Yannik Stradmann, Korbinian Schreiber, Benjamin Cramer, Andreas Baumbach, Dominik Dold, Julian Göltz, Akos F Kungl, Timo C Wunderlich, Andreas Hartel, et al. 2020. Versatile emulation of spiking neural networks on an accelerated neuromorphic substrate. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1–5.
[2] Jonathan Binas, Daniel Neil, Giacomo Indiveri, Shih-Chii Liu, and Michael Pfeiffer. 2016. Precise neural network computation with imprecise analog devices. *arXiv preprint arXiv:1606.07786* (2016).

[3] Benjamin Cramer, Sebastian Billaudelle, Simeon Kanya, Aron Leibfried, Andreas Grübl, Vitali Karasenko, Christian Pehle, Korbinian Schreiber, Yannik Stradmann, Johannes Weis, et al. 2020. Surrogate gradients for analog neuromorphic computing. *arXiv preprint arXiv:2006.07239* (2020).

[4] Steve K Esser, Rathinakumar Appuswamy, Paul Merolla, John V Arthur, and Dharmendra S Modha. 2015. Backpropagation for energy-efficient neuromorphic computing. *Advances in Neural Information Processing Systems* (2015), 1117–1125.

[5] Charlotte Frenkel, Martin Lefebvre, Jean-Didier Legat, and David Bol. 2018. A 0.086-mm ^{2} 12.7-pJ/SOP 64k-synapse 256-neuron online-learning digital spiking neuromorphic processor in 28-nm CMOS. *IEEE transactions on biomedical circuits and systems* 13, 1 (2018), 145–158.

[6] Julian Göltz, Laura Kriener, Andreas Baumbach, Sebastian Billaudelle, Oliver Breitwieser, Benjamin Cramer, Dominik Dold, Akos Ferenc Kungl, Walter Senn, Johannes Schemmel, Karlheinz Meier, and Mihai A Petrovici. 2019. Fast and deep: energy-efficient neuromorphic learning with first-spike times. *arXiv:1912.11443* (2019).

[7] Julian Göltz, Laura Kriener, Andreas Baumbach, Sebastian Billaudelle, Oliver Breitwieser, Benjamin Cramer, Dominik Dold, Akos Ferenc Kungl, Walter Senn, Johannes Schemmel, Karlheinz Meier, and Mihai A Petrovici. 2021. Fast and energy-efficient neuromorphic deep learning with first-spike times. *Nature Machine Intelligence* 3, 9 (2021), 823–835.

[8] Jordan Guerguiev, Timothy P Lillicrap, and Blake A Richards. 2017. Towards deep learning with segregated dendrites. *Elife* 6 (2017), e22901.

[9] Paul Haider, Benjamin Ellenberger, Laura Kriener, Jakob Jordan, Walter Senn, and Mihai A Petrovici. 2021. Latent Equilibrium: Arbitrarily fast computation with arbitrarily slow neurons. *Advances in Neural Information Processing Systems* 34 (2021).

[10] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980* (2014).

[11] Laura Kriener, Julian Göltz, and Mihai A Petrovici. 2021. Yin-Yang dataset repository. https://github.com/lkriener/yin_yang_data_set. Accessed: 2021-01-20.

[12] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[13] Saber Moradi and Giacomo Indiveri. 2013. An event-based neural network architecture with an asynchronous programmable synaptic memory. *IEEE transactions on biomedical circuits and systems* 8, 1 (2013), 98–107.

[14] Manu V Nair and Giacomo Indiveri. 2019. An ultra-low power sigma-delta neuron circuit. In *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1–5.

[15] João Sacramento, Rui Ponte Costa, Yoshua Bengio, and Walter Senn. 2018. Dendritic cortical microcircuits approximate the backpropagation algorithm. *Advances in Neural Information Processing Systems* 31 (2018).

[16] Benjamin Scellier and Yoshua Bengio. 2017. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience* 11 (2017), 24.

[17] Johannes Schemmel, Laura Kriener, Paul Müller, and Karlheinz Meier. 2017. An accelerated analog neuromorphic hardware system emulating NMDA-and calcium-based non-linear dendrites. In *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2217–2226.

[18] Sebastian Schmitt, Johann Klähn, Guillaume Bellec, Andreas Grübl, Maurice Güttler, Andreas Hartel, Stephan Hartmann, Dan Husmann, Kai Husmann, Sebastian Jeltsch, et al. 2017. Neuromorphic hardware in the loop: Training a deep spiking network on the brainscales wafer-scale system. *2017 International Joint Conference on Neural Networks (IJCNN)* (2017), 2227–2234.

[19] Bernhard Scholkopf. 2001. The kernel trick for distances. *Advances in neural information processing systems* (2001), 301–307.

[20] James CR Whittington and Rafal Bogacz. 2019. Theories of error back-propagation in the brain. *Trends in cognitive sciences* 23, 3 (2019), 235–250.

[21] Timo C Wunderlich and Christian Pehle. 2021. Event-based backpropagation can compute exact gradients for spiking neural networks. *Scientific Reports* 11, 1 (2021), 1–17.

[22] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv:1708.07747* (2017).

ACKNOWLEDGMENTS

We wish to thank Sebastian Billaudelle and Benjamin Cramer for valuable discussions, as well as Mike Davies and Intel for their ongoing support. We gratefully acknowledge funding from the European Union under grant agreements 604102, 720270, 785907, 945539 (HBP) and the Manfred Stärk Foundation.

During the development of the dataset some calculations were performed on UBELIX, the HPC cluster at the University of Bern, others were performed on the bwForCluster NEMO, supported by the state of Baden-Württemberg through bwHPC and the German Research Foundation (DFG) through grant no INST 39/963-1 FUGG. Additionally, our work has greatly benefitted from access to the Fenix Infrastructure resources, which are partially funded from the European Union's Horizon 2020 research and innovation programme through the ICEI project under the grant agreement No. 800858.

5. Fast and energy-efficient neuromorphic deep learning with first-spike times

This chapter concerns the manuscript published as

> **Julian Göltz***, Laura Kriener*, Andreas Baumbach, Sebastian Billaudelle, Oliver Breitwieser, Benjamin Cramer, Dominik Dold, Akos Ferenc Kungl, Walter Senn, Johannes Schemmel, Karlheinz Meier and Mihai A Petrovici (Sept. 2021). 'Fast and energy-efficient neuromorphic deep learning with first-spike times'. In: *Nature Machine Intelligence* 3.9, pp. 823–835. DOI: `10.1038/s42256-021-00388-x`.

Author contributions

JG and LK both are first authors of this publication. The *Author Contributions* section of the paper reads:

> JG, AB and MAP designed the conceptual and experimental approach. JG derived the theory, implemented the algorithm, and performed the hardware experiments. LK embedded the algorithm into a comprehensive training framework and performed the simulation experiments. AB and OJB offered substantial software support. SB, BC, JG and AFK provided low-level software for interfacing with the hardware. JG, LK, DD, SB and MAP wrote the manuscript.

To elaborate on this, JG initially started this research project, both deriving the theory and implementing the training method. Generally, JG was crucially involved in every part of the project, specifically also in writing the manuscript.

Manuscript

In accordance with *Springer Nature* policies,[19] in the following the manuscript is reprinted.

[19] `https://www.nature.com/nature-portfolio/reprints-and-permissions/permissions-requests`

https://doi.org/10.1038/s42256-021-00388-x

Check for updates

Fast and energy-efficient neuromorphic deep learning with first-spike times

J. Göltz[1,2,4] ✉, L. Kriener[2,4] ✉, A. Baumbach[1], S. Billaudelle[1], O. Breitwieser[1], B. Cramer[1], D. Dold[1,3], A. F. Kungl[1], W. Senn[2], J. Schemmel[1], K. Meier[1] and M. A. Petrovici[1,2] ✉

For a biological agent operating under environmental pressure, energy consumption and reaction times are of critical importance. Similarly, engineered systems are optimized for short time-to-solution and low energy-to-solution characteristics. At the level of neuronal implementation, this implies achieving the desired results with as few and as early spikes as possible. With time-to-first-spike coding, both of these goals are inherently emerging features of learning. Here, we describe a rigorous derivation of a learning rule for such first-spike times in networks of leaky integrate-and-fire neurons, relying solely on input and output spike times, and show how this mechanism can implement error backpropagation in hierarchical spiking networks. Furthermore, we emulate our framework on the BrainScaleS-2 neuromorphic system and demonstrate its capability of harnessing the system's speed and energy characteristics. Finally, we examine how our approach generalizes to other neuromorphic platforms by studying how its performance is affected by typical distortive effects induced by neuromorphic substrates.

In recent years, the machine learning landscape has been dominated by deep learning methods. Among the benchmark problems they have managed to crack, some remained elusive for a long time[1–3]. It is thus not an exaggeration to say that deep learning dominates our understanding of 'artificial intelligence'[4–8].

Compared to the abstract neural networks used in deep learning, the more biological archetypes—spiking neural networks—still lag behind in terms of performance and scalability[9]. The reasons for this difference in success are numerous; for example, unlike abstract neurons, even an individual biological neuron represents a complex system, with finite response times, membrane dynamics and spike-based communication[10,11], making it more challenging to find reliable coding and computation paradigms[12–14]. Furthermore, one of the major driving forces behind the success of deep learning, the backpropagation of errors algorithm[15–17], has remained incompatible with spiking neural networks until only very recently[18,19].

Despite these challenges, spiking neural networks promise to present some important advantages. The time information inherent to spikes allows a coding scheme for spike-based communication that utilizes both spatial and temporal dimensions[20], unlike spike-count-based approaches[21–24], where the information of spike times is at least partially diluted due to temporal or population averaging. Owing to the inherent parallelism of all biological, as well as many biologically inspired, spiking neuromorphic systems[25], this promises fast, sparse and energy-efficient information processing, and provides a blueprint for computing architectures that could one day rival the efficiency of the brain itself[9,25–27]. This makes spiking neural networks implemented on specialized neuromorphic devices potentially more powerful—at least in principle—than the 'conventional', simple machine learning models currently used on von Neumann machines, even though this potential still remains mostly unexploited[9].

Many attempts have been made to reconcile spiking neural networks with their abstract counterparts in terms of functionality, for example, by featuring spike-based inference models[28–36] and deep models trained on target spike times by shallow learning rules[37,38] or using spike-compatible versions of the error backpropagation algorithm[39–41]. Especially for tasks operating on static information, a particularly elegant way of utilizing the temporal aspect of exact spike times is the time-to-first-spike (TTFS) coding scheme[42]. Here, a neuron encodes its real-valued response to a stimulus as the time elapsed before its first spike in reaction to that stimulus. Such single-spike coding enables fast information processing by explicitly encouraging the emission of as few spikes as early as possible, which meets the physiological constraints and reaction times observed in humans and animals[42–45]. Apart from biological plausibility, such a fast and sparse coding scheme is a natural fit for neuromorphic systems that offer energy-efficient and fast emulation of spiking neural networks[46–52].

For hierarchical TTFS networks, a gradient-descent-based learning rule was proposed in refs. [53,54], using error backpropagation on a continuous function of output spike times. However, this approach is limited to a neuron model without leak, which is neither biologically plausible nor compatible with most analogue very-large-scale integration (VLSI) neuron dynamics[25]. We propose a solution for leaky integrate-and-fire (LIF) neurons with current-based (CuBa) synapses—a widely used dynamical model of spiking neurons with realistic integration behaviour[55–57]. An early version of this work was presented in ref. [58].

For several specific configurations of time constants, we provide analytical expressions for first-spike timing, which, in turn, allow the calculation of exact gradients of any differentiable cost function that depends on these spike times. In hierarchical networks of LIF neurons using the TTFS coding scheme, this enables exact error backpropagation, allowing us to train such networks as universal classifiers on both continuous and discrete data spaces.

As our algorithm only requires knowledge about the afferent and efferent spike times of all neurons, it lends itself to emulation

[1]Kirchhoff-Institute for Physics, Heidelberg University, Heidelberg, Germany. [2]Department of Physiology, University of Bern, Bern, Switzerland. [3]Siemens AI lab, Siemens AG Technology, Munich, Germany. [4]These authors contributed equally: J. Göltz, L. Kriener. ✉e-mail: julian.goeltz@kip.uni-heidelberg.de; laura.kriener@unibe.ch; mihai.petrovici@unibe.ch

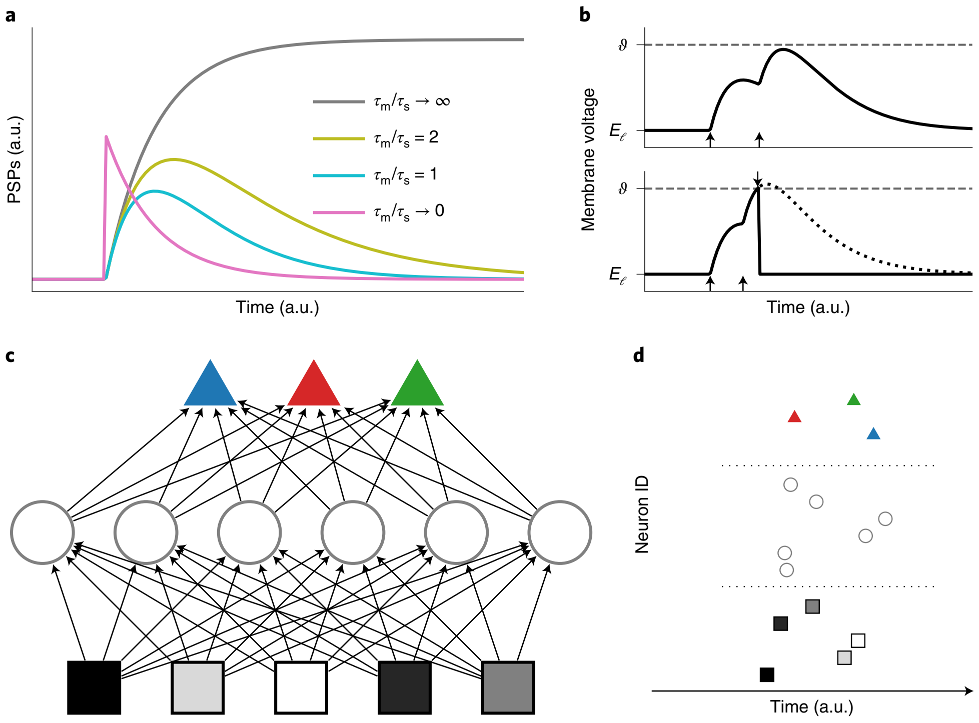

Fig. 1 | Time-to-first-spike coding and learning. a, Postsynaptic potential (PSP) shapes for different ratios of time constants τ_s and τ_m for single neurons. The finiteness of time constants causes the neuron to gradually forget prior input. **b**, One key challenge of this finite memory arises when small variations of the synaptic weights result in disappearing/appearing output spikes, which elicits a discontinuity in the function describing output spike timing. Plots for single neurons are shown in **a** and **b**. **c**,**d**, Application to feedforward hierarchical networks. **c**, Network structure. The geometric shape of the neurons represents a notation of their respective types (input, squares; hidden, circles; label, triangles). The shading of the input neurons (black, grey and white squares) represents the corresponding data, such as pixel brightness. The colour of the label neurons represents their respective class (blue, red, green triangles). **d**, TTFS coding exemplified in a raster plot. As an example of input encoding, the brightness of an input pixel is encoded in the lateness of a spike. Note that, in our framework, TTFS coding simultaneously refers to two individual aspects, namely the input-to-spike-time conversion and the determination of the inferred class by the identity of the first label neuron to fire (red triangle). a.u., arbitrary units.

on neuromorphic hardware. The accelerated, yet power-efficient BrainScaleS-2 platform[48,59] pairs especially well with the sparseness and low latency already inherent to TTFS coding. We show how an implementation of our algorithm on BrainScaleS-2 can obtain similar classification accuracies to software simulations, while displaying highly competitive time and power characteristics, with a combination of 48 µs and 8.4 µJ per classification.

By incorporating information generated on the hardware for updates during training, the algorithm automatically adapts to potential imperfections of neuromorphic circuits, as implicitly demonstrated by our neuromorphic implementation. In further software simulations, we show that our model deals well with various levels of substrate-induced distortions such as fixed-pattern noise and limited parameter precision and control, thus providing a rigorous algorithmic backbone for a wide range of neuromorphic substrates and applications. Such robustness with respect to imperfections of the underlying neuronal substrate represents an indispensable property for any network model aiming for biological plausibility and for every application geared towards physical computing systems[33,34,60–64].

In the following, we first introduce the CuBa LIF model and the TTFS coding scheme, before we demonstrate how both inference and training via error backpropagation can be performed analytically with such dynamics. Finally, the presented model is evaluated both in software simulations and neuromorphic emulations, before studying the effects of several types of substrate-induced distortion.

Results

Leaky integrate-and-fire dynamics. The dynamics of an LIF neuron with CuBa synapses is given by

$$C_m \dot{u}(t) = g_\ell [E_\ell - u(t)] + \sum_i w_i \sum_{t_i} \theta(t - t_i) \exp\left(-\frac{t - t_i}{\tau_s}\right), \quad (1)$$

with membrane capacitance C_m, leak conductance g_ℓ (from which the membrane time constant $\tau_m = C_m/g_\ell$ follows), weights w_i and spike times t_i of presynaptic neuron i, synaptic time constant τ_s and where θ is the Heaviside step function. The first sum runs over all presynaptic neurons and the second over all spikes for each presynaptic neuron. The neuron elicits a spike at time T when the presynaptic input pushes the membrane potential above a threshold ϑ. After spiking, a neuron becomes refractory for a time period τ_{ref}, which is modelled by clamping its membrane potential to a reset value ϱ: $u(t') = \varrho$ for $T \leq t' \leq T + \tau_{ref}$. For convenience and without loss of generality, we set the leak potential $E_\ell = 0$. Equation (1) can be solved analytically and yields subthreshold dynamics as described by equation (9). The choice of τ_m and τ_s ultimately influences the shape of a postsynaptic potential (PSP), starting from a simple exponential ($\tau_m \ll \tau_s$), to a difference of exponentials (with an alpha function for the special case of $\tau_m = \tau_s$) and to a graded step function ($\tau_m \gg \tau_s$) (Fig. 1a). Note that all of these scenarios are conserved under exchange of τ_s and τ_m, as is apparent from the symmetry of the analytical solution (equation (9)).

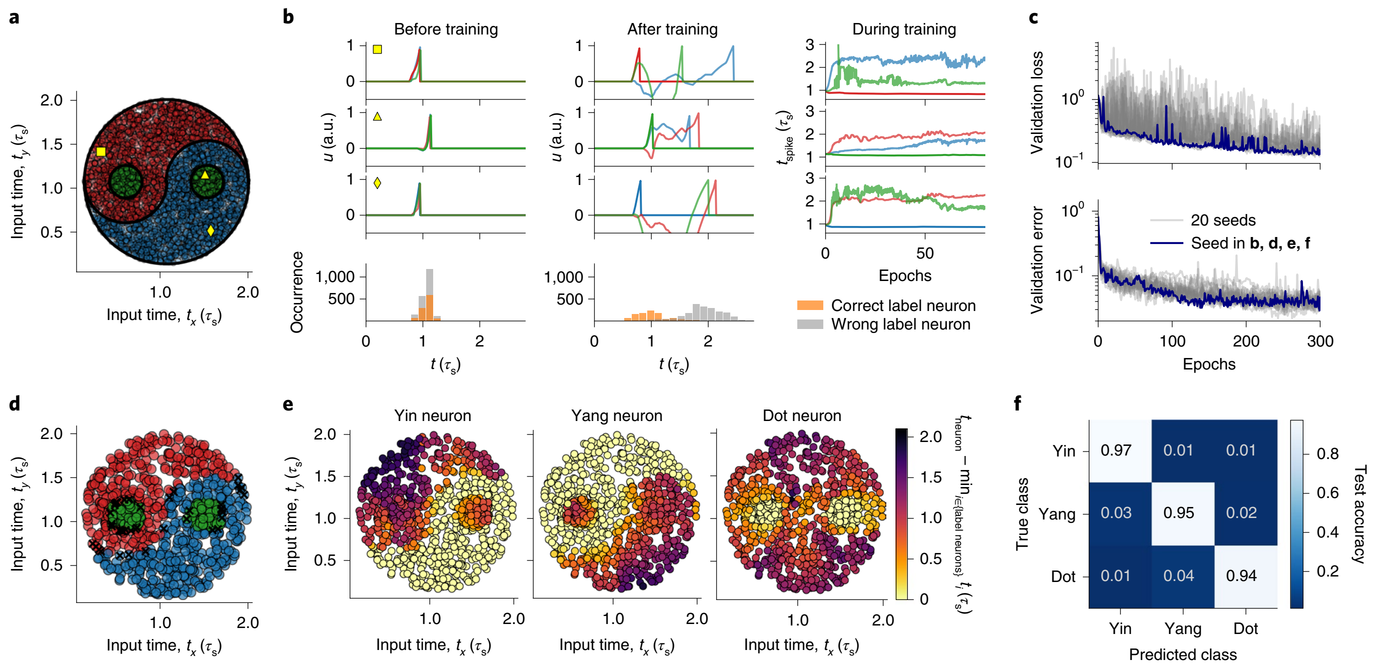

Fig. 2 | Classification of the Yin-Yang dataset. a, Illustration of the Yin-Yang dataset. The samples are separated into three classes, Yin (blue circles), Yang (red circles) and Dot (green circles). The yellow symbols (square, triangle and diamond) mark samples for which the training process is illustrated in **b**. Input times t_x and t_y correspond to the spike time of the inputs associated with the x and y coordinates of individual samples. Units of τ_s indicate times that are measured in multiples of the synaptic time constant, τ_s. **b**, Training mechanism for three example data samples (cf. **a**). For the first three rows, the left and middle columns depict voltage dynamics in the label layer before and after training for 300 epochs, respectively. The voltage traces of the three label neurons are colour-coded according to their corresponding class as in **a**. Before training, the random initialization of the weights causes the label neurons to show similar voltage traces and almost indistinguishable spike times. After training, there is a clear separation between the spike time of the correct label neuron and all others, with the correct neuron spiking first. The evolution of the label spike times during training is shown in the right column for the first 70 epochs. Bottom row: spike histograms over all training samples. Our learning algorithm induces a clear separation between the spike times of correct and wrong label neurons. **c**, Training progress (validation loss as given in equation (6) and error rate) over 300 epochs for 20 training runs with random initializations (grey). The run shown in **b** and **d**–**f** is plotted in blue. **d**, Classification result on the test set (1,000 samples). The colour of each sample indicates the class determined by the trained network. The wrongly classified samples (marked with black X) all lie very close to the border between classes. **e**, Spike times of the Yin, Yang and Dot neurons for all test samples after training. For each sample, spike times were normalized by subtracting the earliest spike time in the label layer. Bright yellow denotes zero difference, that is, the respective label neuron was the first to spike and the sample was assigned to its class. The bright yellow areas resemble the shapes of the Yin, Yang and Dot areas, reflecting the high classification accuracy after training. **f**, Confusion matrix for the test set after training.

The first two cases with finite membrane time constant τ_m are markedly different from the last one, which is also known as either the non-leaky integrate-and-fire (nLIF) or simply the integrate-and-fire (IF) model and was used in previous work[53]. In the nLIF model, input to the membrane is never forgotten until a neuron spikes, as opposed to the LIF model, where the PSP reaches a peak after finite time and subsequently decays back to its baseline. In other words, presynaptic spikes in the LIF model have a purely local effect in time, unlike in the nLIF model, where only the onset of a PSP is localized in time, but the postsynaptic effect remains forever, or until the postsynaptic neuron spikes. A pair of finite time constants thus assigns much more importance to the time differences between input spikes and introduces discontinuities in the neuronal output that make an analytical treatment more difficult (Fig. 1b).

First-spike times. Our spike-timing-based neural code follows an idea first proposed in ref. [53]. Unlike coding in artificial neural networks (ANNs), and different from spike-count-based codes in spiking neural networks (SNNs), this scheme explicitly uses the timing of individual spikes for encoding information. In TTFS coding, the presence of a feature in a stimulus is reflected by the timing of a neuron's first spike after the onset of the stimulus, with earlier spikes representing a more strongly manifested feature. This has the effect that important information inherently propagates quickly through the network, with potentially only few spikes needed for the network to process an input. Consequently, this scheme enables efficient processing of inputs, both in terms of time-to-solution and energy-to-solution (assuming the latter depends, in general on the total number of spikes and the time required for the network to solve, for example, an input classification problem).

To formulate the optimization of a first-spike time T as a gradient-descent problem, we derive an analytical expression for T. This is equivalent to finding the time of the first threshold crossing by solving $u(T)=\vartheta$ for T. Even though there is no general closed-form solution for this problem, analytical solutions exist for specific cases. For example, we show that (Methods)

$$T=\tau_s\left\{\frac{b}{a_1}-\mathcal{W}\left[-\frac{g_\ell\vartheta}{a_1}\exp\left(\frac{b}{a_1}\right)\right]\right\}\quad\text{for }\tau_m=\tau_s\qquad(2)$$

and

$$T=2\tau_s\ln\left[\frac{2a_1}{a_2+\sqrt{a_2^2-4a_1g_\ell\vartheta}}\right]\quad\text{for }\tau_m=2\tau_s,\qquad(3)$$

where $\mathcal{W}$ is the Lambert W function and using the shorthand notations a_n and b for sums over the set of causal presynaptic spikes

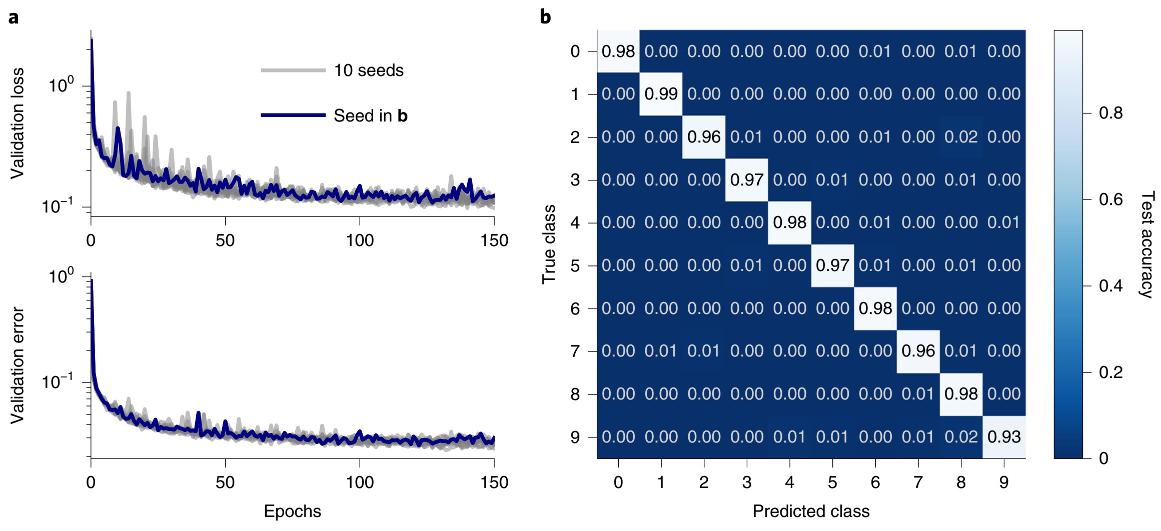

Fig. 3 | Classification of the MNIST dataset. a, Training progress of a network over 150 epochs for 10 different random initializations. The run drawn in blue is the one that produced the results in **b**. **b**, Confusion matrix for the test set after training.

$C=\{i|t_i<T\}$ (equations (11) and (12)). We note that, when calculating the output spike time for a large number of input neurons, determining C can be computationally intensive (Methods). One inherent advantage of physical emulation is the reduction of this calculational burden.

The above equations are differentiable with respect to synaptic weights and presynaptic spike times. As will be shown in the following, this directly translates to solving the credit assignment problem and thus allows exact error propagation through networks of spiking neurons. For easier reading, we focus on one specific case ($\tau_m=\tau_s$), but the others can be treated analogously.

Exact error backpropagation with spikes. Learning in SNNs requires the ability to relate efferent spiking to both afferent weights and spike times. For the output spike time of a neuron k with presynaptic partners i, the first relationship can be formally described by the derivative of the output spike time with respect to the presynaptic weights (equation (22)). Using certain properties of $\mathcal{W}$, we can find a simple expression that can also be made to depend on the output spike time t_k itself:

$$\frac{\partial t_k}{\partial w_{ki}} = -\frac{1}{a_1}\frac{\exp\left(\frac{t_i}{\tau_s}\right)}{\mathcal{W}(z)+1}\left(t_k-t_i\right), \tag{4}$$

with a_1 and z representing functions of w_{ki} and t_i as defined in equations (11) and (18). Using the output spike time as additional information optimizes learning in scenarios where the exact neuron parameters are unknown and the real output spike time differs from the one calculated under ideal assumptions, as discussed later.

Second, the capability to relate errors in the output spike time to errors in the input spike times allows us to recursively propagate changes from neurons to their presynaptic partners:

$$\frac{\partial t_k}{\partial t_i} = -\frac{1}{a_1}\frac{\exp\left(\frac{t_i}{\tau_s}\right)}{\mathcal{W}(z)+1}\frac{w_{ki}}{\tau_s}\left(t_k-t_i-\tau_s\right). \tag{5}$$

Together, equations (4) and (5) effectively and exactly solve the credit assignment problem in appropriately parametrized LIF networks of arbitrary architecture.

We can now apply the findings above to study learning in a layered network. Figure 1c shows a schematic of our feedforward networks and their spiking activity. The input uses the same coding scheme as all other neurons: more prominent features are encoded by earlier spikes. The output of the network is defined by the identity of the label neuron that spikes first (Fig. 1d).

We denote by $t_k^{(l)}$ the output spike time of the kth neuron in the lth layer. For example, in a network with N layers, $t_n^{(N)}$ is the spike time of the nth neuron in the label layer. The weight projecting to the kth neuron of layer l from the ith neuron of layer $l-1$ is denoted by $w_{ki}^{(l)}$.

To apply the error backpropagation algorithm[15,17], we choose a loss function that is differentiable with respect to synaptic weights and spike times. During learning, the objective is to maximize the temporal difference between the correct and all other label spikes. The following loss function fulfils the above requirements:

$$\begin{aligned} L[\mathbf{t}^{(N)},n^*] &= \operatorname{dist}\left(t_{n^*}^{(N)},t_{n\neq n^*}^{(N)}\right) \\ &= \log\left[\sum_n \exp\left(-\frac{t_n^{(N)}-t_{n^*}^{(N)}}{\xi\tau_s}\right)\right], \end{aligned} \tag{6}$$

where $\mathbf{t}^{(N)}$ denotes the vector of label spike times $t_n^{(N)}$, n^* the index of the correct label and $\xi\in\mathbb{R}^+$ is a scaling parameter. This loss function represents a cross entropy between the true label distribution and the softmax-scaled label spike times produced by the network (Methods). Reducing its value therefore increases the temporal difference between the output spike of the correct label neuron and all other label neurons. Notably, it only depends on the spike time difference and is invariant under absolute time shifts, making it independent of the concrete choice of the experiment start, which defines $t=0$. In the case of a non-spiking label neuron, we treat its spike time as $t_n^{(N)}=\infty$. In this case, however, equation (2) is not defined and neither are its derivatives. We therefore introduce a simple, local heuristic to encourage spiking behaviour in large portions of the network (Methods). In some scenarios, learning can be facilitated by the addition of a spike-time-dependent regularization term (Methods).

Gradient descent on the loss function equation (6) can now be easily performed by repeated application of the chain rule.

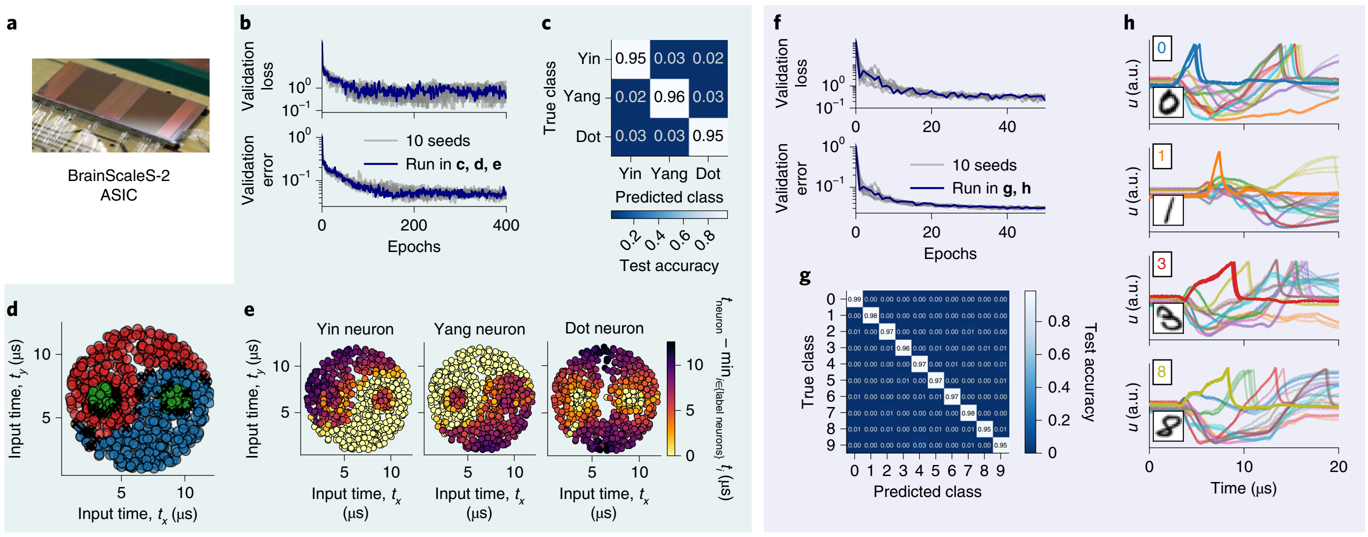

Fig. 4 | Classification on the BrainScaleS-2 neuromorphic platform. **a**, Photograph of a BrainScaleS-2 chip. **b–e**, Results for the Yin-Yang dataset. **b**, Training progress over 200 epochs for 11 different random initializations. The run drawn in blue also produced the results shown in **c–e**. **c**, Confusion matrix for the test set after training. **d**, Classification result on the test set. For each input sample the colour indicates the class determined by the trained network. Wrong classifications are marked with a black X. The wrongly classified samples all lie very close to the border between two classes. **e**, Separation of label spike times (cf. Fig. 2e). For each of the label neurons, bright yellow dots represent data samples for which it was the first to spike, thereby assigning them its class. Similarly to the software simulations, the bright yellow areas align well with the shapes of the Yin, Yang and Dot areas of the dataset. **f–h**, Results for the MNIST dataset. **f**, Evolution of training over 50 epochs for 10 different random initializations. The run drawn in blue is the one that produced the results shown in **g** and **h**. **g**, Confusion matrix for the test set after training. **h**, Exemplary membrane voltage traces on BrainScaleS-2 after training. Each panel shows colour-coded voltage traces of four label neurons for one input that was presented repeatedly to the network (the insets show the input and its correct class). Each trace was recorded four times to highlight the trial-to-trial variations.

Using the exact derivatives, equations (4) and (5), this yields the synaptic plasticity rule

$$\Delta w_{ki}^{(l)} \propto -\frac{\partial L[\mathbf{t}^{(N)}, n^*]}{\partial w_{ki}^{(l)}}$$
$$= -\frac{\partial t_k^{(l)}}{\partial w_{ki}^{(l)}} \underbrace{\frac{\partial L[\mathbf{t}^{(N)}, n^*]}{\partial t_k^{(l)}}}_{\delta_k^{(l)}} = -\frac{\partial t_k^{(l)}}{\partial w_{ki}^{(l)}} \sum_j \frac{\partial t_j^{(l+1)}}{\partial t_k^{(l)}} \delta_j^{(l+1)} . \quad (7)$$

A compact formulation for hierarchical networks that highlights the backpropagation of errors can be found in equations (38) to (40). In either form, only the label layer error and the neuron spike times are required for training, which can either be calculated using equation (2) or by simulating (or emulating) the LIF dynamics (equation (1)).

The computational complexity of the synaptic plasticity rule—a potential limiting factor for on-chip implementations—can be drastically reduced by appropriate approximations. In Supplementary Section D we present early results using such an approach. Note that the simplification is only used in Supplementary Section D and all other results we report in the following were produced using the full analytical equations (4) and (5).

Simulations. After deriving the learning algorithm in the previous chapter, we show its classification capabilities in software simulations. In these simulations we demonstrate successful learning and provide a baseline for the hardware emulations that follow. We use two datasets that emphasize different aspects of interesting real-world scenarios. As an example for low-dimensional, 'continuous' data spaces, in which points belonging to different classes can be arbitrarily close together (thus making separation particularly challenging), we chose the Yin-Yang dataset[65]. For higher-dimensional, discrete input, we used the MNIST dataset[66] as a small-scale image classification scenario.

The results in this section are based on equation (2) for calculating the spike times in the forward pass, and equation (40) for calculating weight updates. Details regarding implementation are provided in the Methods. For the hyperparameters of the discussed experiments, see Supplementary Tables F1 and F2.

Yin-Yang classification task. The first dataset consists of points in the yin-yang figure (Fig. 2a). Each point is defined by a pair of Cartesian coordinates $(x, y) \in [0, 1]^2$. To build in redundancy and capture the intrinsic symmetry of the yin-yang motive, the dataset is augmented with mirrored coordinates $(1 - x, 1 - y)$, enabling networks of neurons without trainable bias to learn the task[65]. The three classes are labelled according to the respective area they occupy, that is, Yin, Yang or Dot. This augmented dataset was specifically designed to require latent variables for classification: a shallow non-spiking classifier reaches $(64.3 \pm 0.2)\%$ test accuracy, an ANN with one hidden layer of size 120 typically around $(98.7 \pm 0.3)\%$. Because of this large gap, our Yin-Yang dataset represents an expressive test of error backpropagation in our hierarchical spiking networks. At the same time, it can be learned by networks that are compatible in size with the current revision of BrainScaleS-2 [67].

After translation of the four features to spike times (Fig. 1 and Methods), they were joined with a bias spike at fixed time, and these five spikes served as input to a network with 120 hidden and three label neurons. We illustrate the training mechanism with voltage traces for three samples belonging to different classes (Fig. 2b). The algorithm changes the weights to create a separation in the label spike times (cf. the left and middle column) that corresponds to correct classification. Note that the voltage traces were just recorded for illustration, as only spike times are required for calculating weight updates. After 300 epochs our networks reached $(95.9 \pm 0.7)\%$ test accuracy for training with 20 different random seeds (Fig. 2c). The classification failed only for samples that were extremely close to the border between two classes (Fig. 2d). Figure 2e shows the spike

Table 1 | Summary of the presented results

Dataset	Hidden neurons	Accuracy (%)	
		Test	Train
Yin-Yang			
In SW	120	95.9 ± 0.7	96.3 ± 0.7
On HW	120	95.0 ± 0.9	95.3 ± 0.7
MNIST			
In SW	350	97.1 ± 0.1	99.6 ± 0.1
In SW ($\tau_s = 2\tau_m$)	350	97.2 ± 0.1	99.7 ± 0.1
MNIST 16 × 16			
In SW	246	97.4 ± 0.2	99.2 ± 0.1
On HW	246	96.9 ± 0.1	98.2 ± 0.1

Accuracies are given as mean and standard deviation. Results are distinguished between software simulations (SW) and hardware emulations (HW). For comparison, on the Yin-Yang dataset a linear classifier achieves (64.3 ± 0.2)% test accuracy, while a (non-spiking, not particularly optimized) ANN with 120 hidden neurons achieves (98.7 ± 0.3)%. As a reference for the MNIST dataset we trained a 784-350-10 fully connected ANN, which reached an average test accuracy of (98.2 ± 0.1)%. The results in this table were obtained without extensive hyperparameter tuning.

times of the label neurons. These vary continuously for inputs belonging to other classes, but drop abruptly at the boundary of the area belonging to their own class, which denotes a clear separation—see, for example, the abrupt change from red (late spike time) to yellow (early spike time) of the Yin neuron when moving from Yang to Yin (Fig. 2e, left).

MNIST classification task. To study the scalability of our approach to larger and more high-dimensional datasets, we applied it to the classification of MNIST handwritten digits[66]. Figure 3 shows training results for networks with 784-350-10 neurons (input-hidden-label layer size), where pixel intensities were translated to spike times. During training, noise was added to the input samples to aid generalization, but no bias spikes were used. As seen in Fig. 3a, training converges for 10 different initial random seeds, reaching a final test accuracy of (97.1 ± 0.1)%. Similar results are also achieved for deeper architectures with multiple hidden layers (Supplementary Table B1 provides additional simulation runs with different network architectures).

For reference, we consider several other results obtained with spiking-time coding. In ref. [53], a maximum test accuracy of 97.55% using a network with a hidden layer of 800 neurons is reported. Note that this work uses non-leaky neurons with effectively infinite membrane memory. Also for non-leaky neurons, but using an approximative approach for calculating gradients, Kheradpisheh and Masquelier[54] report 97.4% using 400 hidden neurons. In ref. [68], a maximum test accuracy of 97.96% was achieved using 340 hidden neurons, supported by a regular spike grid and extensive hyperparameter search.

We note that there also exist trial-averaging and spike-count-based approaches that have the benefit of more straightforward learning rules, but these approaches sacrifice precision, neuronal real-estate or time-to-solution in comparison to frameworks based on the precise timing of single output spikes. For example, Esser et al.[61] report 92.7% using 512 neurons, while Tavanaei et al.[69] require 1,000 hidden neurons to achieve 96.6%.

Fast neuromorphic classification. In our framework, the time to solution is a function of the network depth and the time constants τ_m and τ_s. Assuming typical biological timescales, most input patterns in the above scenario are classified within several milliseconds. By leveraging the speedup of neuromorphic systems such as BrainScaleS[46,67], with intrinsic acceleration factors of 10^3 to 10^4, the same computation can be achieved within microseconds. In the following, we present an implementation of our framework on BrainScaleS-2 and discuss its performance in conjunction with the achieved classification speed and energy consumption. For a proof-of-concept implementation on its predecessor BrainScaleS-1, see Supplementary Section A.

The advantages of such a neuromorphic implementation come at the cost of reduced control. Training needs to cope with phenomena such as spike jitter, limited weight range and granularity, as well as neuron parameter variability, among others. In general, an important aspect of any theory aiming for compatibility with physical substrates, be they biological or artificial, is its robustness to substrate imperfections; our results on BrainScaleS-2 implicitly represent a powerful demonstration of this property. To further substantiate the generalizability of our algorithm to different substrates, we complement our experimental results with a simulation study of various substrate-induced distortive effects.

Learning on BrainScaleS-2. BrainScaleS-2 is a mixed-signal accelerated neuromorphic platform with 512 physical neurons, each being able to receive inputs via 256 configurable synapses. These neurons can be coupled to form larger logical neurons with a correspondingly increased number of inputs. At the heart of each neuron is an analogue circuit emulating LIF neuronal dynamics with an acceleration factor of 10^3 to 10^4 compared to biological timescales.

Owing to variations in the manufacturing process, the realized circuits systematically deviate from each other (fixed-pattern noise). Although these variations can be reduced by calibrating each circuit[70], considerable differences remain (standard deviation on the order of 5% on BrainScaleS-2) and pose a challenge for possible neuromorphic algorithms—along with other features of physical model systems such as spike time jitter or spike loss[33,34,63,71].

The chip's synaptic arrays were configured to support arbitrary fully connected networks of up to 256 emulated neurons with a maximum of 256 inputs per neuron. Each such logical connection was realized via two physical synapses to allow transitions between an excitatory and an inhibitory regime. Synaptic weights on the chip are configurable with 6-bit precision. More details about our set-up are available in the Methods.

We used an in-the-loop training approach[23,33,72], where inference runs emulated on the neuromorphic substrate were interleaved with host-based weight update calculations. For emulating the forward pass, the spike times for each sample in a mini-batch were joined sequentially into one long spike train and then injected into the neuromorphic system via a field-programmable gate array (FPGA). The latter was also used to record the spikes emitted by the hidden and label layers.

Figure 4a–d shows the results of training a spiking network with 120 hidden neurons on BrainScaleS-2 on the Yin-Yang dataset. The system quickly learned to discriminate between the presented patterns, with an average test accuracy of (95.0 ± 0.9)%.

The hardware emulation performs similarly to the software simulations (Fig. 2), with the wrong classifications still only happening along the borders of the areas with different labels (Fig. 4c). The remaining difference in performance after training is attributable to the substrate variability (cf. Fig. 4h). Considering that one of the specific challenges built into the Yin-Yang dataset resides in the continuity of its input space and abrupt class switch between bordering areas, this result highlights the robustness of our approach.

To classify the MNIST dataset using the BrainScaleS-2 system, we emulated and trained a network of size 256-246-10 (Fig. 4f–h). Owing to the restrictions imposed by the hardware on the input dimensionality, we used downsampled images of 16×16 pixels. Across multiple initializations, we achieved a test accuracy of (96.9 ± 0.1)%; similarly to the Yin-Yang dataset, this is only slightly

Table 2 | Comparison of pattern recognition models on the MNIST dataset emulated on neuromorphic back-ends, sorted by classification speed

Platform	Type	Technology	Coding	Input resolution	Network size/ structure	Data augmentation/ regularization	Energy per classification	Classifications per second[a]	Test accuracy (%)	Ref. (year)
Nvidia Tesla P100	Digital	14 nm	ANN	28 × 28	CNN[b]	Dropout	852 µJ	125,000	99.2	Supplementary Section SI.E.2
SpiNNaker	Digital	130 nm	Rate	28 × 28	784-600-500-10	Noisy input encoding	3.3 mJ	91	95.0	[82] (2015)
True North	Digital	28 nm	Rate	28 × 28	CNN	Noisy input encoding	0.27 µJ	1,000	92.7	[61] (2015)
True North	Digital	28 nm	Rate	28 × 28	CNN	Noisy input encoding	108 µJ	1,000	99.4	[61] (2015)
Loihi	Digital	14 nm	Bin. rate	(20 × 20)[c]	400-400-10	Not available	2.5 µJ	5,917	96.2	[83] (2021)
Unnamed (Intel)	Digital	10 nm	Temporal	(28 × 28)[d]	236-20	Stochastic spike loss	1.0 µJ	6,250	88.0	[84] (2018)
BrainScaleS-2	Mixed	65 nm	Temporal	16 × 16	256-246-10	Input noise	8.4 µJ	20,800	96.9	This work; Supplementary Section SI.E.1

[a]Note that some platforms achieve a high number of classifications per second simply by processing a large number of samples in parallel, while other platforms rely on the sequential (but fast) processing of individual samples. [b]Standard architecture given as an example in the PyTorch repository, for details see Supplementary Section SI.E.2. [c]Four (empty) pixels on each margin are cropped to yield the 20 × 20 centre from the 28 × 28 image. [d]The 28 × 28 image is preprocessed using 5 × 5 Gabor filters and 3 × 3 pooling before being sent into the chip. For reference, an ANN running on a graphics processing unit is included in the top row. Note that we include only references that present measurements for both energy and throughput in addition to accuracy. An extended table containing results with partial or estimated measurements is provided as Supplementary Table F3.

lower than in software simulations of equally sized networks (Table 1). The ability of our framework to achieve reliable classification despite such substrate-induced distortions is well illustrated by post-training membrane dynamics measured on the chip (Fig. 4h). In all cases shown here, the correct label neuron spikes before 10 µs and is clearly separable from all other label neurons.

Because of its short intrinsic time constants and overall energy efficiency, the BrainScaleS-2 system enables very fast and energy-efficient acquisition of classification results. Classification of the 10,000 MNIST test samples takes a total of 0.937 s, including data transmission, emulation of dynamics and return of the classification results. The total time on the BrainScaleS-2 chip was 480 ms (a detailed breakdown of the execution time is shown in Supplementary Section E). The power consumption of the chip, measured during runtime, including all chip components needed for spike generation and processing (that is, excluding the host and FPGA) amounted to 175 mW. For measurement details and scalability considerations we refer to Supplementary Section E. This results in an average energy consumption of 8.4 µJ per classification. Table 2 provides a comparison to other neuromorphic platforms.

Note that the networks on the other neuromorphic platforms differ in their architectures, coding schemes and training methods, and while we list some of these differences in the table, a direct comparison in terms of individual numbers remains difficult.

This table only includes references in which measurements for both classification rate and energy are reported. A more comprehensive overview, including studies that lack some of the above measurements, is provided in Supplementary Table F3.

Our current experimental set-up leaves room for substantial optimization. For an estimation of possible improvements and their potential effect on classification rate and energy consumption, see Supplementary Section E and ref. [72]. With these improvements we expect to increase the classification rate by up to a factor of four while simultaneously decreasing the energy-per-classification value by up to a factor of three.

Robustness of time-to-first-spike learning. As noted earlier, a learning scheme operating only on spike times combined with our coding represents a natural fit for neuromorphic hardware, both for requiring commonly accessible observables (that is, spike times, as opposed to, for example, membrane potentials or synaptic currents) and due to its intrinsic efficiency, as it emphasizes few and early spikes. An important indicator of a model's feasibility for neuromorphic emulation is its robustness towards substrate-induced distortions. By experimentally demonstrating its capabilities on BrainScaleS-2, we have implicitly provided one substantive data point for our framework. Here, we present a more comprehensive study of the robustness of our approach.

Most physical neuronal substrates have several forms of variability in common (chapter 5 in ref. [73]). In both digital and mixed-signal systems, synaptic weights are typically limited in both range and resolution. Additionally, the parameters of analogue neuron and synapse circuits exhibit a certain spread. To study the impact of these effects, we included them in software simulations of our model applied to the Yin-Yang classification task.

In this context, we highlight the importance of a detail mentioned in the derivation of equation (4). The output spike time given in equation (2) depends only on neuron parameters, presynaptic spike times and weights, so its derivatives share the same dependencies (equations (22) and (23)). With some manipulations, the equation for the actual output spike time can be inserted (equations (24) and (25)), producing a version of the learning rule that directly depends on the output spike time itself. This version thus allows the incorporation of additional information gained in the forward pass and is therefore expected to be substantially more stable, which is confirmed below.

Using dimensionless weight units (scaled by the inverse threshold), we observe that an upper weight limit of ~3 is sufficient for achieving peak performance (Fig. 5a). This weight value is equivalent to a PSP that covers the distance between leak potential and firing threshold.

If this is not achievable within the typical parametrization range of a neuromorphic chip, the effective maximum weight to the hidden layer can be increased by multiplexing each input into the network (Methods).

In the experiments with limited weight resolution (both in software and on hardware), a floating-point-precision ‘shadow’ copy of synaptic weights was kept in memory. The forward and backward pass used discretized weight values, while the calculated weight updates were applied to the shadow weights[74]. Our model shows approximately constant performance for weight

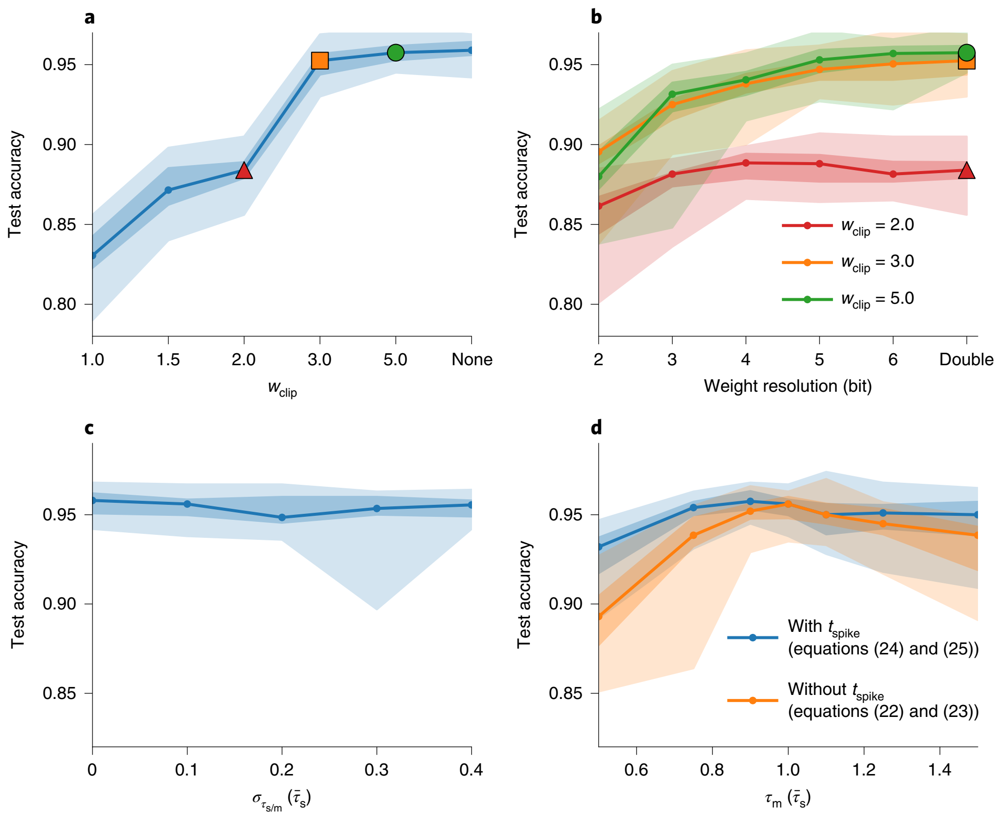

Fig. 5 | Effects of substrate imperfections. Modelled constraints were added artificially into simulated networks. All panels show the median, quartiles, minimum and maximum of the final test accuracy on the Yin-Yang dataset for 20 different initializations. **a**, Limited weight range. The weights were clipped to the range $[-w_{\text{clip}}, w_{\text{clip}}]$ during training and evaluation. The triangle, square and circle mark the clip values that are used in **b**. **b**, Limited weight resolution. For the three weight ranges marked in **a**, the weight resolution was reduced from a double precision float value down to 2 bits. Here, n-bit precision denotes a set-up where the interval $[-w_{\text{clip}}, w_{\text{clip}}]$ is discretized into $2 \times 2^n - 1$ samples (n weight bits plus sign). **c**, Time constants with fixed-pattern noise. For these simulations, each neuron received a random τ_s and τ_m independently drawn from the normal distribution $N(\bar{\tau}_s, \sigma_{\tau_{s/m}})$. This means that the ratio of time constants was essentially never the one assumed by the learning rule. **d**, Systematic shift between time constants. Here τ_s was drawn from $N(\bar{\tau}_s, \sigma_{\tau_{s/m}})$ while τ_m was drawn from $N(\bar{\tau}_m, \sigma_{\tau_{s/m}})$ for each neuron for varying mean $\bar{\tau}_m$ and fixed $\sigma_{\tau_{s/m}} = 0.1\bar{\tau}_s$. The orange curve illustrates a training where the backward pass performs 'naïve' gradient descent, without using explicit information about output spike times. The blue curve, as all other panels, has the output spike time as an observable.

resolutions down to 5 bit, followed by gradual degradation below (Fig. 5b).

Interestingly, adding variability to the synapse and membrane time constants has no discernible effects (Fig. 5c). This is a direct consequence of having used the true output spike times for the learning rule in the backward pass. A comparison to 'naïve' gradient descent without this information is shown in Fig. 5d. These simulations show that the algorithm can be expected to adequately cope with a large amount of fixed-pattern noise on the time constants if the mean of the distributions for τ_m and τ_s match reasonably well with the values assumed by the learning rule (up to 10–20% difference).

Additionally, in Supplementary Section C we investigate trained networks regarding their robustness to adverse effects that appear only after training, such as temperature-induced parameter variations or inactivation of neurons. Our simulations show that trained networks can cope with such effects, suggesting that our training algorithm develops network structures robust even to distortions not present during training.

Finally, we note that all of the effects addressed above also have biological correlates. Although not directly reflecting the variability of biological neurons and synapses, our simulations do suggest that biological variability does not present a fundamental obstacle to our form of TTFS computation.

Discussion

We have proposed a model of first-spike-time learning that builds on a rigorous analysis of neuro-synaptic dynamics with finite time constants and provides exact learning rules for optimizing first-spike times. The resulting form of synaptic plasticity operates on pre- and postsynaptic spike times and effectively solves the credit assignment problem in spiking networks. For the specific case of hierarchical feed-forward topologies, it yields a spike-based form of error backpropagation. In this Article, we have applied this algorithm to networks with one and two hidden layers. Given the reported results, we are confident that our approach scales to even larger and deeper networks.

Although TTFS coding is an exceptionally appealing paradigm for reasons of speed and efficiency, our approach is not restricted to this particular coding scheme. Our learning rules enable a rigorous manipulation of spike times and can be used for a variety of loss functions that target other relationships between spike timings. The time-to-first-spike scenario studied here merely represents the simplest, yet arguably also the fastest and most efficient paradigm for spike-based classification of static patterns. Additionally, our derived theory is applicable to more complex, for example, recurrent, network structures and multi-spike coding schemes, which are needed for processing temporal data streams.

First-spike coding schemes are particularly relevant in the context of biology, where decisions often have to be taken under pressure of time. The action to be taken in response to a stimulus can be considerably sped up by encoding it in first-spike times. In turn, such fast decision making on the order of ~100 ms (refs. [42,43]) will have a particularly sensitive dependence on exact spike times and thus require a corresponding precision of parameters.

At first glance, demands for precision appear at odds with the imperfect, variable nature of microscopic physical substrates, both biological and artificial. We met this challenge by incorporating output spike times directly into the backward pass. With this, the theoretical requirement of exact ratios of membrane to synaptic time constants is substantially softened, which greatly extends the applicability of our framework to a wide range of substrates, including, in particular, BrainScaleS-2.

By requiring only spike times, the proposed learning framework has minimal demands for neuromorphic hardware and becomes inherently robust towards substrate-induced distortions. This further enhances its suitability for a wide range of neuromorphic platforms.

Bolstered by the design characteristics of the BrainScaleS-2 system, our implementation achieves a time to classification of ~10 µs after receiving the first spike. Including relaxation between patterns and communication, the complete MNIST test set with 10,000 samples is classified in less than 1 s with an energy consumption of ~8.4 µJ per classification, which compares favourably with other neuromorphic solutions for pattern classification. The time characteristics of this implementation do not deteriorate for increased layer sizes because neurons communicate asynchronously and their dynamics are emulated independently. For the current incarnation of BrainScaleS-2, an increase in spiking activity only has a negligible effect on power consumption. Furthermore, for larger numbers of neurons we would expect only a weak increase of the power drain.

We also stress that, in contrast to, for example, graphics processing units, our system was used to process input data sequentially. Our reported classification speed is thus a direct consequence of our coding scheme combined with the system's accelerated dynamics. Further increasing the throughput by parallelization (simultaneously using multiple chips) is straightforward and would not affect the required energy per classification.

Due to the complexity of our exact gradient-based rules, our hardware networks were trained using updates calculated off-chip based on emulated spike times. Early, promising simulations using a substantially simplified learning rule, however, suggest the possibility of an on-chip implementation of our framework. Furthermore, we note that our learning rules require three components that can all be made available at the locus of the synapse: pre- and postsynaptic spikes, as in classical spike-timing-dependent plasticity, and an error term, which could be propagated by mechanisms such as those proposed in, for example, refs. [75,76]. This raises the intriguing possibility for our framework to help explain learning in biological substrates as well.

Because, compared to the von Neumann paradigm, artificial brain-inspired computing is only in its infancy, its range of possible applications still remains an open question. This is reflected by most state-of-the-art neuromorphic approaches to information processing, which, to accommodate a wide range of spike-based computational paradigms, aim for a large degree of flexibility in network topology and parametrization. Despite the obvious efficiency trade-off of such general-purpose platforms, we have shown that an embedded version of our framework can achieve a powerful combination of performance, speed, efficiency and robustness. This gives us confidence that a more specialized neuromorphic implementation of our model represents a competitive alternative to current solutions based on von Neumann architectures, especially in edge computing scenarios.

Methods

Preliminaries. In this section we derive the equations from the main Article, starting with the learning rule for $\tau_m \to \infty$, then $\tau_m = \tau_s$, equation (2) and finally $\tau_m = 2\tau_s$, equation (3). The case $\tau_m \to \infty$ has already been discussed in ref. [53] and was reproduced here for completeness and comparison. Owing to the symmetry in τ_m and τ_s of the PSP (equation (14)), the $\tau_m = 2\tau_s$ case describes the $\tau_m = \frac{1}{2}\tau_s$ case as well.

For each, a solution for the spike time T, defined by

$$u(T) = \vartheta, \tag{8}$$

has to be found, given LIF dynamics

$$u(t) = \frac{1}{C_m}\frac{\tau_m\tau_s}{\tau_m - \tau_s}\sum_{\text{spikes } t_i} w_i\kappa(t - t_i), \tag{9}$$

$$\kappa(t) = \theta(t)\left[\exp\left(-\frac{t}{\tau_m}\right) - \exp\left(-\frac{t}{\tau_s}\right)\right], \tag{10}$$

with membrane time constant $\tau_m = C_m/g_\ell$ and the PSP kernel κ given by a difference of exponentials. Here we already assumed our TTFS use case in which each neuron only produces one relevant spike and the second sum in equation (1) reduces to a single term.

For convenience, we use the following definitions:

$$a_n := \sum_{i\in C} w_i \exp\left(\frac{t_i}{n\tau_s}\right), \tag{11}$$

$$b := \sum_{i\in C} w_i \frac{t_i}{\tau_s}\exp\left(\frac{t_i}{\tau_s}\right), \tag{12}$$

with summation over the set of causal presynaptic spikes $C = \{i | t_i < T\}$.

In practice, this definition of the causal set C is not a closed-form expression because the output spike time T depends explicitly on C. However, it can be computed straightforwardly by iterating over the ordered sets of input spike times (for n presynaptic spikes there are n sets $\tilde{C}_i$ each comprising the i first input spikes). For each set $\tilde{C}_i$ one calculates an output spike time T_i and determines if this happens later than the last input of this set and before the next input (the $i+1$th input spike). The earliest such spike T_i is the actual output spike time and the corresponding $\tilde{C}_i$ is the correct causal set. If no such causal set $\tilde{C}_i$ exists, the neuron did not spike and we assign it the spike time $T = \infty$.

The nLIF learning rule for $\tau_m \to \infty$. With this choice of τ_m, the first term in equation (10) becomes 1 and we recover the nLIF case discussed in ref. [53]. Given the existence of an output spike, in equation (8) the spike time T appears only in one place and simple reordering yields

$$\frac{T}{\tau_s} = \ln\left[\frac{a_1}{a_\infty - \vartheta C_m/\tau_s}\right], \tag{13}$$

where we used equation (11) for $n = 1$ and $n = \infty$, the latter being the sum over the weights.

The learning rule for $\tau_m = \tau_s$. According to l'Hôpital's rule, in the limit $\tau_m \to \tau_s$, equation (9) becomes a sum over α-functions of the form

$$u(t) = \frac{1}{C_m}\sum_i w_i\theta(t - t_i)(t - t_i)\exp\left(-\frac{t - t_i}{\tau_s}\right). \tag{14}$$

Using these voltage dynamics for the equation of the spike time equation (8), together with the definitions of equations (11) and (12) and $\tau_m = C_m/g_\ell$, we get the equation

$$0 = g_\ell\vartheta\exp\left(\frac{T}{\tau_s}\right) + \underbrace{b - a_1\frac{T}{\tau_s}}_{=:\,y}. \tag{15}$$

The variable y is introduced to bring the equation into the form

$$h\exp(h) = z \tag{16}$$

which can be solved with the differentiable Lambert W function $h = \mathcal{W}(z)$. The goal is now to bring equation (15) into this form, and this is achieved by reformulation in terms of y:

$$0 = g_\ell\vartheta\exp\left(\frac{b}{a_1}\right)\exp\left(-\frac{y}{a_1}\right) + y \tag{17}$$

$$\underbrace{\frac{y}{a_1}}_{=:\,h}\exp\left(\frac{y}{a_1}\right) = \underbrace{-\frac{g_\ell\vartheta}{a_1\exp\left(\frac{b}{a_1}\right)}}_{=:\,z}. \tag{18}$$

With the definition of the Lambert W function the spike time can be written as

$$\frac{T}{\tau_s} = \frac{b}{a_1} - \mathcal{W}\left[-\frac{g_\ell \vartheta}{a_1}\exp\left(\frac{b}{a_1}\right)\right]. \quad (19)$$

Branch choice. Given that a spike happens, there will be two threshold crossings: one from below at the actual spike time and one from above when the voltage decays back to the leak potential (Supplementary Fig. F1a,b). Correspondingly, the Lambert W function (Supplementary Fig. F1c,d) has two real branches (in addition to infinite imaginary ones), and we need to choose the branch that returns the earlier solution. In case the voltage is only tangent to the threshold at its maximum, the Lambert W function only has one solution.

For choosing the branch in the other cases we need to look at h from the definition, that is

$$h = \frac{y}{a_1} = \frac{b}{a_1} - \frac{T}{\tau_s}. \quad (20)$$

In a setting with only one strong enough input spike, the summations in a_n and b reduce to yield $h = (t_i - T)/\tau_s$. Because the maximum of the PSP for $\tau_m = \tau_s$ occurs at $t_i + \tau_s$, we know that the spike must occur at $T \leq t_i + \tau_s$ and therefore

$$-1 \leq \frac{t_i - T}{\tau_s} = h. \quad (21)$$

This corresponds to the branch cut of the Lambert W function, meaning we must choose the branch with $h \geq -1$. For a general setting, if we know a spike exists, we expect a_n and b to be positive. To get the earlier threshold crossing, we need the branch that returns the larger $\mathcal{W}$ (Supplementary Fig. F1d), that is, where $\mathcal{W} = h > -1$.

Derivatives. The derivatives for t_i in the causal set $i \in C$ come down to

$$\frac{\partial T}{\partial w_i}(\mathbf{w}, \mathbf{t}) = \frac{\tau_s}{a_1}\exp\left(\frac{t_i}{\tau_s}\right)\left[z\mathcal{W}'(z) + \left(\frac{t_i}{\tau_s} - \frac{b}{a_1}\right)(1 - z\mathcal{W}'(z))\right], \quad (22)$$

$$\frac{\partial T}{\partial t_i}(\mathbf{w}, \mathbf{t}) = \frac{w_i}{a_1}\exp\left(\frac{t_i}{\tau_s}\right)\left[1 + \left(\frac{t_i}{\tau_s} - \frac{b}{a_1}\right)(1 - z\mathcal{W}'(z))\right]. \quad (23)$$

A crucial step is to reinsert the definition of the spike time where possible (cf. Fig. 5d). For this we need the derivative of the Lambert W function $z\mathcal{W}'(z) = \frac{\mathcal{W}(z)}{\mathcal{W}(z)+1}$ that follows from differentiating its definition equation (16) with $h = \mathcal{W}(z)$ with respect to z. With this equation one can calculate the derivative of equation (19) with respect to incoming weights and times as functions of presynaptic weights, input spike times and output spike time:

$$\frac{\partial T}{\partial w_i}(\mathbf{w}, \mathbf{t}, T) = -\frac{1}{a_1}\frac{1}{\mathcal{W}(z)+1}\exp\left(\frac{t_i}{\tau_s}\right)(T - t_i), \quad (24)$$

$$\frac{\partial T}{\partial t_i}(\mathbf{w}, \mathbf{t}, T) = -\frac{1}{a_1}\frac{1}{\mathcal{W}(z)+1}\exp\left(\frac{t_i}{\tau_s}\right)\frac{w_i}{\tau_s}(T - t_i - \tau_s). \quad (25)$$

These equations are equivalent to equations (4) and (5) shown in the main text.

The learning rule for $\tau_m = 2\tau_s$. Inserting the voltage (equation (9)) into the spike time (equation (8)) yields

$$\begin{aligned} g_\ell \vartheta = {} & \exp\left(-\tfrac{T}{\tau_m}\right)\sum_{i\in C} w_i \exp\left(\tfrac{t_i}{\tau_m}\right) \\ & - \exp\left(-\tfrac{T}{\tau_s}\right)\sum_{i\in C} w_i \exp\left(\tfrac{t_i}{\tau_s}\right). \end{aligned} \quad (26)$$

Reordering and rewriting this in terms of a_1, a_2 and τ_s (with $\tau_m = 2\tau_s$) we get

$$0 = -a_1\left[\exp\left(-\frac{T}{2\tau_s}\right)\right]^2 + a_2\exp\left(-\frac{T}{2\tau_s}\right) - g_\ell\vartheta. \quad (27)$$

This is written such that its quadratic nature becomes apparent, making it possible to solve for $\exp(-T/2\tau_s)$ and thus

$$\frac{T}{\tau_s} = 2\ln\left[\frac{2a_1}{a_2 + \sqrt{a_2^2 - 4a_1 g_\ell \vartheta}}\right]. \quad (28)$$

Branch choice. The quadratic equation has two solutions that correspond to the voltage crossing at spike time and relaxation towards the leak later; again, we want the earlier of the two solutions. It follows from the monotonicity of the logarithm that the earlier time is the one with the larger denominator. Due to an output spike requiring an excess of recent positively weighted input spikes, a_n are positive and the + solution is the correct one.

Derivatives. Using the definition $x = \sqrt{a_2^2 - 4a_1 g_\ell\vartheta}$ for brevity, the derivatives of equation (28) are

$$\frac{\partial T}{\partial w_i}(\mathbf{w}, \mathbf{t}) = 2\tau_s\left[\frac{1}{a_1} + \frac{2g_\ell\vartheta}{(a_2 + x)x}\right]\exp\left(\frac{t_i}{\tau_s}\right) - \frac{2\tau_s}{x}\exp\left(\frac{t_i}{2\tau_s}\right), \quad (29)$$

$$\frac{\partial T}{\partial t_i}(\mathbf{w}, \mathbf{t}) = 2w_i\left[\frac{1}{a_1} + \frac{2g_\ell\vartheta}{(a_2 + x)x}\right]\exp\left(\frac{t_i}{\tau_s}\right) - \frac{w_i}{x}\exp\left(\frac{t_i}{2\tau_s}\right). \quad (30)$$

Again, inserting the output spike time yields

$$\frac{\partial T}{\partial w_i}(\mathbf{w}, \mathbf{t}, T) = \frac{2\tau_s}{a_1}\left[1 + \frac{g_\ell\vartheta}{x}\exp\left(\frac{T}{2\tau_s}\right)\right]\exp\left(\frac{t_i}{\tau_s}\right) - \frac{2\tau_s}{x}\exp\left(\frac{t_i}{2\tau_s}\right), \quad (31)$$

$$\frac{\partial T}{\partial t_i}(\mathbf{w}, \mathbf{t}, T) = \frac{2w_i}{a_1}\left[1 + \frac{g_\ell\vartheta}{x}\exp\left(\frac{T}{2\tau_s}\right)\right]\exp\left(\frac{t_i}{\tau_s}\right) - \frac{w_i}{x}\exp\left(\frac{t_i}{2\tau_s}\right). \quad (32)$$

Error backpropagation in a layered network. Our goal is to update the network's weights such that they minimize the loss function $L[\mathbf{t}^{(N)}, n^*]$. For weights projecting into the label layer, updates are calculated via

$$\Delta w_{ni}^{(N)} \propto -\frac{\partial L[\mathbf{t}^{(N)}, n^*]}{\partial w_{ni}^{(N)}} = -\frac{\partial t_n^{(N)}}{\partial w_{ni}^{(N)}}\frac{\partial L[\mathbf{t}^{(N)}, n^*]}{\partial t_n^{(N)}}. \quad (33)$$

The weight updates of deeper layers can be calculated iteratively by application of the chain rule:

$$\Delta w_{ki}^{(l)} \propto -\frac{\partial L[\mathbf{t}^{(N)}, n^*]}{\partial w_{ki}^{(l)}} = -\frac{\partial t_k^{(l)}}{\partial w_{ki}^{(l)}}\delta_k^{(l)}, \quad (34)$$

where the second term is a propagated error that can be calculated recursively with a sum over the neurons in layer $(l+1)$:

$$\delta_k^{(l)} := \frac{\partial L[\mathbf{t}^{(N)}, n^*]}{\partial t_k^{(l)}} = \sum_j \frac{\partial t_j^{(l+1)}}{\partial t_k^{(l)}}\delta_j^{(l+1)}. \quad (35)$$

In the following we treat the $\tau_m = \tau_s$ case, but the calculations can be performed analogously for the other cases. Rewriting equations (24) and (25) in a layer-wise setting, the derivatives of the spike time for a neuron k in arbitrary layer l are

$$\frac{\partial t_k^{(l)}}{\partial w_{ki}^{(l)}}(\mathbf{w}, \mathbf{t}^{(l-1)}, \mathbf{t}^{(l)}) = -\frac{1}{a_1}\exp\left(\frac{t_i^{(l-1)}}{\tau_s}\right)\frac{1}{\mathcal{W}(z)+1}\left(t_k^{(l)} - t_i^{(l-1)}\right), \quad (36)$$

$$\frac{\partial t_k^{(l)}}{\partial t_i^{(l-1)}}(\mathbf{w}, \mathbf{t}^{(l-1)}, \mathbf{t}^{(l)}) = -\frac{1}{a_1}\exp\left(\frac{t_i^{(l-1)}}{\tau_s}\right)\frac{1}{\mathcal{W}(z)+1}\frac{w_{ki}^{(l)}}{\tau_s}\left(t_k^{(l)} - t_i^{(l-1)} - \tau_s\right). \quad (37)$$

Inserting equations (35) to (37) into equations (33) and (34) yields a synaptic learning rule that implements exact error backpropagation on spike times.

This learning rule can be rewritten to resemble the standard error backpropagation algorithm for ANNs:

$$\boldsymbol{\delta}^{(N)} = \frac{\partial L}{\partial \mathbf{t}^{(N)}}, \quad (38)$$

$$\boldsymbol{\delta}^{(l-1)} = \left(\widehat{\mathbf{B}}^{(l)} - 1\right) \odot \boldsymbol{\rho}^{(l-1)} \odot \left(\mathbf{w}^{(l),\mathrm{T}}\boldsymbol{\delta}^{(l)}\right), \quad (39)$$

$$\Delta\mathbf{w}^{(l)} = -\eta\tau_s\left(\boldsymbol{\delta}^{(l)}\boldsymbol{\rho}^{(l-1),\mathrm{T}}\right) \odot \widehat{\mathbf{B}}^{(l)}, \quad (40)$$

where $\odot$ is the element-wise product, the T superscript denotes the transpose of a matrix and $\boldsymbol{\delta}^{(l-1)}$ is a vector containing the backpropagated errors of layer $(l-1)$. The individual elements of the tensors above are given by

$$\rho_i^{(l)} = -\frac{1}{a_1}\exp\left(\frac{t_i^{(l)}}{\tau_s}\right)\frac{1}{\mathcal{W}(z)+1}, \quad (41)$$

$$\widehat{B}_{ki}^{(l)} = \frac{t_k^{(l)} - t_i^{(l-1)}}{\tau_s}. \quad (42)$$

BrainScaleS-2. The application-specific integrated circuit (ASIC) is built around an analogue neuromorphic core that emulates the dynamics of neurons and synapses. All state variables, such as membrane potentials and synaptic currents, are physically represented in their respective circuits and evolve continuously in time. Considering the natural time constants of such integrated analogue circuits, this emulation takes place at 1,000-fold accelerated timescales compared to the biological nervous system. One BrainScaleS-2 chip features 512 adaptive exponential leaky integrate-and-fire (AdEx) neurons, which can be freely configured; these circuits can be restricted to LIF dynamics as required by our training framework[77]. Both the membrane and synaptic time constants were calibrated to 6 µs.

Each neuron circuit is connected to one of four synapse matrices on the chip, and integrates stimuli from its column of 256 CuBa synapses[59]. Each synapse holds a 6-bit weight value. Its sign is shared with all other synapses located on the same synaptic row. The presented training scheme, however, allows weights to continuously transition between excitation and inhibition. We therefore allocated pairs of synapse rows to convey the activity of single presynaptic partners, one configured for excitation and the other one for inhibition.

Synapses receive their inputs from an event routing module allowing to connect neurons within a chip as well as to inject stimuli from external sources. Events emitted by the neuron circuits are annotated with a time stamp and then sent off-chip. The neuromorphic ASIC is accompanied by an FPGA to handle communication with the host computer. It also provides mechanisms for low-latency experiment control, including the timed release of spike trains into the neuromorphic core. The FPGA is also used to record events and digitized membrane traces originating from the ASIC. BrainScaleS-2 only permits recording one membrane trace at a time. Each membrane voltage shown in Fig. 4h therefore originates from a different repetition of the experiment.

The ASIC is controlled by a layered software stack[78] that exposes the necessary interfaces to a high-level user via Python bindings. These were used in our framework, which is described in the following.

Simulation software. Our experiments were performed using custom modules for the deep learning library PyTorch[79]. The network module implements layers of LIF neurons whose spike times are calculated according to equation (2). This method of determining the spike times of the neurons is fastest, but also memory-intensive. An alternative implementation integrates the dynamical equations of the LIF neurons in a layer, which also yields the neuron spike times. Even though both approaches are technically equivalent, this method is slower and should only be employed if the computing resources are limited.

The activations passed between the layers during the forward pass are the spike times. The equations describing the weight updates for the network (equation (40)) are realized in a custom backward-pass module for the network.

Training and regularization methods. To train a given dataset using our learning framework, the input data have to be translated into spike times first. We do this by defining the times of the earliest and latest possible input spike t_{early} and t_{late} and mapping the range of input values linearly to the time interval $[t_{\mathrm{early}}, t_{\mathrm{late}}]$.

If the dataset requires a bias to be solvable, our framework allows its addition. These bias spikes essentially represent additional input spikes for a layer, which have the same spike time for any input. The weights from the neurons to these 'bias sources' are learned in the same way as all the other synaptic weights. For the Yin-Yang dataset, the addition of a bias spike facilitated training. For some samples, due to the low number of inputs, the relatively low activity that is received by the network is spread out over a long time interval. The additional spike in the middle of the available interval decreases the maximum distance between input spikes for the hidden layer. In contrast, the MNIST dataset has a much higher input dimensionality and the spikes are more distributed over the input time interval. Therefore, the activity provided to the hidden layer at any point in time is high, even without additional bias.

Implementing our learning algorithm as custom PyTorch modules allows us to use the training architecture provided by the library. The simulations were performed using mini-batch training in combination with with the Adam optimizer[80] and learning rate scheduling (the parameters are provided in Supplementary Tables F1 and F2).

To assist learning we employ several regularization techniques. The term $+\alpha\left[\exp\left(t_{n^*}^{(N)}/\beta\tau_{\mathrm{s}}\right)-1\right]$ with scaling parameters $\alpha, \beta \in \mathbb{R}^+$, can be added to the loss in equation (6). This regularizer further pushes the correct neuron towards earlier spike times.

Gaussian noise on the input spike times can be used to combat overfitting. This proved beneficial for the training of the MNIST dataset.

Weight updates Δw with absolute value larger than a given hyperparameter are set to zero to compensate divergence for vanishing denominator in equation (40).

As noted previously, the weight update equations are only defined for neurons that elicit a spike. To prevent fully quiescent networks we add a hyperparameter that controls how many neurons without an output spike are allowed. If the portion of non-spiking neurons is above this threshold, we increase the input weights of the silent neurons. In the case of multiple layers where this applies, only the first such layer with insufficient spikes is boosted. If neurons in a layer are too inactive multiple times in direct succession, the boost to the weights increases exponentially.

Training on hardware. In principle, our training framework can be used to train any neuromorphic hardware platform that (1) can receive a set of input spikes and yield the output spike times of all neurons in the emulated network and (2) can update the weight configuration on the hardware according to the calculated weight updates. In our framework, the hardware replaces the computed forward pass through the network. For the calculation of the loss and the following backward pass, the hardware output spikes are treated as if they had been produced by a forward pass in simulation. The backward pass is identical to pure simulation.

As accessible value ranges of neuron parameters are typically determined by the hardware platform in use, a translation factor between the neuron parameters and weights in software and the parameters realized on hardware needs to be determined. In our experiments with BrainScaleS-2, the translation between hardware and software parameter domain was determined by matching of PSP shapes and spike times predicted by a software forward pass to the ones produced by the chip.

The implicit assumption of having only the first spike emitted by every neuron be relevant for downstream processing can effectively be ensured by using a long enough refractory period. Because the only information-carrying signal that is not reset upon firing is the synaptic current, which is forgotten on the timescale of τ_{s}, we found that, in practice, setting the refractory time $\tau_{\mathrm{ref}} > \tau_{\mathrm{s}}$ leads to most neurons eliciting only one spike before the classification of a given input pattern.

For training the Yin-Yang dataset on BrainScaleS-2, having only five inputs proved insufficient due to the combination of limited weights and neuron variability. We therefore multiplexed each logical input into five physical spike sources, totalling 25 inputs spikes per pattern. Adding further copies of the inputs effectively increased the weights for each individual input. This method has the added benefit of averaging out some of the effects of the fixed-pattern noise on the input circuits as multiple of them are employed for the same task.

Data availability

We used the MNIST[66] and the Yin-Yang dataset[65]. For the latter, see https://github.com/lkriener/yin_yang_data_set.

Code availability

Code for the simulations[81] is available at https://github.com/JulianGoeltz/fastAndDeep.

Received: 17 November 2020; Accepted: 30 July 2021;
Published online: 17 September 2021

References

1. Krizhevsky, A., Sutskever, I. & Hinton, G. E. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems* 1097–1105 (NIPS, 2012).
2. Silver, D. et al. Mastering the game of Go without human knowledge. *Nature* **550**, 354–359 (2017).
3. Brown, T. B. et al. Language models are few-shot learners. Preprint at https://arxiv.org/pdf/2005.14165.pdf (2020).
4. Brooks, R., Hassabis, D., Bray, D. & Shashua, A. Is the brain a good model for machine intelligence?. *Nature* **482**, 462–463 (2012).
5. Ng, A. What artificial intelligence can and can't do right now. *Harvard Business Review* (9 November 2016).
6. Hassabis, D., Kumaran, D., Summerfield, C. & Botvinick, M. Neuroscience-inspired artificial intelligence. *Neuron* **95**, 245–258 (2017).
7. Sejnowski, T. J. *The Deep Learning Revolution* (MIT Press, 2018).
8. Richards, B. A. et al. A deep learning framework for neuroscience. *Nat. Neurosci.* **22**, 1761–1770 (2019).
9. Pfeiffer, M. & Pfeil, T. Deep learning with spiking neurons: opportunities and challenges. *Front. Neurosci* **12**, 774 (2018).
10. Gerstner, W. What is different with spiking neurons? In *Plausible Neural Networks for Biological Modelling. Mathematical Modelling: Theory and Applications* Vol 13. (eds Mastebroek, H. A. K. & Vos, J. E.) 23–48 (Springer, 2001).
11. Izhikevich, E. M. Which model to use for cortical spiking neurons? *IEEE Trans. Neural Netw.* **15**, 1063–1070 (2004).
12. Gerstner, W. *Spiking Neurons* (MIT Press, 1998).
13. Maass, W. Searching for principles of brain computation. *Curr. Opin. Behav. Sci.* **11**, 81–92 (2016).
14. Davies, M. Benchmarks for progress in neuromorphic computing. *Nat. Mach. Intell.* **1**, 386–388 (2019).
15. Linnainmaa, S. *The Representation of the Cumulative Rounding Error of an Algorithm as a Taylor Expansion of the Local Rounding Errors.* Master's thesis (in Finnish), Univ. Helsinki 6–7 (1970).
16. Werbos, P. J. Applications of advances in nonlinear sensitivity analysis. In *System Modeling and Optimization. Lecture Notes in Control and Information Sciences* Vol. 38 (eds Drenick, R. F. & Kozin, F.) 762–770 (Springer, 1982).

17. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning representations by back-propagating errors. *Nature* **323**, 533–536 (1986).
18. Tavanaei, A., Ghodrati, M., Kheradpisheh, S. R., Masquelier, T. & Maida, A. Deep learning in spiking neural networks. *Neural Netw* **111**, 47–63 (2018).
19. Neftci, E. O., Mostafa, H. & Zenke, F. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Process.* **36**, 51–63 (2019).
20. Gütig, R. & Sompolinsky, H. The tempotron: a neuron that learns spike timing-based decisions. *Nat. Neurosci.* **9**, 420–428 (2006).
21. Cao, Y., Chen, Y. & Khosla, D. Spiking deep convolutional neural networks for energy-efficient object recognition. *Int. J. Comput. Vis.* **113**, 54–66 (2015).
22. Diehl, P. U., Zarrella, G., Cassidy, A., Pedroni, B. U. & Neftci, E. Conversion of artificial recurrent neural networks to spiking neural networks for low-power neuromorphic hardware. In *Proc. 2016 IEEE International Conference on Rebooting Computing* (*ICRC*) 1–8 (IEEE, 2016).
23. Schmitt, S. et al. Neuromorphic hardware in the loop: training a deep spiking network on the BrainScaleS wafer-scale system. In *Proc. 2017 International Joint Conference on Neural Networks* (*IJCNN*) 2227–2234 (2017).
24. Wu, J., Chua, Y., Zhang, M., Yang, Q., Li, G., & Li, H. Deep spiking neural network with spike count based learning rule. In *International Joint Conference on Neural Networks* 1–6 (IEEE, 2019).
25. Thakur, C. S. T. et al. Large-scale neuromorphic spiking array processors: a quest to mimic the brain. *Front. Neurosci.* **12**, 891 (2018).
26. Mead, C. Neuromorphic electronic systems. *Proc. IEEE* **78**, 1629–1636 (1990).
27. Roy, K., Jaiswal, A. & Panda, P. Towards spike-based machine intelligence with neuromorphic computing. *Nature* **575**, 607–617 (2019).
28. Petrovici, M. A., Bill, J., Bytschok, I., Schemmel, J. & Meier, K. Stochastic inference with deterministic spiking neurons. Preprint at https://arxiv.org/pdf/1311.3211.pdf (2013).
29. Neftci, E., Das, S., Pedroni, B., Kreutz-Delgado, K. & Cauwenberghs, G. Event-driven contrastive divergence for spiking neuromorphic systems. *Front. Neurosci.* **7**, 272 (2014).
30. Petrovici, M. A., Bill, J., Bytschok, I., Schemmel, J. & Meier, K. Stochastic inference with spiking neurons in the high-conductance state. *Phys. Rev. E* **94**, 042312 (2016).
31. Neftci, E. O., Pedroni, B. U., Joshi, S., Al-Shedivat, M. & Cauwenberghs, G. Stochastic synapses enable efficient brain-inspired learning machines. *Front. Neurosci.* **10**, 241 (2016).
32. Leng, L. et al. Spiking neurons with short-term synaptic plasticity form superior generative networks. *Sci. Rep.* **8**, 10651 (2018).
33. Kungl, A. F. et al. Accelerated physical emulation of Bayesian inference in spiking neural networks. *Front. Neurosci.* **13**, 1201 (2019).
34. Dold, D. et al. Stochasticity from function-why the Bayesian brain may need no noise. *Neural Netw.* **119**, 200–213 (2019).
35. Jordan, J. et al. Deterministic networks for probabilistic computing. *Sci. Rep.* **9**, 18303 (2019).
36. Hunsberger, E. & Eliasmith, C. Training spiking deep networks for neuromorphic hardware. Preprint at https://arxiv.org/pdf/1611.05141.pdf (2016).
37. Kheradpisheh, S. R., Ganjtabesh, M., Thorpe, S. J. & Masquelier, T. STDP-based spiking deep convolutional neural networks for object recognition. *Neural Netw.* **99**, 56–67 (2018).
38. Illing, B., Gerstner, W. & Brea, J. Biologically plausible deep learning-but how far can we go with shallow networks?. *Neural Netw* **118**, 90–101 (2019).
39. Bohte, S. M., Kok, J. N. & La Poutré, J. A. Spikeprop: backpropagation for networks of spiking neurons. In *8th European Symposium on Artificial Neural Networks* 419–424 (2000).
40. Zenke, F. & Ganguli, S. Superspike: supervised learning in multilayer spiking neural networks. *Neural Comput.* **30**, 1514–1541 (2018).
41. Huh, D. & Sejnowski, T. J. Gradient descent for spiking neural networks. In *Advances in Neural Information Processing Systems* Vol. 31, 1433–1443 (NIPS, 2018).
42. Thorpe, S., Delorme, A. & Van Rullen, R. Spike-based strategies for rapid processing. *Neural Netw.* **14**, 715–725 (2001).
43. Thorpe, S., Fize, D. & Marlot, C. Speed of processing in the human visual system. *Nature* **381**, 520–522 (1996).
44. Johansson, R. S. & Birznieks, I. First spikes in ensembles of human tactile afferents code complex spatial fingertip events. *Nat. Neurosci.* **7**, 170–177 (2004).
45. Gollisch, T. & Meister, M. Rapid neural coding in the retina with relative spike latencies. *Science* **319**, 1108–1111 (2008).
46. Schemmel, J. et al. A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *Proc. 2010 IEEE International Symposium on Circuits and Systems* 1947–1950 (IEEE, 2010).
47. Akopyan, F. et al. TrueNorth: design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip. *IEEE Trans. Comput. Aided Design Integrated Circuits Syst.* **34**, 1537–1557 (2015).
48. Billaudelle, S. et al. Versatile emulation of spiking neural networks on an accelerated neuromorphic substrate. In *IEEE International Symposium on Circuits and Systems* 1–5 (IEEE, 2020).
49. Davies, M. et al. Loihi: a neuromorphic manycore processor with on-chip learning. *IEEE Micro* **38**, 82–99 (2018).
50. Mayr, C., Höppner, S., & Furber, S. SpiNNaker 2: a 10 million core processor system for brain simulation and machine learning-keynote presentation. In *Communicating Process Architectures 2017 & 2018* 277–280 (IOS Press, 2019).
51. Pei, J. et al. Towards artificial general intelligence with hybrid Tianjic chip architecture. *Nature* **572**, 106–111 (2019).
52. Moradi, S., Qiao, N., Stefanini, F. & Indiveri, G. A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (dynaps). *IEEE Trans. Biomed. Circuits Syst.* **12**, 106–122 (2017).
53. Mostafa, H. Supervised learning based on temporal coding in spiking neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **29**, 3227–3235 (2017).
54. Kheradpisheh, S. R. & Masquelier, T. S4NN: temporal backpropagation for spiking neural networks with one spike per neuron. *Int. J. Neural Syst.* **30**, 2050027 (2020).
55. Rauch, A., La Camera, G., Luscher, H.-R., Senn, W. & Fusi, S. Neocortical pyramidal cells respond as integrate-and-fire neurons to in vivo-like input currents. *J. Neurophysiol.* **90**, 1598–1612 (2003).
56. Gerstner, W. & Naud, R. How good are neuron models? *Science* **326**, 379–380 (2009).
57. Teeter, C. et al. Generalized leaky integrate-and-fire models classify multiple neuron types. *Nat. Commun.* **9**, 709 (2018).
58. Göltz, J. *Training Deep Networks with Time-to-First-Spike Coding on the BrainScaleS Wafer-Scale System*. Master's thesis, Universität Heidelberg (2019); http://www.kip.uni-heidelberg.de/Veroeffentlichungen/details.php?id=3909
59. Friedmann, S. et al. Demonstrating hybrid learning in a flexible neuromorphic hardware system. *IEEE Trans. Biomed. Circuits Syst.* **11**, 128–142 (2017).
60. Prodromakis, T. & Toumazou, C. A review on memristive devices and applications. In *Proc. 2010 17th IEEE International Conference on Electronics, Circuits and Systems* 934–937 (IEEE, 2010).
61. Esser, S. K., Appuswamy, R., Merolla, P., Arthur, J. V. & Modha, D. S. Backpropagation for energy-efficient neuromorphic computing. In *Advances in Neural Information Processing Systems* 1117–1125 (NIPS, 2015).
62. van De Burgt, Y., Melianas, A., Keene, S. T., Malliaras, G. & Salleo, A. Organic electronics for neuromorphic computing. *Nat. Electron.* **1**, 386–397 (2018).
63. Wunderlich, T. et al. Demonstrating advantages of neuromorphic computation: a pilot study. *Front. Neurosci.* **13**, 260 (2019).
64. Feldmann, J., Youngblood, N., Wright, C., Bhaskaran, H. & Pernice, W. All-optical spiking neurosynaptic networks with self-learning capabilities. *Nature* **569**, 208–214 (2019).
65. Kriener, L., Göltz, J. & Petrovici, M. A. The yin-yang dataset. Preprint at https://arxiv.org/pdf/2102.08211.pdf (2021).
66. LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **86**, 2278–2324 (1998).
67. Schemmel, J., Billaudelle, S., Dauer, P. & Weis, J. Accelerated analog neuromorphic computing. Preprint at https://arxiv.org/pdf/2003.11996.pdf (2020).
68. Comsa, I. M. et al. Temporal coding in spiking neural networks with alpha synaptic function. In *Proc. 2020 IEEE International Conference on Acoustics, Speech and Signal Processing* (*ICASSP*) 8529–8533 (IEEE, 2020).
69. Tavanaei, A., Kirby, Z. & Maida, A. S. Training spiking ConvNets by STDP and gradient descent. In *Proc. 2018 International Joint Conference on Neural Networks* (*IJCNN*) 1–8 (IEEE, 2018).
70. Aamir, S. A. et al. An accelerated LIF neuronal network array for a large-scale mixed-signal neuromorphic architecture. *IEEE Trans. Circuits Syst. I Regular Papers* **65**, 4299–4312 (2018).
71. Petrovici, M. A. et al. Characterization and compensation of network-level anomalies in mixed-signal neuromorphic modeling platforms. *PLoS ONE* **9**, e108590 (2014).
72. Cramer, B. et al. Training spiking multi-layer networks with surrogate gradients on an analog neuromorphic substrate. Preprint at https://arxiv.org/pdf/2006.07239.pdf (2020).
73. Petrovici, M. A. *Form Versus Function: Theory and Models for Neuronal Substrates* (Springer, 2016).
74. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R. & Bengio, Y. Quantized neural networks: training neural networks with low precision weights and activations. *J. Mach. Learn. Res.* **18**, 6869–6898 (2017).
75. Payeur, A., Guerguiev, J., Zenke, F., Richards, B. A. & Naud, R. Burst-dependent synaptic plasticity can coordinate learning in hierarchical circuits. Preprint at *bioRxiv* https://doi.org/10.1101/2020.03.30.015511 (2020).
76. Sacramento, J., Ponte Costa, R., Bengio, Y. & Senn, W. Dendritic cortical microcircuits approximate the backpropagation algorithm. In *Advances in Neural Information Processing Systems* Vol. 31, 8721–8732 (NIPS, 2018).
77. Aamir, S. A. et al. A mixed-signal structured AdEx neuron for accelerated neuromorphic cores. *IEEE Trans. Biomed. Circuits Syst.* **12**, 1027–1037 (2018).
78. Müller, E. et al. Extending BrainScaleS OS for BrainscaleS-2. Preprint at https://arxiv.org/pdf/2003.13750.pdf (2020).

79. Paszke, A. et al. PyTorch: an imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems* Vol. 32, 8024–8035 (NIPS, 2019).
80. Kingma, D. P. & Ba, J. Adam: a method for stochastic optimization. Preprint at https://arxiv.org/pdf/1412.6980.pdf (2014).
81. Göltz, J. et al. Fast and energy-efficient neuromorphic deep learning with first-spike times (Zenodo, 2021); https://doi.org/10.5281/zenodo.5115007
82. Stromatias, E. et al. Scalable energy-efficient, low-latency implementations of trained spiking deep belief networks on SpiNNaker. In *Proc. 2015 International Joint Conference on Neural Networks* (*IJCNN*) 1–8 (2015).
83. Renner, A., Sheldon, F., Zlotnik, A., Tao, L. & Sornborger, A. The backpropagation algorithm implemented on spiking neuromorphic hardware. Preprint at https://arxiv.org/pdf/2106.07030.pdf (2021).
84. Chen, G. K., Kumar, R., Sumbul, H. E., Knag, P. C. & Krishnamurthy, R. K. A 4096-neuron 1M-synapse 3.8-pJ/SOP spiking neural network with on-chip STDP learning and sparse weights in 10-nm FinFET CMOS. *IEEE J. Solid State Circuits* **54**, 992–1002 (2018).

Acknowledgements
We thank J. Jordan and N. Gürtler for valuable discussions, S. Schmitt for assistance with BrainScaleS-1, V. Karasenko, P. Spilger and Y. Stradmann for taming physics, as well as M. Davies and Intel for their ongoing support (L.K., W.S., M.A.P.). Some calculations were performed on UBELIX, the HPC cluster at the University of Bern. Our work has greatly benefitted from access to the Fenix Infrastructure resources, which are partially funded from the European Union's Horizon 2020 research and innovation programme through the ICEI project under grant agreement no. 800858. Some simulations were performed on the bwForCluster NEMO, supported by the state of Baden–Württemberg through bwHPC and the German Research Foundation (DFG) through grant no. INST 39/963-1 FUGG. We gratefully acknowledge funding from the European Union for the Human Brain Project under grant agreements 604102 (J.S., K.M., M.A.P.), 720270 (S.B., O.B., B.C., J.S., K.M., M.A.P.), 785907 (S.B., O.B., B.C., W.S., J.S., K.M., M.A.P.), 945539 (L.K., A.B., S.B., O.B., B.C., W.S., J.S., M.A.P.) and the Manfred Stärk Foundation (J.G., A.B., D.D., A.F.K., K.M., M.A.P.).

Author contributions
J.G., A.B. and M.A.P. designed the conceptual and experimental approach. J.G. derived the theory, implemented the algorithm and performed the hardware experiments. L.K. embedded the algorithm into a comprehensive training framework and performed the simulation experiments. A.B. and O.B. offered substantial software support. S.B., B.C., J.G. and A.F.K. provided low-level software for interfacing with the hardware. J.G., L.K., D.D., S.B. and M.A.P. wrote the manuscript.

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/s42256-021-00388-x.

Correspondence and requests for materials should be addressed to J. Göltz, L. Kriener or M. A. Petrovici.

Peer review information *Nature Machine Intelligence* thanks the anonymous reviewers for their contribution to the peer review of this work.

nature portfolio

https://doi.org/10.1038/s42256-021-00388-x

Supplementary information

Fast and energy-efficient neuromorphic deep learning with first-spike times

In the format provided by the authors and unedited

Supplementary Information

SI.A Learning with time-to-first-spike (TTFS) coding on BrainScaleS-1

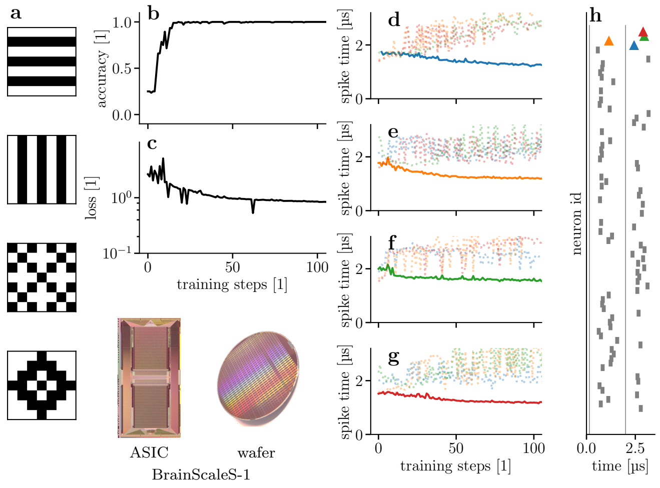

Figure SI.A1: Training a spiking network on the wafer-scale BrainScaleS-1 system. **(a)** Simple data set consisting of 4 classes with 7×7 input pixels. Accuracy **(b)** and loss **(c)** during training of the four pattern data set. **(d-g)** Evolution of the spike times in the label layer for the four different patterns. In each, the neuron coding the correct class is shown with a solid line and in full color. **(h)** Raster plot for the second pattern (e, correct class ▲) after training.

To demonstrate the applicability of our approach to different neuromorphic substrates, we also tested it on the BrainScaleS-1 system [1]. This version of BrainScaleS has a very similar architecture to BrainScaleS-2, but its component chips are interconnected through post-processing on their shared wafer (wafer-scale integration). More importantly for our coding scheme and learning rules, its circuits emulate conductance-based (CoBa) instead of current-based (CuBa) neurons. Furthermore, due to the different fabrication technology and design choices [in particular, the floating-gate parameter memory, see 1–3], the parameter variability and spike time jitter are significantly higher than on BrainScaleS-2 [4].

The training procedure was analogous to the one used on BrainScaleS-2 although using a different code base. To accommodate the CoBa synapse dynamics, we introduced global weight scale factors that modeled the distance between reversal and leak potentials and the total conductance, which were multiplied to the synaptic weights to achieve a CuBa. This approximation could then be trained with our learning rules. Despite this approximation and the considerable substrate variability, our framework was able to compensate well and classify the data set (Fig. SI.A1) correctly after only few training steps.

SI.B Additional experiments

In addition to the simulation results collected in Table 2 we provide additional training results on the MNIST data set here (Table SI.B1). We quantify the effect of noisy input spike times on generalization by comparing a noiseless training run and a run with input noise, both using the hyperparameters shown in Table SI.F1. Additionally, we train a network with a larger hidden layer as well as a deeper network with two hidden layers. Finally, we illustrate the effect of the weight quantization on the training of the MNIST data set by using the same 6-bit quantization as on the BrainScaleS-2.

Table SI.B1: Additional simulation runs on the MNIST data set. The values given as the baseline are taken from Table 2. With the noted exception of training length. Apart from the number of training epochs (see footnotes), the hyperparameters for simulations with the input resolution of 28×28 are the same as in Table SI.F1 and the simulations for the input resolution of 16×16 used the hyperparameters given in Table SI.F2.

simulation	**input resolution**	**hidden neurons**	**accuracy** [%] **test**	**train**
baseline	28×28	350	97.1 ± 0.1	99.6 ± 0.1
without noise	28×28	350	95.7 ± 0.3	99.7 ± 0.1
larger hidden layer	28×28	800	97.3 ± 0.1	99.8 ± 0.1
two hidden layers[1]	28×28	400-400	97.1 ± 0.1	99.5 ± 0.1
baseline[2]	16×16	246	97.4 ± 0.2	99.2 ± 0.1
6-bit weight resolution[2]	16×16	246	97.3 ± 0.1	99.1 ± 0.1

[1] This network was trained for 300 epochs.
[2] This network was trained for 150 epochs.

SI.C Robustness to post-training variations

We have already shown that our learning mechanism is able to cope well with noise and parameter variability which are present during training (Figs. 4 and 5). In addition to these distortions which can be accounted for by the learning mechanism, it is interesting to measure the performance of the trained network under adverse effects that were not present during training. This is especially relevant for analog circuits where, for example, temperature changes can lead to shifts in the analog neuron parameters. We model this effect by training 10 networks on the MNIST data set using the ideal parameters of $\vartheta = 1$ and $\tau_s = \tau_m = 1$ for the neuron threshold and time constant and then evaluating their performance on the test data set for shifted values of the threshold and time constant (Fig. SI.C1 a, b). These simulations show that the trained networks cope well, even if the relative shifts to the parameters are much larger than what can be typically expected due to temperature changes on BrainScaleS-2.

Furthermore, we consider a scenario which is less likely on neuromorphic platforms, but may be more relevant in biological networks. In biology, neural networks have to be robust against the death of neuron cells within the network. For each of the 10 fully trained networks we delete a percentage of its hidden population and evaluate the performance on the test set. As the consequences of this procedure strongly depend on exact choice of the deleted neurons, we repeat each deletion scenario for each network 10 times with different random seeds. Figure SI.C1c shows that networks trained with our learning mechanism exhibit stability towards sudden neuron death after training and even for 5 % neuron death the bound of the second quartile is still at 92.3 % accuracy. Note also that if plasticity is ongoing, the network can learn to recover much of its performance after apoptosis.

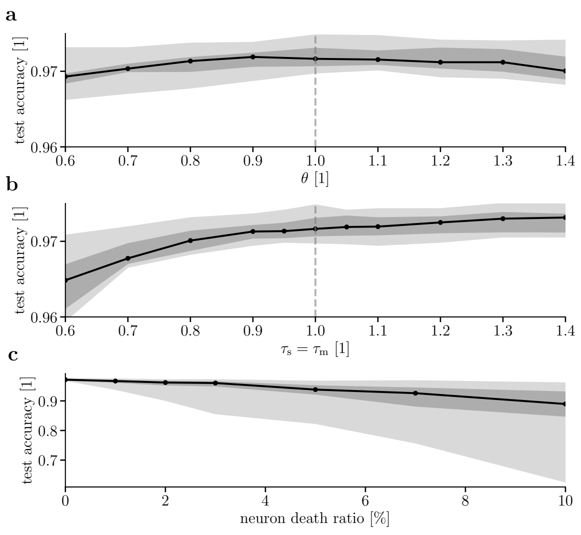

Figure SI.C1: Robustness to variations not present during training. All panels show median (black), quartiles (dark gray), as well as the entire range between minimum and maximum (light gray) in the shaded regions. **(a)** Dependence of test accuracy for evaluation for 10 trained networks with shifted threshold value θ. **(b)** Test accuracies for shifts in the neuron time constant τ_s and τ_m. **(c)** Influence of random deletion of hidden neurons on test accuracies. For each neuron death ratio, 10 different random sets of hidden neurons were deleted. These ten deletion sets were applied to the same ten networks as in (a) and (b).

SI.D Simplification of the learning rule

The learning rule for $\tau_m = \tau_s$ described in the main paper and derived in the Methods is computationally rather demanding: it needs multiple evaluations of the exponential function as well as an evaluation of the Lambert W function $\mathcal{W}$, for which no closed form exists. As the computational complexity of plasticity mechanisms on many neuromorphic

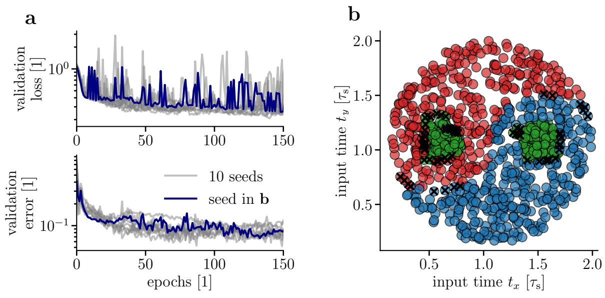

Figure SI.D1: Training on the Yin-Yang data set with a simplified learning rule. We approximated the learning rule to have less complex updates (Eqns. SI.D1 and SI.D2). **(a)** shows the training process of 150 epochs. We reach a test accuracy of (91.7 ± 1.4) % and training accuracy (91.7 ± 1.2) % averaged over 10 seeds. **(b)** shows the classification as in Fig. 2 after training for the highlighted training in (a).

chips is limited, we investigate the possibility of approximating the derivatives Eqns. (4) and (5) by replacing the exponential functions as well as $\mathcal{W}$ by a constant λ[1]:

$$\frac{\partial t_k}{\partial w_{ki}} = -\lambda \left(t_k - t_i\right) , \qquad \text{(SI.D1)}$$

$$\frac{\partial t_k}{\partial t_i} = -\lambda \frac{w_{ki}}{\tau_s} \left(t_k - t_i - \tau_s\right) . \qquad \text{(SI.D2)}$$

The approximated version consists only of simple differences and multiplications making this learning rule more amenable for on-chip implementations.

To examine the approximated learning rule in the standard setup with $\tau_m = \tau_s$ we chose $\lambda = 0.0192$ by evaluating $\frac{1}{a_1}\frac{1}{\mathcal{W}(z)+1}$ for a few inputs into the hidden layer. Using this extreme simplification we trained a network to classify the Yin-Yang data set (Fig. SI.D1). While the network learned the task correctly and achieved a test accuracy of (91.7 ± 1.4) %, this represents a small but noticeable drop in accuracy compared to the full learning rule (Table 2). We also observed that these simplified rules led to more instability for longer training periods (not shown here). Nonetheless, these promising results give us confidence that that a more careful choice of the constant or a replacement with a simple, but non-constant term can alleviate these problems while retaining a simple form of the learning rule.

Note, in particular, that Eqn. (SI.D2) explicitly contains the term $t_i + \tau_s$. This term represents the maximum of a postsynaptic potential (PSP) and changes sign when the output spike at T happens before versus after the maximum. This simple difference captures the major non-monotonic relationship in the time derivative. As the maximum of the PSP is given by a closed form solution

[1]This effectively leads to ρ being a constant in Eqns. (39) and (40).

$t_{\max} = t_i + \frac{\tau_\mathrm{m}\tau_\mathrm{s}}{\tau_\mathrm{s}-\tau_\mathrm{m}} \log\frac{\tau_\mathrm{s}}{\tau_\mathrm{m}}$ for arbitrary combinations of τ_s and τ_m, it seems natural to investigate a slightly altered learning rule for different time constants.

SI.E Power consumption and execution time measurements

Table 1 in the main manuscript compares the energy consumption and classification speed of our model on BrainScaleS-2 with other neuromorphic platforms and an ANN on a GPU. This section details how the power and classification speed measurements were performed, as well as their implications for the scalability of and potential improvements to our setup. Additionally, we present our measurement technique for the GPU reference.

SI.E.1 BrainScaleS-2

Power breakdown The full BrainScaleS-2 chip consumed a total of 175 mW measured during runtime with the INA219 chip [5]. This overall figure encompasses the chip's high-speed communication links (approx. 60 mW), the digital periphery as well as its clocking infrastructure (approx. 80 mW), and the biasing of analog circuits (approx. 35 mW). Importantly, we could not observe a significant change in power consumption between the network during inference and an emulation of an inactive network. This implies that the cost of event transport and synaptic processing is negligible on the reported scales and that the overall figure would not be impacted by increased activity levels. As inactive synapses mostly contribute to the overall power consumption through negligible leakage currents, the power consumption would not be impacted by an increase of the neuron circuit's fan-in that would allow the training on larger input spaces.

Execution time breakdown We define the round-trip time for an on-chip inference run as starting before the forward pass through the network in our PyTorch implementation and ending when all classification results produced by the chip are available in PyTorch. For the classification of the full MNIST test data set on BrainScaleS-2 we measured a round-trip time of 0.937 s.

Due to this conservative definition of the round-trip time, our measurement includes a significant amount of time on the host (for data pre- and post-processing) and for communication between host and the neuromorphic system. Fig. SI.E1 shows a detailed breakdown of the execution time. We see that once the data arrives on the chip, it takes 480 ms to process the 10 000 images of the test set. This results in a classification every 48 µs or equivalently a classification rate of 20 800 images per second.

Considering the relevant hardware time constants of $\tau_\mathrm{s} \approx \tau_\mathrm{m} \approx 6\,\mu\mathrm{s}$ and the typical time to solution of around $1\,\tau_\mathrm{s}$ to $1.5\,\tau_\mathrm{s}$, a classification duration per sample of 48 µs seems surprisingly long. This is owed to the sequential presentation of data samples to the network, for which we need to ensure that all residual activity – membrane voltages as well as synaptic currents – from the last sample has fully decayed before the next sample is presented. Currently, this is achieved by simply waiting between samples, but Cramer et al. [6] present an alternative: The plasticity processing unit (PPU) is able to trigger a reset of all membrane voltages and synaptic currents on the chip. Using this mechanism, Cramer et al. [6] shorten the classification time per image to 11.8 µs. The usage of artificial resets would also be a viable optimization for our model. It would require the previously switched off PPU to be activated and would therefore slightly increase the power consumption (by approximately 20 mW). This increase in power would however be more than compensated by the approximately quadrupled sample throughput.

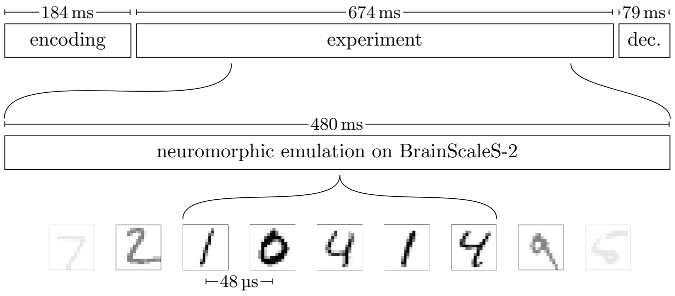

Figure SI.E1: Breakdown of the execution time for inference on the MNIST test set. The total time of about one second consists of an encoding, an experiment and a decoding phase. The encoding phase includes the translation of PyTorch tensors into spike trains and the encoding of the spike trains into instructions for the neuromorphic chip. In the experiment phase the instructions are sent from the host to the chip, the emulation is performed and the results are read out from the chip and communicated back to the host. In the final decoding phase the emulations results are converted back to PyTorch tensors.

SI.E.2 GPU

For comparison to conventional computing hardware we add power and classification speed measurements on a Nvidia Tesla P100 GPU to Table 1. For the measurements on the GPU we use the convolutional neural network given as standard example in the PyTorch repository (`https://github.com/pytorch/examples/blob/507493d7b5fab51d55af88c5df9eadceb144fb67/mnist/main.py`).

The power measurements are performed using the tool nvidia-smi which is running in the background while in the foreground we run a PyTorch program which repeats the classification of the MNIST test data set (in one batch of size 10 000) for 10 times. Figure SI.E2 shows the power consumption over the full program runtime. We see that the GPU is only active for 10 short periods, the duration of which match the measured times during which the PyTorch program uses the GPU (Fig. SI.E2 b). The power consumption is calculated as an average over these intervals, resulting in 106.5 W.

The speed measurements were performed using time measurements in Python and averaged over the 10 classifications, resulting in a classification time per image of 8 µs. This amounts to an energy-per-classification value of 852 µJ.

As an additional reference we attempted to determine the power consumption and classification speed for a fully connected network with a hidden layer of 246 neurons (same size as the hidden layer on BrainScaleS-2) on GPU. However, due to the fact that the classification was a factor of 20 to 25 faster than for the CNN, we were not able to measure the power in a fine enough resolution with nvidia-smi to yield reliable values. To estimate a lower-bound for the energy per classification in this case, we can take the power consumption of the GPU in the phases where it was not actively used in the CNN measurement (i.e. power values between the peaks in Fig. SI.E2a) which is approximately 34 W. This “idle” power consumption for the CNN case seemed to approximately match the averaged power drain for the fully connected network. This amounts to a lower-bound estimate of the energy-per-classification value on the order of 10 µJ.

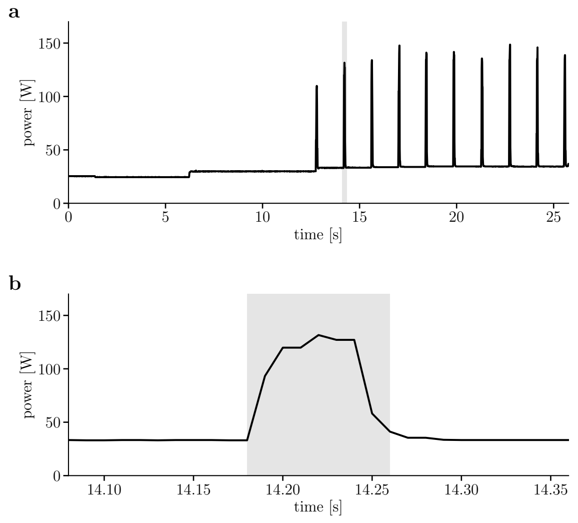

Figure SI.E2: Power consumption of Nvidia Tesla P100 GPU during classification of MNIST test data. (a) Power consumption of a standard PyTorch network for MNIST classification while running inference on the test data set for 10 times. **(b)** Zoom on a peak in the power consumption. The shaded area corresponds to the time during which the GPU is actively used (measured from within Python).

SI.F Additional data

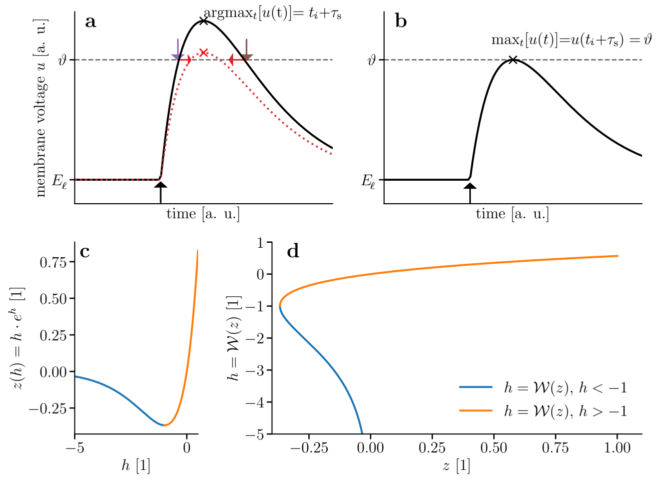

Figure SI.F1: (a) Membrane dynamics for one strong input spike at t_i (upward arrow) with two threshold crossings due to leak pullback (earlier violet, later brown). The change induced by a reduction of the input weight is shown in red. **(b)** Edge case without crossing and exactly one time where $u(t) = \vartheta$. **(c)** Defining relation for the Lambert W function $\mathcal{W}$, evidently not an injective map. **(d)** Distinguishing between $h \lesseqgtr -1$ allows to define the inverse function of (c), the Lambert W function $\mathcal{W}$.

Table SI.F1: Neuron, network and training parameters used to produce the results in Figs. 2 and 3.

Parameter name	**Yin-Yang**	**MNIST**
Neuron parameters		
g_ℓ	1.0	1.0
E_ℓ	0.0	0.0
ϑ	1.0	1.0
τ_m	1.0	1.0
τ_s	1.0	1.0
Network parameters		
size input	5	784
size hidden layer	120	350
size output layer	3	10
bias time[1]	$[0.9\tau_\mathrm{s}, 0.9\tau_\mathrm{s}]$	no bias
weight init mean[1]	$[1.5, 0.5]$	$[0.05, 0.15]$
weight init stdev[1]	$[0.8, 0.8]$	$[0.8, 0.8]$
t_early	$0.15\tau_\mathrm{s}$	$0.15\tau_\mathrm{s}$
t_late	$2.0\tau_\mathrm{s}$	$2.0\tau_\mathrm{s}$
Training parameters		
training epochs	300	150
batch size	150	80
optimizer	Adam	Adam
Adam parameter β	$(0.9, 0.999)$	$(0.9, 0.999)$
Adam parameter ϵ	10^{-8}	10^{-8}
learning rate	0.005	0.005
lr-scheduler	StepLR	StepLR
lr-scheduler step size	20	15
lr-scheduler γ	0.95	0.9
input noise σ	no noise	0.3
max ratio missing spikes[1]	$[0.3, 0.0]$	$[0.15, 0.05]$
max allowed Δw	0.2	0.2
weight bump value	0.0005	0.005
α	0.005	0.005
ξ [2]	0.2	0.2

[1] Parameter given layer wise [hidden layer, output layer].

[2] ξ implemented differently in code-base developed by the authors.

Table SI.F2: Network and training parameters for training on BrainScaleS-2 used to produce the results in Fig. 4. In contrast to Table SI.F1, the neuron parameters are not given here, as they are determined by the used chip.

Parameter name	**Yin-Yang**	**16×16 MNIST**
Network parameters		
size input	25	256
size hidden layer	120	246
size output layer	3	10
bias time[1]	$[0.9\tau_\mathrm{s}, \text{no bias}]$	no bias
weight init mean[1]	$[0.1, 0.075]$	$[0.01, 0.006]$
weight init stdev[1]	$[0.12, 0.15]$	$[0.03, 0.1]$
t_early	$0.15\tau_\mathrm{s}$	$0.15\tau_\mathrm{s}$
t_late	$2.0\tau_\mathrm{s}$	$2.0\tau_\mathrm{s}$[3]
Training parameters		
training epochs	400	50
batch size	40	50
optimizer	Adam	Adam
Adam parameter β	$(0.9, 0.999)$	$(0.9, 0.999)$
Adam parameter ϵ	10^{-8}	10^{-8}
learning rate	0.002	0.003
lr-scheduler	StepLR	StepLR
lr-scheduler step size	20	10
lr-scheduler γ	0.95	0.9
input noise σ	no noise	0.3
max ratio missing spikes[1]	$[0.3, 0.05]$	$[0.5, 0.5]$
max allowed Δw	0.2	0.2
weight bump value	0.0005	0.005
α	0.005	0.005
ξ [2]	0.2	0.2

[1] Parameter given layer wise [hidden layer, output layer].

[2] ξ implemented differently in code-base developed by the authors.

[3] After translation of pixel values to spike times, inputs spikes with $t_\mathrm{input} = t_\mathrm{late}$ were not sent into the network.

Table SI.F3: Extension of literature review for pattern recognition models on neuromorphic back-ends, including results which do not detail certain measurements.

platform	**type**	**coding**	**network size/structure**	**energy per classification**	**classifications per second**	**test accuracy**	**reference**
SpiNNaker	digital	rate	764-600-500-10	3.3 mJ	91	95.0 %	[7], 2015
True North[1]	digital	rate	CNN	0.27 µJ	1000	92.7 %	[8], 2015
True North[1]	digital	rate	CNN	108 µJ	1000	99.4 %	[8], 2015
FPGA (nLIF neurons)[2]	digital	temporal	784-600-10	-	-	96.8 %	[9], 2017
Loihi	digital	bin. rate	400-400-10	2.5 µJ	5917	96.2 %	[10], 2021
Loihi[3]	digital	temporal	1920-10	-	-	96.4 %	[11], 2018
unnamed (Intel)[4]	digital	temporal	236-20	1.0 µJ	6250	88.0 %	[12], 2018
unnamed (Intel)[5]	digital	temporal	784-1024-512-10	12.4 µJ	-	98.2 %	[12], 2018
unnamed (Intel)[5]	digital	temporal	784-1024-512-10	1.7 µJ	-	97.9 %	[12], 2018
SPOON[6]	digital	temporal	CNN	0.3 µJ	8547	97.5 %	[13], 2020
BrainScaleS-2	mixed	temporal	256-246-10	8.4 µJ	20 800	96.9 %	this work

[1] In [8] it is stated that "The instrumentation available measures active power for the network in operation and leakage power for the entire chip, which consists of 4096 cores. We report energy numbers as active power plus the fraction of leakage power for the cores in use.". For the first result 5 cores were used, while the second result requires 1920 cores.
[2] No energy or speed measurements reported.
[3] No energy or speed measurements reported. Images were preprocessed with an algorithm described as "using scan-line encoders".
[4] Images preprocessed with 4 5×5 Gabor filters and 3×3 pooling.
[5] No speed measurements reported.
[6] Reported energy values are pre-silicon simulations.

References

1. Schemmel, J. et al. A wafer-scale neuromorphic hardware system for large-scale neural modeling. *Proceedings of 2010 IEEE International Symposium on Circuits and Systems,* 1947–1950 (2010).
2. Srowig, A. et al. Analog floating gate memory in a 0.18 μm single-poly CMOS process. *Internal FACETS Documentation* (2007).
3. Koke, C. Device Variability in Synapses of Neuromorphic Circuits. *PhD thesis Heidelberg University* (2017).
4. Schmitt, S. et al. Neuromorphic hardware in the loop: Training a deep spiking network on the brainscales wafer-scale system. *2017 International Joint Conference on Neural Networks (IJCNN),* 2227–2234 (2017).
5. *INA219* Rev. G. Texas Instruments (Dec. 2015). `https://www.ti.com/lit/ds/symlink/ina219.pdf`.
6. Cramer, B. et al. Training spiking multi-layer networks with surrogate gradients on an analog neuromorphic substrate. *arXiv preprint arXiv:2006.07239* (2020).
7. Stromatias, E. et al. Scalable energy-efficient, low-latency implementations of trained spiking deep belief networks on spinnaker. *2015 International Joint Conference on Neural Networks (IJCNN),* 1–8 (2015).
8. Esser, S. K., Appuswamy, R., Merolla, P., Arthur, J. V. & Modha, D. S. Backpropagation for energy-efficient neuromorphic computing. *Advances in Neural Information Processing Systems,* 1117–1125 (2015).
9. Mostafa, H., Pedroni, B. U., Sheik, S. & Cauwenberghs, G. Fast classification using sparsely active spiking networks. *2017 IEEE International Symposium on Circuits and Systems (ISCAS),* 1–4 (May 2017).
10. Renner, A., Sheldon, F., Zlotnik, A., Tao, L. & Sornborger, A. The Backpropagation Algorithm Implemented on Spiking Neuromorphic Hardware. *arXiv preprint arXiv:2106.07030* (2021).
11. Lin, C.-K. et al. Programming spiking neural networks on intel's loihi. *Computer* **51,** 52–61 (2018).
12. Chen, G. K., Kumar, R., Sumbul, H. E., Knag, P. C. & Krishnamurthy, R. K. A 4096-neuron 1M-synapse 3.8-pJ/SOP spiking neural network with on-chip STDP learning and sparse weights in 10-nm FinFET CMOS. *IEEE Journal of Solid-State Circuits* **54,** 992–1002 (2018).
13. Frenkel, C., Legat, J.-D. & Bol, D. A 28-nm Convolutional Neuromorphic Processor Enabling Online Learning with Spike-Based Retinas. *2020 IEEE International Symposium on Circuits and Systems (ISCAS),* 1–5 (2020).

6. Synchronization and semantization in deep spiking networks

The manuscript *Synchronization and semantization in deep spiking networks* has been submitted for publication in a peer-reviewed journal and is currently available as a preprint on *arXiv* as

> Jonas Oberste-Frielinghaus, Anno C. Kurth, **Julian Göltz**, Laura Kriener, Junji Ito, Mihai A. Petrovici and Sonja Grün (2025). *Synchronization and semantization in deep spiking networks.* arXiv: `2508.12975 [q-bio.NC]`. DOI: `10.48550/arXiv.2508.12975`.

Author contributions

Conceptualization, JOF, JI, MAP, SG; Methodology, JG, LK, MAP; Software, JOF, ACK, JG, LK; Formal Analysis JOF, ACK, JI, SG; Investigation, JG, LK, MAP; Visualization, JOF, ACK; Writing – Original Draft, JOF, ACK, JG, LK, JI, MAP, SG; Writing – Review & Editing, JOF, ACK, JG, LK, JI, MAP, SG; Funding Acquisition, MAP, SG; Resources, MAP, SG; Project Administration, JOF, SG; Supervision, JI, MAP, SG.

Manuscript

In accordance with the CC BY-NC-ND licence, the manuscript is reprinted in the following.

Synchronization and semantization in deep spiking networks

Jonas Oberste-Frielinghaus[1,2*], Anno C. Kurth[1,3], Julian Göltz[4,5], Laura Kriener[6,5], Junji Ito[1], Mihai A. Petrovici[5], Sonja Grün[1,7,8]

[1] Institute for Advanced Simulation (IAS-6), Jülich Research Centre, Jülich, Germany
[2] RWTH Aachen University, Aachen, Germany
[3] RIKEN Center for Brain Science, Wako, Saitama, Japan
[4] Kirchhoff-Institute for Physics, Heidelberg University, Heidelberg, Germany
[5] Department of Physiology, University of Bern, Bern, Switzerland
[6] Institute of Neuroinformatics, University of Zurich and ETH Zurich, Zurich, Switzerland
[7] JARA Brain Institute I (INM-10), Jülich Research Centre, Jülich, Germany
[8] Theoretical Systems Neurobiology, RWTH Aachen University, Germany

[*] j.oberste-frielinghaus@fz-juelich.de

keywords: spiking neural networks, network dynamics, computation, latency coding, synchrony, artificial intelligence, training algorithm

Abstract

Recent studies have shown how spiking networks can learn complex functionality through error-correcting plasticity, but the resulting structures and dynamics remain poorly studied. To elucidate how these models may link to observed dynamics in vivo and thus how they may ultimately explain cortical computation, we need a better understanding of their emerging patterns. We train a multi-layer spiking network, as a conceptual analog of the bottom-up visual hierarchy, for visual input classification using spike-time encoding. After learning, we observe the development of distinct spatio-temporal activity patterns. While input patterns are synchronous by construction, activity in early layers first spreads out over time, followed by re-convergence into sharp pulses as classes are gradually extracted. The emergence of synchronicity is accompanied by the formation of increasingly distinct pathways, reflecting the gradual semantization of input activity. We thus observe hierarchical networks learning spike latency codes to naturally acquire activity patterns characterized by synchronicity and separability, with pronounced excitatory pathways ascending through the layers. This provides a rigorous computational hypothesis for the experimentally observed synchronicity in the visual system as a natural consequence of deep learning in cortex.

Significance Statement

Recent advances in AI have rekindled the hypothesis of deep learning in the brain, but there remains a significant gap at the microscopic scale, as cortical neurons communicate with sparse and discrete signals, rather than continuously in time. Building on an analytical model of deep learning with spikes, we investigate the emergence of spatio-temporal structures in hierarchical spiking networks. We find that neuronal populations learn to form tight pulse packets for downstream communication and observe distinct pathways of neuronal excitation that become increasingly separated with network depth, indicating the progressive semantization of neuronal activity. This puts forth a rigorous computational hypothesis for the well-established experimental observations of synchrony and semantization in sensory cortex.

Introduction

Artificial neuronal networks (ANNs) are the backbone of modern machine learning applications. Since the formulation of the perceptron (Rosenblatt, 1958), ANNs have gradually diverged away from the biology that originally inspired them, but their recent success across many domains has prompted a broad interest to reevaluate their applicability as models of processing in the brain (Richards et al., 2019). Many of these studies focus on visual processing, as it is among the best studied computational tasks, both in cortex and as an application for AI. As an example, Convolutional Neural Networks (LeCun et al., 1998; Krizhevsky et al., 2012) are used successfully as model of the visual system (Yamins and DiCarlo, 2016; Lindsay, 2021).

However, the underlying models remain very close or even identical to conventional ANNs, in particular by using continuous neuronal transfer functions. This is markedly different from cortical networks, in which interneuron communication is dominated by action potentials, or spikes, i.e., cortical networks are spiking neural networks (SNNs). A continuous transfer function can be approximated in SNNs by considering the average spike rates over time or populations of neurons, leading to an interpretation that the aforementioned models are operating in a purely rate coding framework. Even though rate coding has been highly influential in Neuroscience, may it be for characterizing response properties of single neuron (Hubel and Wiesel, 1962; Georgopoulos et al., 1982) or neural populations (from population rates (Georgopoulos et al., 1986; Churchland et al., 2012) to geometric interpretation of the evolution of the population vector (Gao et al., 2017; Gallego et al., 2017; Stringer et al., 2019; Morales-Gregorio et al., 2024)), rate coding is far from the only operational mode of the cortex. Alternative, well-established computational interpretations of cortical activity emphasize the fine temporal nature of neural activity, e.g., (Abeles, 1991; Thorpe et al., 2001; Izhikevich, 2006). They are supported by experimental findings such as the coordinated spiking on millisecond scale (Riehle et al., 1997; Prut et al., 1998; Kilavik et al., 2009; Torre et al., 2016) or characteristic temporal sequences of spikes (Yiling et al., 2023; Xie et al., 2024; Sotomayor-Gómez et al., 2025).

The main reason for using rate-based models, i.e., models that only communicate via firing rates emulating a continuous transfer function and not precise spikes, lies in the difficulty of training SNNs. Indeed, it is not obvious how to calculate gradients of discrete spiking activity, which would be necessary for a straightforward application of error backpropagation. However, recent years have seen the development of various approaches capable of overcoming this challenge, most notably approximate surrogate methods (Neftci et al., 2017; Zenke and Ganguli, 2018; Yin et al., 2023) and exact spike-time gradients (Bohte et al., 2002; Wunderlich and Pehle, 2021; Göltz et al., 2021). These now allow the training of deep spiking networks to performances comparable with their conventional counterparts. Thus, such networks can form the basis of a more rigorous reassessment of the deep learning hypothesis in the brain, now also taking into account a more realistic form of spike-based, as opposed to continuous, communication.

With trained networks it is possible to study how their structure and activity is shaped through learning and which characteristic patterns emerge. In particular, the aspects of propagation and transformation of the neural code (Perkel and Bullock, 1968) and their underlying mechanisms can be investigated thoroughly. There have been extensive studies about the propagation of activity in SNNs, e.g., in simulations (Diesmann et al., 1999; van Rossum et al., 2002; Vogels and Abbott, 2005) or in vitro (Reyes, 2003; Barral et al., 2019), but the studied networks were not trained to perform a particular task.

Here, we consider multi-layered SNNs trained by exact gradient descent as visual image classifiers using a spike latency code (Göltz et al., 2021). Thereby we approach their activity as we would approach electrophysiological recordings, but with the added benefit of having access to all observables in the network, as opposed to the massive subsampling that is characteristic of in-vivo data (Levina et al., 2022). In the following, we show how these networks form very distinct activity and connectivity patterns. In particular, we show that neuron subpopulations in these networks learn to synchronize their firing in response to patterns of a particular class. This is a phenomenon frequently observed in the cortex, e.g., (Gray and Singer, 1989; Gray et al., 1989)), but here we show that it arises from learning by gradient descent, thus providing a functional explanation. Moreover, we observe how these populations grow increasingly distinct across the network hierarchy, demonstrating the semantization of activity as it propagates downstream. This bundling of activity in space and time maps closely to various experimental observations, thus establishing a first step towards a rigorous link between the theory of learning by gradient descent in spiking networks and in-vivo recordings of cortical activity.

Results

Activity in the network

We investigate a feed-forward network with all-to-all connections between consecutive layers consisting of an input, four hidden, and an output layer as depicted in Figure 1a. As a classical visual benchmark that does not require complicated structures lacking direct biological equivalents, such as perfect copies of convolutional kernels or max-pooling layers, we chose classification of the MNIST dataset (LeCun et al., 1998) as task for the network. Importantly, and unlike in classical ANNs, the neurons in the hidden layers obey Dale's law (Eccles, 1957), meaning that each neuron has either only excitatory or only inhibitory outgoing connections. To roughly approximate the ratio found in cortex (Markram et al., 2004), each hidden layer consists of 300 excitatory and 100 inhibitory neurons. The output layer has 10 neurons, one for each image class.

To understand how the network processes the inputs, we first examine how spiking activity propagates through the layers in response to an arbitrary image of a handwritten digit (see Figure 1a) after training (Figure 1b). The input image is converted into a set of spike times that specify the activity of the input layer. Each neuron in the input layer corresponds to a pixel of the input image, the brightness of which determines whether the corresponding neuron fires earlier or later; the darker the pixel, the earlier the neuron fires. The spiking activity of the input layer is passed to the subsequent layer (layer 1), where incoming spikes influence the membrane potential of the neurons. If, for a given neuron in layer 1, the evoked membrane potential exceeds the threshold, the neuron emits a spike that is passed to all neurons in the next layer (layer 2), and so on. Typically, to bring a neuron to fire, it needs to receive sufficient synchronous

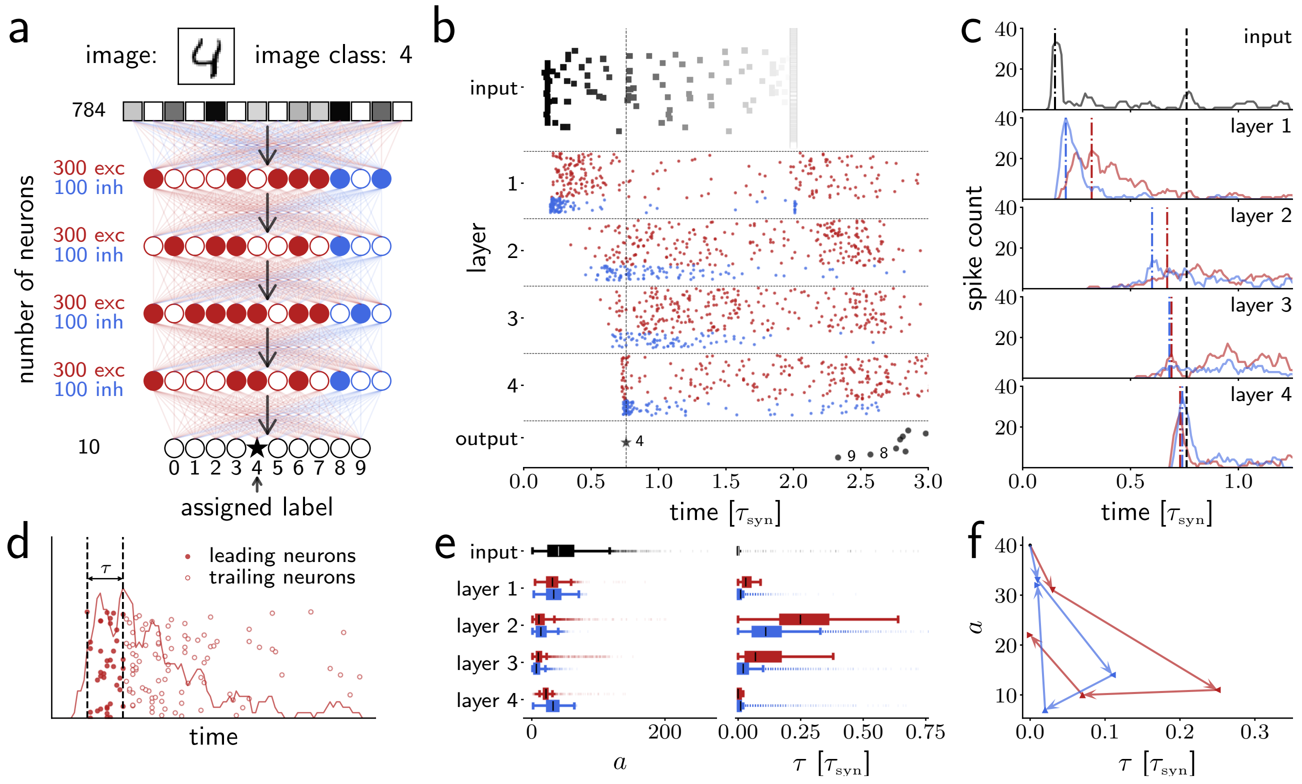

Figure 1: Propagation of activity through the network. (**a**) Structure of the network and the task. One randomly chosen image of class 4 is passed into the network via the input layer. Each pixel is represented by a neuron (784 in total). The input propagates through four hidden layers with 300 excitatory and 100 inhibitory neurons. The output layer contains 10 output neurons, each representing one class. The network assigns a label according to the neuron in the output layer that spikes first. (**b**) Activity in response to the image in **a** as raster plot. Time is shown as multiples of the synaptic time constant τ_{syn}, so all our conclusions remain scale-invariant with respect to the specific time constants in the network. The dots represent the spike times of the individual neurons. The image is represented by a latency code via spike times in the input layer. The brightness of the dot corresponds to the brightness of the pixel in the image (the darker, the earlier). In the hidden layers (1-4) red dots correspond to excitatory, blue dots to inhibitory neurons. The first spike in the output layer is marked by an asterisk. The image is classified correctly as “4”. (**c**) Spike time histogram for the activity in **b**. The spike count was measured in a sliding window of 0.05 τ_{syn} in 0.01 τ_{syn} steps. In hidden layer 1-4 the spike count is separated between excitatory and inhibitory neurons. The colored vertical lines as well as the first black line in the panel for the input layer denote the maximum of the histogram before the first output spike (dashed black line). (**d**) Illustration of the characterization of the activity. We determine the rise time τ as the time from the first spike to the maximum of the histogram, we term neurons active during this time *leading* neurons and the others *trailing* neurons. a is the number of leading neurons. (**e**) Box plots of the distributions of a (left) and τ (right) across all images, separate for excitatory and inhibitory neurons in the hidden layers. The line marks the median, the box marks the range between the first (Q1) and the third quantile (Q3), the whiskers range from the box to the lowest data point above $\mathrm{Q1} - 1.5(\mathrm{Q3} - \mathrm{Q1})$ and the highest data point below $\mathrm{Q3} + 1.5(\mathrm{Q3} - \mathrm{Q1})$, fliers represent outliers. (**f**) State space representation of the medians of the distributions in **e** in sequence of the layers. Arrows point in the direction of the propagation of activity in the network.

input from the preceding layer. This way, spiking activity propagates downstream, from layer to layer. In the output layer, the label assigned to the input is determined by which of the 10 output neurons emits a spike first. On the test dataset, the trained network achieves an accuracy of 0.98, i.e, the assigned label matches the image class of the input for 98% of the test images.

To examine the propagation of the activity quantitatively, we first calculate the spike time histogram for the spiking activity shown in Figure 1b. Starting in the input layer, we observe a sharp peak in the histogram (Figure 1c first row). Over the next two layers, the spike times spread out, and hence the histogram peak becomes less prominent. Then in layer 3, the excitatory neurons synchronize again, ultimately resulting in a very sharp peak for both the excitatory and inhibitory neurons in layer 4. Overall, these activity profiles resemble the propagation of a *pulse packet* i.e., a synchronous volley of spikes, through the layers, which first disperses and then re-synchronizes over the layers.

We characterize the pulse packets of excitatory and inhibitory neurons in each layer individually by, on the one hand, determining the rise time τ of the spike time histogram, i.e., the time from the first spike to the maximum of the histogram. The rise time gives an estimate of how synchronous the spikes occur in the layer. On the other hand, we count the number a of neurons that fire spikes during this time (see Figure 1d). These neurons we term “leading neurons”, and the neurons that fire after this period we term “trailing neurons”.

Figure 1e shows in the form of box plots the distributions of the number of leading neurons (a, left) and the rise time of the histogram (τ, right) across all images. Over all images the same trend as we observe in the example shown in Figure 1c solidifies; the activity starts with a high and sharp peak (large a and small τ), then gets dispersed (smaller a and larger τ), and then builds up again (see Figure 1f). This trend is evident for both the excitatory and the inhibitory neurons.

In summary, we see that the propagation of activity in the network is characterized by a pulse packet that decays and then builds up again. The conditions for networks to exhibit this kind of activity have been investigated extensively (Diesmann et al., 1999; Tetzlaff et al., 2002; Vogels and Abbott, 2005; Kumar et al., 2008; Shinozaki et al., 2010). More on this point will follow in the discussion.

While the characterization of the activity as a pulse packet allows a quantitative description of the activity propagation through the layers, it does not immediately provide functional implications of the observed activity for information processing. Assuming that the pulse packet plays a relevant role in achieving a correct classification, for a pulse packet to represent the image class of the input, it would need to encode the information by the identity of the neurons that contribute spikes to it. Since we observe that the pulse packet is gradually built up as it propagates through the layers, we also expect that its representation of the image class would be progressively consolidated towards deeper layers, i.e., a more specific subset of neurons would provide spikes to the pulse packet in deeper layers. This leads us to a close examination of the identity of the leading neurons, as shown in the following section.

Representation of classes in the activity

Next, we investigate how different classes are represented in the population activity of each layer. We focus on the leading neurons here because these neurons are likely most important for the classification, since the network operates on a latency code and these neurons fired spikes with the shortest latencies in the individual layer. Thereby we consider a set $\mathcal{V}_x^l$ of the leading neurons in layer l for image x (see Methods: Set of leading neurons for details).

Figure 2a shows, separately for each layer, the leading neurons $\mathcal{V}_x^l$ for 100 randomly chosen test images (10 for each class). While in the early layers no particular structure can be discerned, in the deeper layers certain neurons fire across all images of a particular class, forming a bar-code-like pattern. We note that these observations are not dependent on our specific way of defining $\mathcal{V}_x^l$; other equally plausible definitions of $\mathcal{V}_x^l$ lead to essentially identical observations (see Supplemental Information: Figure S1).

To quantify the consistency of the leading neurons in response to different images of the same class, we calculate the similarity $\rho_{x,y}^l$ of the leading neuron sets $\mathcal{V}_x^l$ and $\mathcal{V}_y^l$ in layer l for two respective images x and y as (Figure 2b):

$$\rho_{x,y}^l = \frac{N^l|\mathcal{V}_x^l \cap \mathcal{V}_y^l| - |\mathcal{V}_x^l||\mathcal{V}_y^l|}{\sqrt{|\mathcal{V}_x^l||\mathcal{V}_y^l|(N^l - |\mathcal{V}_x^l|)(N^l - |\mathcal{V}_y^l|)}} , \tag{1}$$

where N^l is the number of neurons in layer l and $|\mathcal{V}|$ denotes the cardinality of the set $\mathcal{V}$ (see Methods: Similarity of sets of leading neurons for details). If the two sets are identical, $\rho_{x,y}^l = 1$; if the activity is maximally dissimilar (which would be the case if half of the neurons were leading neurons for image x and the other half for image y), $\rho_{x,y}^l = -1$; $\rho_{x,y}^l = 0$ implies chance overlap. Figure 2b shows the similarity calculated for all pairs of images used in Figure 2a for all layers, again grouped by the image classes. Diagonal blocks correspond to similarities between images from the same class, while off-diagonal blocks quantify the similarity of the activity for images of different classes. In the input layer and hidden layer 1, there is little difference between within-class and between-class similarities. In layer 2, the degrees of the similarities within the diagonal blocks are higher than those in off-diagonal blocks, implying that images of the same class evoke more consistent activity than images of different classes. This trend solidifies as activity propagates across layers, reaching its maximum in layer 4, where neural representations of images from the same class are almost identical. The distribution of the overlap measures calculated for all pairs of the test images (Supplemental Information: Figure S2) confirms that this observation is not only for the 100 images randomly chosen here, but generally applies to all images.

To quantify the specificity of individual neurons in the representation of different image classes, we evaluate the Information Gain (IG) of a neuron, i.e., the information about the class of an input image gained by finding that neuron as a leading neuron for that image (for details see Methods: Information gain). An IG of 0 implies that the neural firing is independent of the class of the input image; an IG of 1 signifies that the neuron is fully indicative of a specific class. Figure 2c shows the distributions of IGs across all neurons of the respective layers, separately shown for excitatory (red) and inhibitory (blue) neurons, excluding neurons that were not a leading neuron for any image. In the input layer, we have a broad distribution of IGs with one peak roughly at 0.1 and another at 1.0. The latter represents the neurons that are leading neurons for exactly one image, thus being fully indicative of the class of that image. The IG distributions for excitatory neurons in the hidden layers shift more and more towards an IG of 1.0 in the deeper layers. In contrast, IGs of inhibitory neurons are generally low and do not grow towards deeper layers, indicating that inhibitory neurons are less specific for one particular image class than excitatory neurons throughout the layers. This is visualized by the two examples of the image class distribution for the neurons with the median IG in the respective layer and subpopulation (Figure 2c, inset) for all images when the neuron was a leading neuron. For example, the excitatory neuron in the last layer is almost exclusively active for images of class 4, while the activity

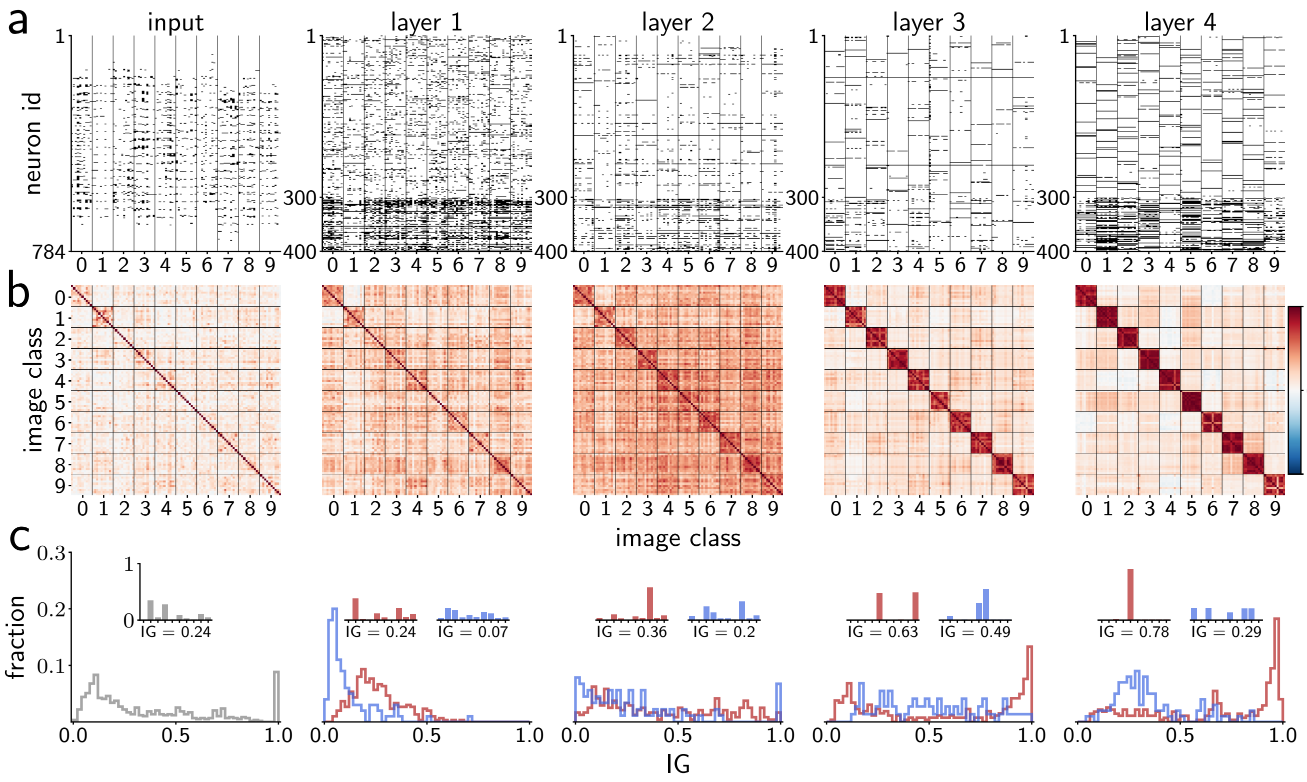

Figure 2: Representation of labels across the layers. (**a**) State of activity for all neurons for 100 images (10 randomly selected for each of the 10 classes), ordered according to the classes. If the neurons is a leading neuron for the image it is marked black, if it is a trailing neuron it is marked white (for the definition see Figure 1d). For the hidden layers (1-4) the inhibitory neurons are indexed with numbers between 301 and 400, the first 300 being excitatory neurons. (**b**) Matrices of similarities ρ of any two leading neuron sets shown in **a**, ordered by image class, and color-coded (see colorbar on the right). (**c**), Distributions of the specificity of all neurons per layer measured by the information gain (IG) and normalized by the number of neurons (neurons that are never active are excluded). On the very left this distribution is shown for the input neurons, then for the hidden layers (1-4) (left to right), separately for the excitatory (red) and inhibitory (blue) neurons. To illustrate the meaning of the IG, for each population, we choose a neuron with the median IG and plot the distribution of the image classes of the images if the neuron was a leading neuron in the inset.

of the inhibitory neuron is more broadly distributed. This is consistent with the findings in the visual cortex, where inhibitory neurons are more broadly tuned than excitatory neurons (Sohya et al., 2007; Niell and Stryker, 2008; Lundqvist et al., 2010).

Connectivity structure and path identification

So far we have concentrated on the neural activity, disregarding the knowledge of synaptic weights in the network. We now turn to the synaptic weights and ask if we can find a relation between the connectivity structure of the network and the specificity of the neural activity. In particular, we aim to identify neurons that have strong (direct or indirect) synaptic impacts on a specific output neuron. To this end, we focus only on excitatory neurons, since the high specificity in the representation of image classes was found almost exclusively for excitatory neurons.

Our procedure for connectivity structure analysis is schematically illustrated in Figure 3a. We start by considering one specific output neuron o ($o = 4$ and 9 in Figure 3a top and bottom, respectively). Then we identify neurons in the last hidden layer that are stronger connected to this output neuron o than to the other output neurons. The identified neurons (marked in red in Figure 3a) constitute a subset $\mathcal{P}_o^4$ of excitatory neurons in layer $l = 4$ with positive impact on the output neuron o, and complementarily, all the other excitatory neurons in layer 4 (marked in gray) are grouped into a subset $\mathcal{N}_o^4$ of neurons that do not have positive impact (for details see Methods: Assignment of neural subsets and paths). In a similar manner, the subset $\mathcal{P}_o^3$ for layer 3 is defined by the excitatory neurons that preferentially target neurons in $\mathcal{P}_o^4$, and the subset $\mathcal{P}_o^3$ by all the other excitatory neurons in layer 3. This procedure is repeated upstream through the whole network, and also for the other output neurons, defining $\mathcal{P}_o^l$ and $\mathcal{N}_o^l$ for all layers l and all output neurons o. Combining the subsets of neurons from all layers, we obtain a path $\mathcal{P}_o = \cup_{\forall l}\, \mathcal{P}_o^l$ through the network for each output neuron o, as well as a set of neurons not included in the path $\mathcal{N}_o = \cup_{\forall l}\, \mathcal{N}_o^l$. Accordingly, we call the subsets $\mathcal{P}_o^l$ stages of the path $\mathcal{P}_o$. Note that our construction of the paths is based solely on the connection preference of neurons for an output neuron, irrespective of the neural activity. At the end, for each output neuron, a “path” through the network is identified, along which the neurons strongly influence the output neuron.

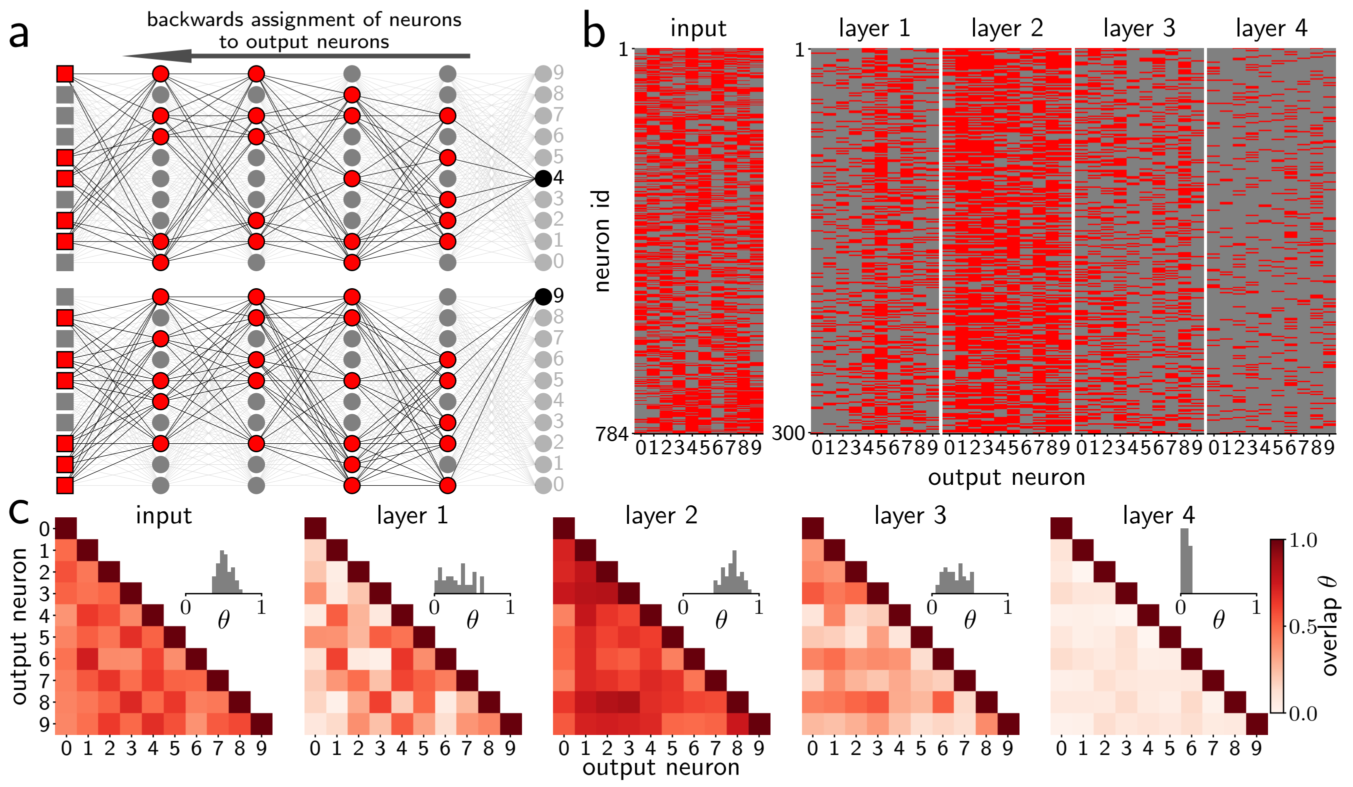

Figure 3: Connectivity structure for the separation of image classes. (**a**) Sketch for two identified paths, in the upper row for output neuron 4 in the lower for output neuron 9. In the upper row, neurons in the path i.e., in $\mathcal{P}_4$ are marked in red and neurons not in the path i.e., in $\mathcal{N}_4$, are marked grey. The neurons in $\mathcal{P}_4$ have stronger connections to the neurons within the path converging on output neuron 4. The path is identified by tracing the connections to output neuron 4 backwards through the network (see Methods: Assignment of neural subsets and paths). The same holds for the lower row for output neuron 9, mutatis mutandis. Each neuron can take part in multiple paths i.e., be part of $\mathcal{P}_4$ and $\mathcal{P}_9$ simultaneously. (**b**) Assignment of neurons to paths denoted by the identity of their respective label neurons on the abscissa. (**c**) Pairwise overlap between stages of the paths (Equation 2). The degree of overlap is displayed in the colorbar. The insets show the distribution of overlap scores between pairs of pathways belonging to different output neurons (i.e., of the off diagonal elements of the presented matrices).

Figure 3b shows the resulting subsets $\mathcal{P}_o^l$ and $\mathcal{N}_o^l$ for all 10 output neurons for all layers, where neurons in $\mathcal{P}_o^l$ are illustrated in red and neurons in $\mathcal{N}_o^l$ in gray. In layer 4 we observe fewer neurons in $\mathcal{P}_o^4$ than $\mathcal{N}_o^4$ for all output neurons, in contrast to, e.g., layer 2 where much more neurons are in $\mathcal{P}_o^2$ than in $\mathcal{N}_o^2$. At first, the paths become denser up until layer 2, where many neurons participate in different paths. Then, from layer 2 to layer 4, the paths become increasingly sparse, such that fewer and fewer neurons contribute to the path to each individual output neuron.

To quantify how these sets of neurons become more specific to a particular output neuron in deeper layers, we calculate their pairwise overlap $\theta_{i,j}^l$ for all combinations of output neurons, akin to the cosine similarity of vectors:

$$\theta_{i,j}^l = \frac{|\mathcal{P}_i^l \cap \mathcal{P}_j^l|}{\sqrt{|\mathcal{P}_i^l||\mathcal{P}_j^l|}} \,. \tag{2}$$

If the intersection of the two sets is empty, then $\theta_{i,j}^l = 0$; if the two sets are identical, then $\theta_{i,j}^l = 1$.

The resulting overlaps are shown in Figure 3c, separately for each layer. First the overlap overall increases up to layer 2 and then clearly drops towards the output layer, indicating that the stages of the paths become increasingly separate from each other. Remarkably, this structure emerges spontaneously through the learning, with the loss function based on the spike times of the output neurons. This indicates that the progressively separated paths would be optimal for routing activity towards a specific destination as fast as possible. Furthermore, this structure would also ensure non-overlapping representations of various input classes towards deeper layers.

Activity propagates along paths

After the separate analysis of activity and structure, we combine the two. We ask to what extent connectivity corresponds to dynamics, i.e., how the identified paths relate to the activity patterns discussed before. As depicted in Figure 4a, images of class 4 should activate the neurons in $\mathcal{P}_4$ and their activity should propagate along this path, and the same should hold for images of class 9 and $\mathcal{P}_9$. This would naturally explain the observed specificity of the leading neurons in their response to images of various classes.

In Figure 4b, we show the spikes of excitatory neurons in the same network activity as shown in Figure 1b, but here the spikes are labeled according to their membership in two different paths – on the left: path $\mathcal{P}_4$ to output neuron 4 (red: $\mathcal{P}_4$, gray: $\mathcal{N}_4$), and on the right: path

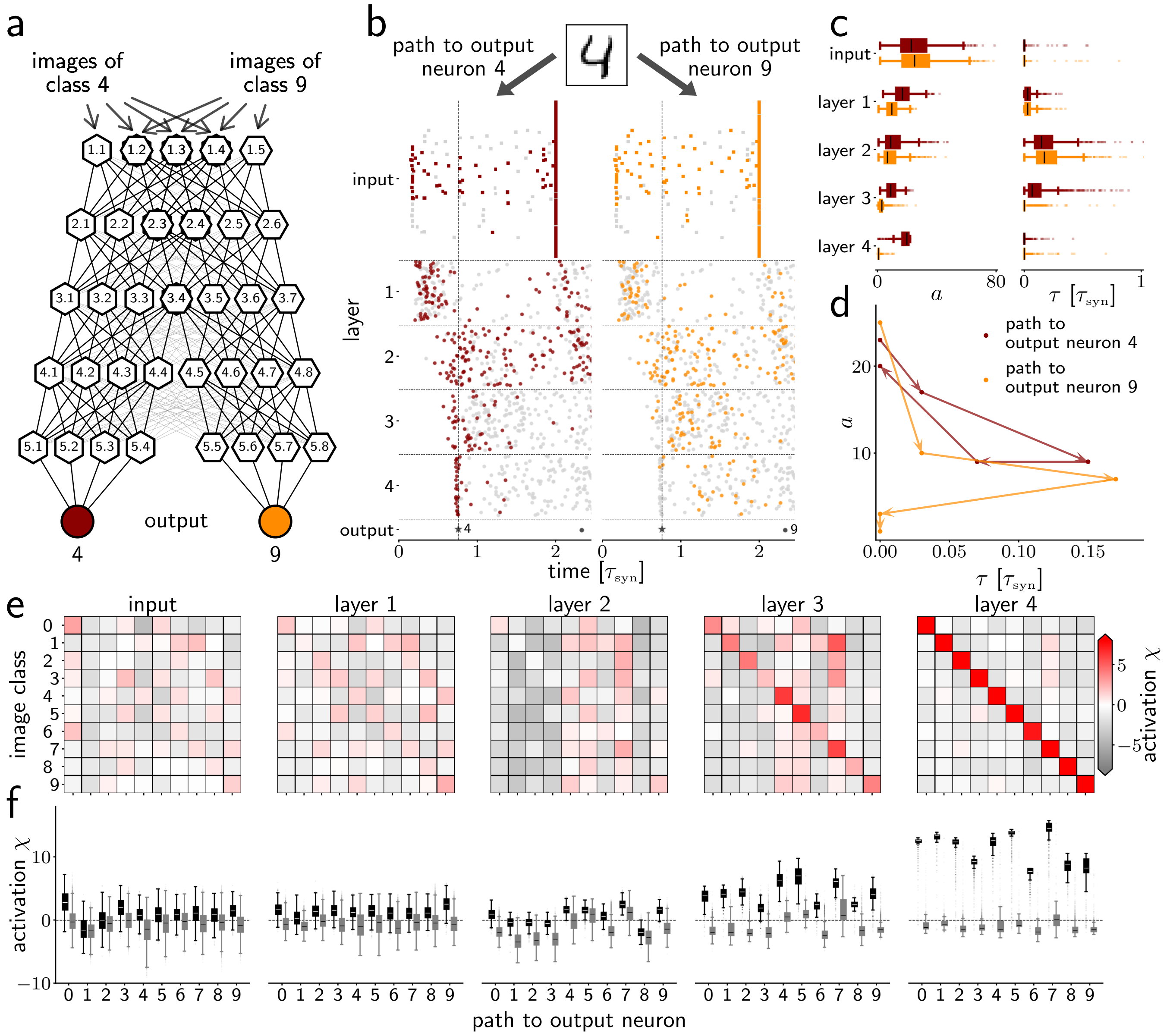

Figure 4: Activity propagation along the identified paths. (**a**) Sketch of the separation of the path to output neuron 4 (⬡) and path to output neuron 9 (⬡) in the deeper layers of the network. (**b**), Two raster plots (as in Figure 1b) of the activity in all layers in response to the same image (class 4) with the neurons labeled according to $\mathcal{P}_4$ (dark red) and $\mathcal{N}_4$ (gray) (left) and $\mathcal{P}_9$ (dark orange) and $\mathcal{N}_9$ (gray) (right). Inhibitory neurons not shown. (**c**) Box plots of the distributions across all images from class 4 of the number of active neurons a (left) and rise time τ (right) for the neurons in $\mathcal{P}_4$ (red) and $\mathcal{P}_9$ (orange) analogous to Figure 1e but this time only for a subset of neurons. (**d**) Corresponding state space representation of the medians of the distributions for the two paths in **d** in sequence of the layers, analogous to Figure 1d. (**e**) Evaluation of the activation of the paths. Each matrix entry is the mean activation of the paths χ (Equation 3) for all images of a given class . The colorbar indicates the mean activation, red indicates a strong activation of $\mathcal{P}_o^l$, gray indicates predominant activation of $\mathcal{N}_o^l$. (**f**) Box plots for the distribution of the activation for all images, split between the activation when the image class and output neuron are the same (black) or different (gray) for each layer. An activation of zero corresponds to chance level (dashed black line).

$\mathcal{P}_9$ to output neuron 9 (orange: $\mathcal{P}_9$, gray: $\mathcal{N}_9$). Note that both panels show the same spiking activity, merely labeled differently with regard to different paths. Specifically, the left panel highlights the spikes through the path to the correct output neuron, and the right panel to an incorrect output neuron. We observe more and earlier spikes for neurons belonging to $\mathcal{P}_4$ compared to those in $\mathcal{P}_9$, most evidently in the deeper layers of the network, before the first spike occurs in the output layer. Furthermore, the spikes of the neurons in $\mathcal{P}_4$ are precisely synchronized in the deeper layers. In contrast, the neurons in $\mathcal{P}_9$ emit only a small amount of asynchronous spikes deeper into the network before the first spike in the output layer.

This relates to our earlier observation, namely, the shaping and propagation of pulse packets through the network (Figure 1c-f). Hence, we employ here again the same quantification of the activity using the rise time τ and the number of leading neurons a, but in this case separately for the neurons in each individual path. The distributions of a and τ across all images of class 4 for the two paths considered above (Figure 4c, $\mathcal{P}_4$ in red and $\mathcal{P}_9$ in orange) show that for both paths the rise time first increases and then decreases, similarly to the previous observation from all excitatory neurons together (Figure 1e, red arrows). However, the two paths behave differently regarding the number of leading neurons. For $\mathcal{P}_4$, the number of active neurons first decreases and then increases, again as observed in Figure 1e (red). In contrast, for $\mathcal{P}_9$, the number of active neurons decreases constantly, which in part explains the decrease of τ in the deeper layers in this path, i.e., there are hardly any leading neurons and hence the rise time is bound to be extremely short. When an image of class 9 is given as the input, we observe the exact inverse (Supplemental Information: Figure S3), with activity along path $\mathcal{P}_9$ being shaped into a compact and stable pulse, and the activity along $\mathcal{P}_4$ gradually fading out. Thus, the present example clearly shows that the activity through the "correct" pathway, i.e., the one corresponding to the correct output neuron, survives and gets strengthened, and the activity in the incorrect path dies out (Figure 4d).

For the quantification of this observation we define the activation $\chi^l_{o,x}$ of the stage $\mathcal{P}^l_o$ in response to image x in layer l as

$$\chi^l_{o,x} = \frac{|\mathcal{V}^l_x \cap \mathcal{P}^l_o| - |\mathcal{V}^l_x \cap \mathcal{N}^l_o| - \mu^l_{o,x}}{\sigma^l_{o,x}} , \tag{3}$$

where we again consider the set of leading neurons $\mathcal{V}^l_x$ as defined earlier. This measure evaluates if the neurons within the path $\mathcal{P}^l_o$ or those outside the path $\mathcal{N}^l_o$ are more activated. $\mu^l_{o,x}$ and $\sigma^l_{o,x}$ are for normalizing $\chi^l_{o,x}$ (for details see Methods: Activation of neural subsets) such that $\chi_{o,x}$ equals zero when the neurons are randomly active independently of their assignments to the path. The larger $\chi^l_{o,x}$ is, the more neurons within the path are activated compared to the neurons outside the path. We obtain, for each image x, in total ten $\chi^l_{o,x}$ per layer, one for each output neuron.

In Figure 4e, we show the mean activation for each of the ten paths across all images of each individual class as a matrix: rows for image classes and columns for output neurons; one matrix per layer. The early layers do not show a concentration of activity on the correct path: the diagonal elements of the matrices up until layer 2 are not distinguishable from the off-diagonal elements. In layer 3 and 4, each path is activated in response almost only to the images of its corresponding class, indicating that in these layers the correct path is selectively activated, with the incorrect paths activated only at a chance level ($\chi \approx 0$). Notably, some paths are in general more active to all images (i.e., the columns for these paths are "more" red) than the others, but deeper in the network the activation is highest for images of the correct class. This is quantitatively confirmed by the distributions of activation (Figure 4f) across all images, shown separately for the correct paths (black) and the incorrect paths (gray). The activation gets increasingly higher for the correct paths deeper in the network, whereas the activation of the incorrect paths stays around zero throughout the layers.

A high activation indicates that the neurons preferentially propagating activity through the path are earlier active than neurons that would propagate activity not through the path. The activation of the individual path needs to be high enough so that the activity further propagates along that path. The deciding factor for the selection of the correct path is not the activation of the path compared to the other paths, but whether the activation of the given path is sufficient to further propagate activity.

Discussion

We analyze the spiking activity and connectivity structure of a deep SNN with distinct excitatory and inhibitory populations, trained to classify visual input. In response to different images, we observe pulse packets i.e., synchronous volleys of spikes, propagating through the network that first broaden and then sharpen again. While these pulse packets propagate downstream through the network, the neurons active within the packet become increasingly indicative of the image class. This in particular holds true for the excitatory neurons, while inhibitory neurons generally respond in a less specific manner. Turning to the network connectivity, starting from individual output neurons we identify different paths each of which corresponds to one image class. Comparing these paths reveals an increasing separation with network depth on a structural level. Connecting the analysis of the spiking activity with the identified paths, we demonstrate that upon presentation of an image the evoked activity propagates along the path to the class of the presented image.

Feed-forward networks supporting the propagation of a pulse packet have been discussed extensively in the context of *synfire chains* (SFCs) (Abeles, 1982; Abeles, 1991; Diesmann et al., 1999; Tetzlaff et al., 2002; Kumar et al., 2008; Trengove et al., 2013). SFCs were suggested as a model for reliable and fast propagation of activity in neural networks. Each of the paths in the network studied here that are activated upon presentation of the various images can be identified as a SFC. Thus, images are classified by triggering activity along the SFC corresponding to the correct class. In this sense, the studied network can be thought of as computing with SFCs.

Given the large number of neurons in the network, it seems plausible that many more SFCs can be embedded than required to classify MNIST. In practice, we expect that the number of paths depends on the one hand on the statistics of the input data, i.e., the inter- and intra-class variability, and on the other hand the capacity limit to embed paths in the network (Bienenstock, 1995). The evoked activity by different images of the same class needs to converge to the same path while activity for images from different classes needs to be distinguishable.

Each image class is represented by a distinct subset of neurons that consistently spike early upon the presentation of an image of a given class. With this, the latency code representing the image is transformed into a binary code of the leading neurons representing the image class. The neurons in the deeper layer are specialized, i.e., clearly representative of a particular class. This is similar to the well-known "grandmother neurons" or concept cells in areas higher in the visual hierarchy (Kobatake and Tanaka, 1994; Quiroga et al., 2005; Rust and DiCarlo, 2010; Quiroga, 2012). The representation of the image class becomes clearer with network depth, denoising and semantizing the input through the propagation of signals along the paths (Kadmon and Sompolinsky, 2016; Zajzon et al., 2023). Thus our network reproduces a prominent characteristic of neurons in the visual hierarchy.

Remarkably, the network was not trained with this mechanism in mind: the loss function is based on the spike times of the output neurons and trained the network with regular error backpropagation (for details see Göltz et al. (2021)). Images were provided in form of spike times with a latency code, an efficient and easy-to-implement code for rapid processing (Thorpe et al., 2001). The described mechanism automatically emerged through the training. We view this as a direct consequence of the interplay between the spike latency coding in the input, the loss function that enforces competition between the output neurons for who spikes first, and the learning algorithm which ultimately moves spike times to produce the desired outcome.

The network analyzed in this study forms a structure that enables the fast propagation through multiple layers of a network. Visual processing in the brain shares a similar property: it is well known that in the human brain the visual processing from image presentation to recognition is very fast ($\sim 150\,\mathrm{ms}$) (Thorpe et al., 1996; Hung et al., 2005). Additionally, simple object recognition often relies on the first feed-forward sweep of activity (Lamme and Roelfsema, 2000; Roelfsema, 2023), and information is transmitted by the first spikes in response to a stimulus (Johansson and Birznieks, 2004). The processing in our network relies also only on the feed-forward sweep of activity. This is sufficient for classifying MNIST. For more complex object recognition (Kar et al., 2019) or other cortical processes, like attention (Lamme and Roelfsema, 2000; Supèr et al., 2001) or learning (Hinton et al., 1995) recurrent connections are suggested to be required. The influence of recurrent connections on the here studied mechanism needs to be studied in future work.

By including excitatory and inhibitory neurons, we recover another property observed in cortical networks: excitatory neurons are more sharply tuned to a specific stimulus, while the inhibitory neurons are less specific (Sohya et al., 2007; Niell and Stryker, 2008; Lundqvist et al., 2010). Inhibitory neurons are employed during the propagation of the activity, but they do not carry the main information about the image class. Rather, they regulate the network by providing unspecific inhibitory input to the excitatory neurons in the next layer, akin to the *blanket of inhibition*, i.e. the dense and unspecific innervation of excitatory neurons by inhibitory neurons, found in cortical circuits (Fino and Yuste, 2011; Karnani et al., 2014). Similarly, inhibition has been found to restrict the spatial spread and temporal persistence of neural activity in visual cortex (Haider et al., 2013). Additionally, inhibition could facilitate the synchronization of the pulse packets, as had been reported in a previous study (Shinozaki et al., 2010). In our network the inhibitory neurons develop a similar facilitating role though the training.

We note that the size of the employed network consists of a much larger number of neurons and layers than necessary to classify MNIST. In the original implementation, Göltz et al. (2021) showed that the task can already be solved by a network with only one hidden layer. Since in this work we aimed at investigating the relation between signal propagation and computation, we chose a network that contained more layers. We expect the result to be transferable to more complex visual tasks, since MNIST does not contain any structure that inherently enforces the observations we report here.

Recently the theoretical analysis of the dynamics of learning capabilities in artificial neuronal networks has gained attention (Schoenholz et al., 2017; Fischer et al., 2023; van Meegen and Sompolinsky, 2025). The approaches in these studies allow for a statistical assessment across different networks. In contrast, here we focused the analysis on one concrete realization of an SNN. This complementary approach enables a dissection of the relationship between structure and function on a more fine-grained level, doing justice to the individuality of each trained network. However, results for other realizations are qualitatively similar (Supplemental Information: Figure S4). Focusing on a specific network acknowledges the fact that natural neural networks are not the averages of a distribution, but a concrete instance that grow and adjust to fulfill a specific function. With our *idiographic* (Windelband, 1998) approach we provide insight into SNNs, even if we base the analysis on only few examples. In this way, our approach is similar to the analysis of neuroscientific experiments, where one also has access to only a few subjects (Fries and Maris, 2022).

Future work could address how different spike timing codes, imposed by construction, would shape the learned activity in the network. Similarly, the impact of different learning rules could be analyzed. In this way, the universality of the identified shared properties between the visual system and the networks studied here can be investigated. Additionally, a thorough analysis of SNNs may help to also improve their performance (Dold and Petersen, 2025).

Expanding the approach of our analysis to more complex networks and more complex visual tasks will strengthen the connection between functional neural networks and fundamental concepts in neuroscience. This includes whether trained SNNs form receptive fields, or if through the training binding emerges (Singer and Gray, 1995). Moreover, future analyses could also address networks with recurrent connections and ongoing activity. Thus, we view this work explicitly as a starting point for further studies of how structures inside the brain are capable of learning efficient spike-based codes. While our comparatively simple networks already allow the formulation of clear and rigorous links between gradient descent on spike times and observations in cortex, further extensions of our model will provide additional insight into the computational role of the various components – in structure and dynamics – observed in the brain.

Methods

Network setup

The investigated networks are multi-layer, feed-forward, all-to-all connected networks of spiking neurons. Here, we elaborate on the setup of the experiments (for details see Göltz et al., 2021): The neurons are leaky integrate-and-fire neurons with exponential synapses and a long refractory time constant to ensure single spikes per neuron. Following an input sample, the spiking activity of the neurons is given by a differentiable function, and its derivatives are used to optimize the parameters in the network with gradient descent (Equations 2, 4, 5 in Göltz et al., 2021) in a mini-batch training setup. The precise parameters of training as well as the training code are given alongside the trained network, see Code and data availability.

In a change from the referenced manuscript, here we respect Dale's law and separate the neurons into an excitatory and an inhibitory population in the hidden layers. We ensure the desired effect by clipping the outgoing weights to positive and negative values, respectively.

Rise time

The rise time τ_x is measured on the basis of the population spike time histogram in each layer individually for the activity in response to image x. For the calculations in this paper, we calculate the spike time histogram with a sliding bin size of $0.05\,\tau_\mathrm{syn}$ with non-exclusive binning. The rise time is defined as the center of the first bin that corresponds to a maximum after the first spike in the layer.

Set of leading neurons

We define a set of leading neurons $\mathcal{V}_x^l$ for image x in layer l on the basis of the rise time τ_x.

$$\mathcal{V}_x^l = \{i | t_{i,x} <= \min_i(t_{i,x}) + \tau_x\} \,, \tag{4}$$

with the spike time $t_{i,x}$ of each neuron i.

Similarity of sets of leading neurons

To measure similarity between the sets, we define a measure on the basis of the Pearson product-moment correlation coefficient. For that, we need to define a mean μ, a variance and covariance in the context of our sets. To this end we interpret the neurons of in a layer as a binary vector with N^l elements $v_{x,i}^l$ for which $v_{x,i}^l = 1\,\forall\, i \in \mathcal{V}_x^l$ and $v_{x,i}^l = 0\,\forall\, i \notin \mathcal{V}_x^l$. In this framework we use the mean over the entries of this vector.:

$$\mu_x^l = \frac{1}{N^l}\sum_{i=1}^{N^l} v_{x,i}^l = \frac{|\mathcal{V}_x^l|}{N^l} \,. \tag{5}$$

Accordingly, for the variance we have:

$$\begin{aligned}\mathrm{Var}_x^l &= \frac{1}{N^l}\sum_{i=1}^{N^l}(v_{x,i}^l - \mu_x^l)^2 && (6)\\ &= \frac{1}{N^l}\sum_{i=1}^{N^l}\left((v_{x,i}^l)^2 - 2v_{x,i}^l\mu_x^l + (\mu_x^l)^2\right) && (7)\\ &= \frac{1}{N^l}\left(\sum_{i=1}^{N^l}(v_{x,i}^l)^2\right) - (\mu_x^l)^2 && (8)\\ &= \frac{|\mathcal{V}_x^l|}{N^l} - \left(\frac{|\mathcal{V}_x^l|}{N^l}\right)^2 \,, && (9)\end{aligned}$$

and the covariance:

$$\begin{aligned}\mathrm{Cov}_{x,y}^l &= \frac{1}{N^l}\sum_{i=1}^{N^l}(v_{x,i}^l - \mu_x^l)(v_{y,i}^l - \mu_y^l) && (10)\\ &= \frac{1}{N^l}\left(\sum_{i=1}^{N^l} v_{x,i}^l v_{y,i}^l\right) - \mu_x^l\mu_y^l && (11)\\ &= \frac{|\mathcal{V}_x^l \cap \mathcal{V}_y^l|}{N^l} - \frac{|\mathcal{V}_x^l||\mathcal{V}_y^l|}{(N^l)^2} \,. && (12)\end{aligned}$$

From this we obtain the similarity between sets as defined in Equation 1:

$$\begin{aligned}\rho_{x,y}^l &= \frac{\mathrm{Cov}_{x,y}^l}{\sqrt{\mathrm{Var}_x^l\mathrm{Var}_y^l}} && (13)\\ &= \frac{N^l|\mathcal{V}_x^l \cap \mathcal{V}_y^l| - |\mathcal{V}_x^l||\mathcal{V}_y^l|}{\sqrt{|\mathcal{V}_x^l||\mathcal{V}_y^l|(N^l - |\mathcal{V}_x^l|)(N^l - |\mathcal{V}_y^l|)}} \,. && (14)\end{aligned}$$

Information gain

We first measure the entropy H of the distribution of image classes X. we then measure the neuron-conditional entropy $H(X|i)$, i.e., the entropy of the posterior distribution of X given that neuron i was active. The information gain IG_i for neuron i is the normalized difference between these two entropies:

$$\mathrm{IG}_i = \frac{H(X) - H(X|i)}{H(X)}. \tag{15}$$

Assignment of neural subsets and paths

For identifying the paths, we start with the neurons in the last hidden layer, since they are directly responsible for the classification by the output neurons. Let's consider output neuron o. To evaluate which neurons in layer 4 preferentially target this neuron, we compare the connection weight $w^4_{o,i}$ of each neuron i to the output neuron o with the average weight of given neuron to all output neurons: $\bar{w}^4_i = \frac{1}{10}\sum_{j=0}^{9} w^4_{j,i}$ with respect to the standard deviation $\sigma^4_{w_i}$ of these weights. All neurons in layer 4 that fulfill $w^4_{o,i} > \bar{w}^4_i + \sigma^4_{w_i}$ are assigned to the set $\mathcal{P}^4_o$ of neurons in layer 4 with strong impact on output neuron o. The neurons that do not fulfill this condition are in set $\mathcal{N}^4_o$, resulting in:

$$\mathcal{P}^4_o = \{i | w^4_{o,i} > \bar{w}^4_i + \sigma^4_{w_i}\} \tag{16}$$
$$\mathcal{N}^4_o = \{i | w^4_{o,i} \leq \bar{w}^4_i + \sigma^4_{w_i}\}, \tag{17}$$

with $\sigma^4_{w_i} = \sqrt{\frac{1}{10}\sum_{j=0}^{9}(w^4_{j,i} - \bar{w}^4_i)^2}$.

Then in the penultimate layer (layer 3), we calculate for each neuron the average connection weight to neurons in $\mathcal{P}^4_o$ and $\mathcal{N}^4_o$, respectively:

$$\bar{w}^3_{i,p,o} = \frac{1}{|\mathcal{P}^l_o|}\sum_{j\in\mathcal{P}^4_o} w^3_{j,i} \text{ and} \tag{18}$$
$$\bar{w}^3_{i,n,o} = \frac{1}{|\mathcal{N}^l_o|}\sum_{j\in\mathcal{N}^4_o} w^3_{j,i}. \tag{19}$$

On this basis we again assign neurons to two sets:

$$\mathcal{P}^3_o = \{i | \bar{w}^3_{i,p,o} > \bar{w}^3_{i,n,o}\} \tag{20}$$
$$\mathcal{N}^3_o = \{i | \bar{w}^3_{i,p,o} < \bar{w}^3_{i,n,o}\}. \tag{21}$$

This procedure is repeated backwards through the whole network, until all excitatory neurons in the hidden layers and the neurons in the input layer are assigned.

Activation of neural subsets

The activation $\chi^l_{o,x}$ as defined in Equation 3 is normalized with respect to random activity of the neurons. It is used to evaluate whether $|\mathcal{V}^l_x \cap \mathcal{P}^l_o|$ or $|\mathcal{V}^l_x \cap \mathcal{N}^l_o|$ is larger. For this it takes into account the expected value $\mu_{o,i}$ and the standard deviation $\sigma_{o,i}$, if the leading neurons would be drawn randomly from $\mathcal{P}^l_o$ or $\mathcal{N}^l_o$ respectively.

$$\mu_{o,i} = |\mathcal{V}^l_i|\left(2\frac{|\mathcal{P}^l_o|}{N^l} - 1\right) \tag{22}$$
$$\sigma_{o,i} = \sqrt{4\frac{|\mathcal{P}^l_o|}{N^l}\left(1 - \frac{|\mathcal{P}^l_o|}{N}\right)|\mathcal{V}^l_{o,i}|\frac{N - |\mathcal{V}^l_i|}{N-1}} \tag{23}$$

The probability of drawing a neuron from $\mathcal{P}^l_o$ is $\frac{|\mathcal{P}^l_o|}{N^l}$. Accordingly, the probability for drawing a neuron from $\mathcal{N}^l_o$ is $\frac{|\mathcal{N}^l_o|}{N^l} = 1 - \frac{|\mathcal{P}^l_o|}{N^l}$. We draw $|\mathcal{V}^l_i|$ neurons without replacement from the layer. This corresponds to a hypergeometric distribution, thus follows $\mu_{o,i}$ as mean and $\sigma_{o,i}$ as standard deviation if the neurons were drawn randomly.

Code and data availability

Code for the network simulations is available at `https://github.com/JulianGoeltz/fastAndDeep`. Code for the analysis will be made available as of the date of publication. Any additional information required to recreate the results reported in this paper is available from the lead contact upon request.

Author contributions

Conceptualization, JOF, JI, MAP, SG; Methodology, JG, LK, MAP; Software, JOF, ACK, JG, LK; Formal Analysis JOF, ACK, JI, SG; Investigation, JG, LK, MAP; Visualization, JOF, ACK; Writing – Original Draft, JOF, ACK, JG, LK, JI, MAP, SG; Writing – Review & Editing, JOF, ACK, JG, LK, JI, MAP, SG; Funding Acquisition, MAP, SG; Resources, MAP, SG; Project Administration, JOF, SG; Supervision, JI, MAP, SG.

Acknowledgements

We thank Markus Diesmann, Günther Palm and Shigeru Shinomoto for intense and fruitful discussions. This research was funded by the European Union's Horizon 2020 Framework programme for Research and Innovation under Specific Grant Agreements No. 945539 (HBP SGA3) and No. 101147319 (EBRAINS 2.0 Project), the NRW-network 'iBehave'(NW21-049) and the Helmholtz Joint Lab SMHB. This work was performed as part of the Helmholtz School for Data Science in Life, Earth and Energy (HDS-LEE). We further wish express our gratitude to the Manfred Stärk Foundation for their continuing support of the NeuroTMA Lab.

References

Abeles, M. (1982). *Local Cortical Circuits: An Electrophysiological Study.* Studies of Brain Function. Berlin, Heidelberg, New York: Springer-Verlag.

— (1991). *Corticonics: Neural Circuits of the Cerebral Cortex.* 1st. Cambridge: Cambridge University Press.

Barral, J., X.-J. Wang, and A. D. Reyes (2019). “Propagation of temporal and rate signals in cultured multilayer networks”. *Nature Communications* 10.1. Publisher: Nature Publishing Group, p. 3969.

Bienenstock, E. (1995). “A model of neocortex”. *Network: Computation in Neural Systems* 6.2, pp. 179–224.

Bohte, S. M., J. N. Kok, and H. La Poutré (2002). “Error-backpropagation in temporally encoded networks of spiking neurons”. *Neurocomputing* 48.1, pp. 17–37.

Churchland, M. M. et al. (2012). “Neural population dynamics during reaching”. *Nature* 487.7405, p. 51.

Diesmann, M., M.-O. Gewaltig, and A. Aertsen (1999). “Stable propagation of synchronous spiking in cortical neural networks”. *Nature* 402.6761, pp. 529–533.

Dold, D. and P. C. Petersen (2025). “Causal pieces: analysing and improving spiking neural networks piece by piece”. *ArXiv.*

Eccles, J. C. (1957). *The physiology of nerve cells.* Baltimore: Johns Hopkins Press.

Fino, E. and R. Yuste (2011). “Dense inhibitory connectivity in neocortex”. *Neuron* 69.6, pp. 1188–1203.

Fischer, K., D. Dahmen, and M. Helias (2023). “Field theory for optimal signal propagation in ResNets”. *ArXiv.*

Fries, P. and E. Maris (2022). “What to Do If N Is Two?” *Journal of Cognitive Neuroscience* 34.7, pp. 1114–1118.

Gallego, J. A. et al. (2017). “Neural manifolds for the control of movement”. *Neuron* 94.5, pp. 978–984.

Gao, P. et al. (2017). “A theory of multineuronal dimensionality, dynamics and measurement”. *BioRxiv*, p. 214262.

Georgopoulos, A., A. Schwartz, and R. Kettner (1986). “Neuronal population coding of movement direction.” *Science* 4771.233, pp. 1416–1419.

Georgopoulos, A. et al. (1982). “On the relations between the direction of two-dimensional arm movements and cell discharge in primate motor cortex.” *Journal of Neuroscience* 11.2, pp. 1527–1537.

Göltz, J. et al. (2021). “Fast and energy-efficient neuromorphic deep learning with first-spike times”. *Nature Machine Intelligence* 3 (9), pp. 823–835.

Gray, C. M. and W. Singer (1989). “Stimulus-specific neuronal oscillations in orientation columns of cat visual cortex”. *Proceedings of the National Academy of Sciences of the United States of America* 86, pp. 1698–1702.

Gray, C. M. et al. (1989). “Oscillatory responses in cat visual cortex exhibit inter-columnar synchronization which reflects global stimulus properties”. *Nature* 338.6213. Publisher: Nature Publishing Group, pp. 334–337.

Haider, B., M. Häusser, and M. Carandini (2013). “Inhibition dominates sensory responses in the awake cortex”. *Nature* 493, pp. 97–100.

Hinton, G. E. et al. (1995). “The ”Wake-Sleep” Algorithm for Unsupervised Neural Networks”. *Science* 268.5214, pp. 1158–1161.

Hubel, D. H. and T. N. Wiesel (1962). “Receptive Fields, Binocular Interaction, and Functional Architecture in the Cat’s Visual Cortex”. *Journal of Physiology* 160, pp. 106–154.

Hung, C. P. et al. (2005). “Fast Readout of Object Identity from Macaque Inferior Temporal Cortex”. *Science* 310.5749, pp. 863–866.

Izhikevich, E. M. (2006). “Polychronization: computation with spikes”. *Neural Computation* 18.2, pp. 245–282.

Johansson, R. and I. Birznieks (2004). “First spikes in ensembles of human tactile afferents code complex spatial fingertip events.” *Nature Neuroscience* 2.7, pp. 170–177.

Kadmon, J. and H. Sompolinsky (2016). “Optimal Architectures in a Solvable Model of Deep Networks”. In: *Advances in Neural Information Processing Systems 29.* Ed. by D. D. Lee et al. Curran Associates, Inc., pp. 4781–4789.

Kar, K. et al. (2019). “Evidence that recurrent circuits are critical to the ventral stream’s execution of core object recognition behavior”. *Nature Neuroscience* 22.6, pp. 974–983.

Karnani, M. M., M. Agetsuma, and R. Yuste (2014). “A blanket of inhibition: functional inferences from dense inhibitory connectivity”. *Current Opinion in Neurobiology* 26, pp. 96–102.

Kilavik, B. E. et al. (2009). “Long-term modifications in motor cortical dynamics induced by intensive practice”. *Journal of Neuroscience* 29.40, pp. 12653–12663.

Kobatake, E. and K. Tanaka (1994). “Neuronal selectivities to complex object features in the ventral visual pathway of the macaque cerebral cortex”. *Journal of Neurophysiology* 71.3, pp. 856–867.

Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012). “Imagenet classification with deep convolutional neural networks”. In: *Advances in Neural Information Processing Systems.* Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., pp. 1097–1105.

Kumar, A. et al. (2008). “The High-Conductance State of Cortical Networks”. *Neural Computation* 20.1, pp. 1–43.

Lamme, V. A. and P. R. Roelfsema (2000). “The distinct modes of vision offered by feedforward and recurrent processing”. *Trends in Neurosciences* 23, pp. 571–579.

LeCun, Y et al. (1998). “Gradient-based learning applied to document recognition”. *Procedings of the IEEE* 86.11, pp. 2278–2324.

Levina, A., V. Priesemann, and J. Zierenberg (2022). “Tackling the subsampling problem to infer collective properties from limited data”. *Nature Reviews Physics* 4.12. Publisher: Nature Publishing Group, pp. 770–784.

Lindsay, G. W. (2021). “Convolutional Neural Networks as a Model of the Visual System: Past, Present, and Future”. *Journal of Cognitive Neuroscience* 33.10, pp. 2017–2031.

Lundqvist, M., A. Compte, and A. Lansner (2010). “Bistable, Irregular Firing and Population Oscillations in a Modular Attractor Memory Network”. *PLOS Computational Biology* 6.6, e1000803.

Markram, H. et al. (2004). "Interneurons of the neocortical inhibitory system". *Nature Reviews Neuroscience* 5.10, pp. 793–807.

Morales-Gregorio, A. et al. (2024). "Neural manifolds in V1 change with top-down signals from V4 targeting the foveal region". *Cell Reports* 43.7, p. 114371.

Neftci, E. O. et al. (2017). "Event-driven random backpropagation: Enabling neuromorphic deep learning machines". *Frontiers in Neuroscience* 11, p. 324.

Niell, C. M. and M. P. Stryker (2008). "Highly Selective Receptive Fields in Mouse Visual Cortex". *Journal of Neuroscience* 28.30, pp. 7520–7536.

Perkel, D. H. and T. H. Bullock (1968). "Neural coding". *Neurosciences Research Program Bulletin* 6.3, pp. 221–348.

Prut, Y. et al. (1998). "Spatiotemporal structure of cortical activity: properties and behavioral relevance". *Journal of Neurophysiology* 79.6, pp. 2857–2874.

Quiroga, R. Q. et al. (2005). "Invariant visual representation by single neurons in the human brain". *Nature* 435.7045. Publisher: Nature Publishing Group, pp. 1102–1107.

Quiroga, R. Q. (2012). "Concept cells: the building blocks of declarative memory functions". *Nature Reviews Neuroscience* 13.8. Publisher: Nature Publishing Group, pp. 587–597.

Reyes, A. D. (2003). "Synchrony-dependent propagation of firing rate in iteratively constructed networks in vitro". *Nature Neuroscience* 6.6, pp. 593–599.

Richards, B. A. et al. (2019). "A deep learning framework for neuroscience". *Nature Neuroscience* 22.11. Publisher: Nature Publishing Group, pp. 1761–1770.

Riehle, A. et al. (1997). "Spike synchronization and rate modulation differentially involved in motor cortical function". *Science* 278.5345, pp. 1950–1953.

Roelfsema, P. R. (2023). "Solving the binding problem: Assemblies form when neurons enhance their firing rate—they don't need to oscillate or synchronize". *Neuron* 111.7, pp. 1003–1019.

Rosenblatt, F (1958). "The perceptron: a probabilistic model for information storage and organization in the brain". *Psychol Rev* 65, pp. 386–408.

Rust, N. C. and J. J. DiCarlo (2010). "Selectivity and Tolerance ("Invariance") Both Increase as Visual Information Propagates from Cortical Area V4 to IT". *Journal of Neuroscience* 30.39. Publisher: Society for Neuroscience Section: Articles, pp. 12978–12995.

Schoenholz, S. S. et al. (2017). "Deep information propagation". *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings.*

Shinozaki, T. et al. (2010). "Flexible traffic control of the synfire-mode transmission by inhibitory modulation: Nonlinear noise reduction". *Physical Review E* 81.1, p. 011913.

Singer, W. and C. Gray (1995). "Visual feature integration and the temporal correlation hypothesis." *Annual Review of Neuroscience* 18, pp. 555–586.

Sohya, K. et al. (2007). "GABAergic Neurons Are Less Selective to Stimulus Orientation than Excitatory Neurons in Layer II/III of Visual Cortex, as Revealed by In Vivo Functional Ca2+ Imaging in Transgenic Mice". *Journal of Neuroscience* 27.8, pp. 2145–2149.

Sotomayor-Gómez, B., F. P. Battaglia, and M. Vinck (2025). "Firing rates in visual cortex show representational drift, while temporal spike sequences remain stable". *Cell Reports* 44.4, p. 115547.

Stringer, C. et al. (2019). "High-dimensional geometry of population responses in visual cortex". *Nature* 571.7765, pp. 361–365.

Supèr, H., H. Spekreijse, and V. A. F. Lamme (2001). "Two distinct modes of sensory processing observed in monkey primary visual cortex (V1)". *Nature Neuroscience* 4.3. Publisher: Nature Publishing Group, pp. 304–310.

Tetzlaff, T., T. Geisel, and M. Diesmann (2002). "The ground state of cortical feed-forward networks". *Neurocomputing* 44–46, pp. 673–678.

Thorpe, S., A. Delorme, and R. Van Rullen (2001). "Spike-based strategies for rapid processing". *Neural Networks* 14.6, pp. 715–725.

Thorpe, S., D. Fize, and C. Marlot (1996). "Speed of processing in the human visual system". *Nature* 381, pp. 520–522.

Torre, E. et al. (2016). "Synchronous spike patterns in macaque motor cortex during an instructed-delay reach-to-grasp task". *Journal of Neuroscience* 36.32, pp. 8329–8340.

Trengove, C., C. van Leeuwen, and M. Diesmann (2013). "High-capacity embedding of synfire chains in a cortical network model". *Journal of Computational Neuroscience* 34.2, pp. 185–209.

van Meegen, A. and H. Sompolinsky (2025). "Coding schemes in neural networks learning classification tasks". *Nature Communications* 16.1. Publisher: Nature Publishing Group, p. 3354.

van Rossum, M. C. W., G. G. Turrigiano, and S. B. Nelson (2002). "Fast Propagation of Firing Rates through Layered Networks of Noisy Neurons". *Journal of Neuroscience* 22.5, pp. 1956–1966.

Vogels, T. P. and L. F. Abbott (2005). "Signal propagation and logic gating in networks of integrate-and-fire neurons". *Journal of Neuroscience* 25.46, pp. 10786–10795.

Windelband, W. (1998). "History and Natural Science". *Theory & Psychology* 8.1, pp. 5–22.

Wunderlich, T. C. and C. Pehle (2021). "Event-based backpropagation can compute exact gradients for spiking neural networks". *Scientific Reports* 11.1. Publisher: Nature Publishing Group, p. 12829.

Xie, W. et al. (2024). "Neuronal sequences in population bursts encode information in human cortex". *Nature* 635.8040. Publisher: Nature Publishing Group, pp. 935–942.

Yamins, D. L. K. and J. J. DiCarlo (2016). "Using goal-driven deep learning models to understand sensory cortex". *Nature Neuroscience* 19.3. Publisher: Nature Publishing Group, pp. 356–365.

Yiling, Y. et al. (2023). "Robust encoding of natural stimuli by neuronal response sequences in monkey visual cortex". *Nature Communications* 14.1. Publisher: Nature Publishing Group, p. 3021.

Yin, B., F. Corradi, and S. M. Bohté (2023). "Accurate online training of dynamical spiking neural networks through Forward Propagation Through Time". *Nature Machine Intelligence* 5.5. Publisher: Nature Publishing Group, pp. 518–527.

Zajzon, B. et al. (2023). "Signal denoising through topographic modularity of neural circuits". *eLife* 12, e77009.

Zenke, F. and S. Ganguli (2018). "SuperSpike: Supervised Learning in Multilayer Spiking Neural Networks". *Neural Computation* 30.6, 1514–1541.

Supplemental Information

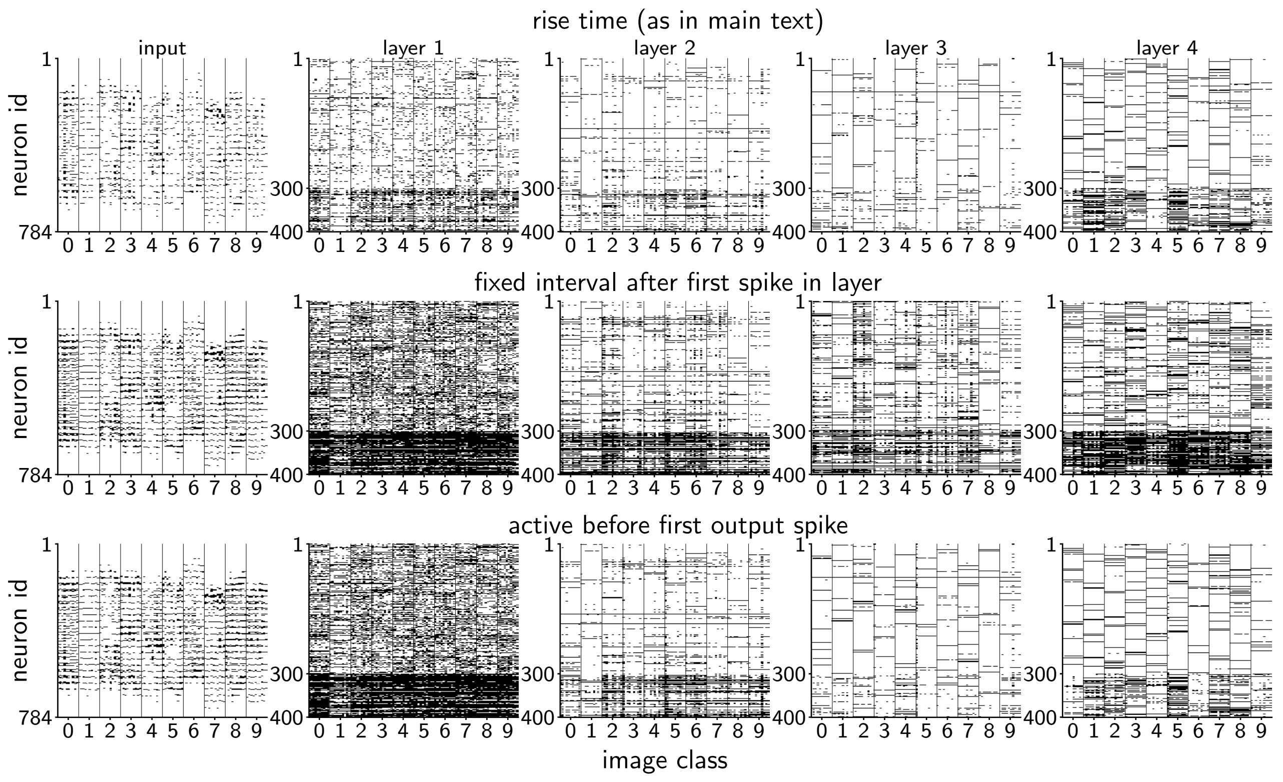

Figure S1: Robustness for the definition of the sets of leading neurons. We can define the set of leading neurons in different ways: in the first row we show result on the basis the definition as in the main text, in the middle row the set is defined as all neurons that spike in the first 0.5 $\tau_{\rm syn}$ after the first spike in the layer and in the last row the set is defined as all neurons that spike before the first output spike.

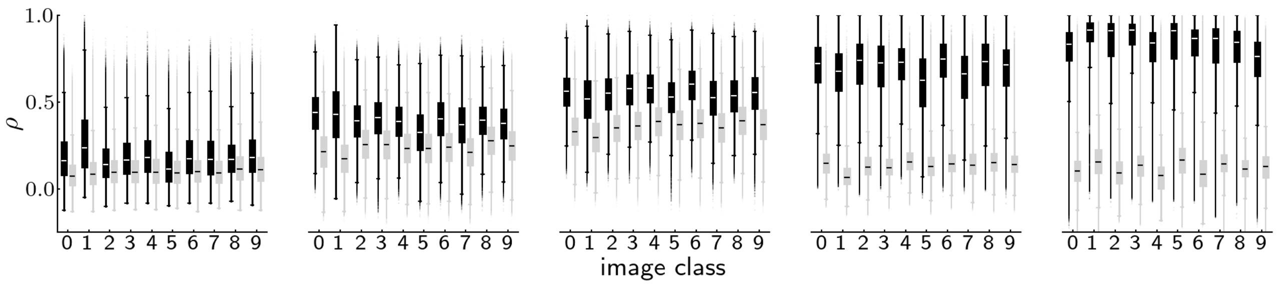

Figure S2: Distribution of similarities. Here we show the distribution of similarities across all images, split between image pairs of the same class (black) and different classes (grey).

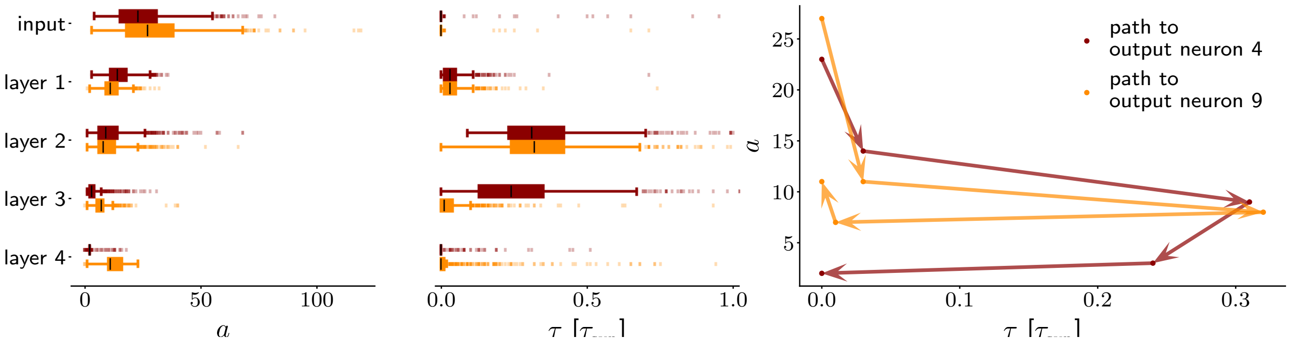

Figure S3: Robustness of the result for images from another class. The results in this figure correspond to the result shown in Figure 4c and Figure 4d, but for all images from class 9.

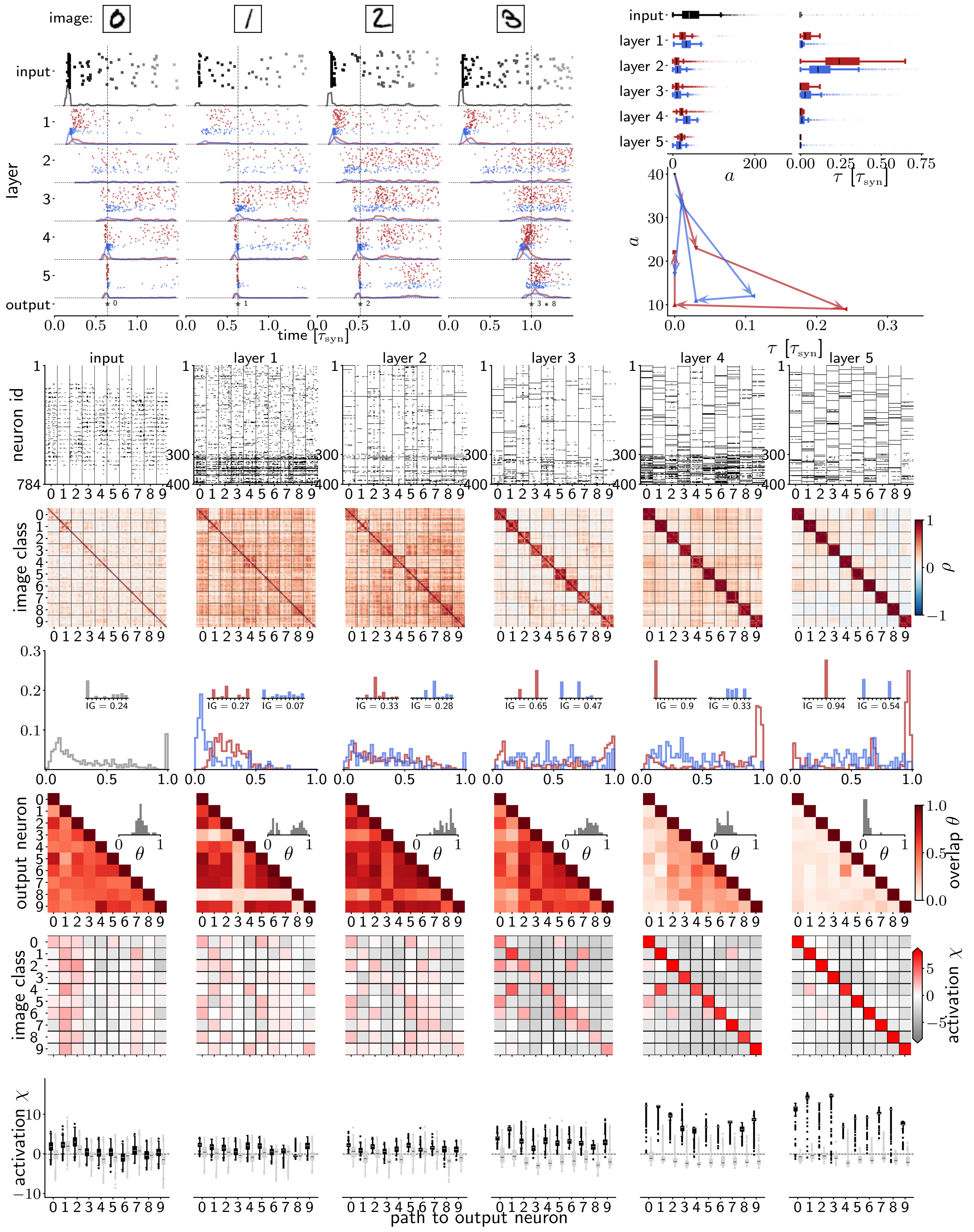

Figure S4: Summary figure for all results obtained from network initialized with another seed and 5 hidden layers in total. The results are analogous to the corresponding figures in the main document.

7. DelGrad: exact event-based gradients for training delays and weights on spiking neuromorphic hardware

This chapter concerns the manuscript published as

> **Julian Göltz***, Jimmy Weber*, Laura Kriener*, Sebastian Billaudelle, Peter Lake, Johannes Schemmel, Melika Payvand and Mihai A. Petrovici (Sept. 2025). 'DelGrad: Exact Event-Based Gradients for Training Delays and Weights on Spiking Neuromorphic Hardware'. In: *Nature Communications* 16.1. DOI: `10.1038/s41467-025-63120-y`.

Author contributions

The *Author Contributions Statement* from the manuscript reads

> JG, JW, and LK jointly developed the theory, designed the experiments, implemented the code, executed simulation and hardware experiments; SB, PL, and JS contributed to the hardware experiments; MP and MAP supervised the project, contributed to the experiment design, and provided helpful guidance throughout; JG, JW, LK, MP, and MAP wrote and revised the manuscript.

This paper is the result of a collaboration between JG, JW, and LK, and as such all three share the first authorship. In addition to driving the theory extension, experiment conceptualisation, and code design, JG performed all hardware experiments in the main manuscript. Moreover, JG, JW and LK took on most of the writing and divided the revision work equally.

An initial version of the hardware experiments, with delays implemented off-chip, was the topic of a Master's thesis (Lake 2025) supervised by the author.

Manuscript

In accordance with *Springer Nature* policies,[20] in the following the manuscript is reprinted.

[20] `https://www.nature.com/nature-portfolio/reprints-and-permissions/permissions-requests`

nature communications

Article

https://doi.org/10.1038/s41467-025-63120-y

DelGrad: exact event-based gradients for training delays and weights on spiking neuromorphic hardware

Received: 24 December 2024

Accepted: 25 July 2025

Published online: 09 September 2025

Check for updates

Julian Göltz[1,2,4], Jimmy Weber[3,4], Laura Kriener[2,3,4], Sebastian Billaudelle[1,3], Peter Lake[1], Johannes Schemmel[1], Melika Payvand[3,5] & Mihai A. Petrovici[2,5]

Spiking neural networks (SNNs) inherently rely on the timing of signals for representing and processing information. Augmenting SNNs with trainable transmission delays, alongside synaptic weights, has recently shown to increase their accuracy and parameter efficiency. However, existing training methods to optimize such networks rely on discrete time, approximate gradients, and full access to internal variables such as membrane potentials. This limits their precision, efficiency, and suitability for neuromorphic hardware due to increased memory and I/O-bandwidth demands. Here, we propose DelGrad, an analytical, event-based training method to compute exact loss gradients for both weights and delays. Grounded purely in spike timing, DelGrad eliminates the need to track any other variables to optimize SNNs. We showcase this key advantage by implementing DelGrad on the BrainScaleS-2 mixed-signal neuromorphic platform. For the first time, we experimentally demonstrate the parameter efficiency, accuracy benefits, and stabilizing effect of adding delays to SNNs on noisy hardware. DelGrad thus provides a new way for training SNNs with delays on neuromorphic substrates, with substantial improvements over previous results.

The mammalian brain has always represented the ultimate example of computational prowess, and therefore remains an important source of inspiration for understanding intelligence and replicating it in artificial substrates. In particular, its specific mechanisms for transmitting and processing information have been the subject of intense scrutiny and debate. Among these, the pulsed communication between neurons, predominantly based on all-or-none events, called action potentials or spikes, stands out as a distinguishing feature, and has thus been suggested to play an important role in the brain's remarkable combination of computational performance and energy efficiency[1,2]. Consequently, spike-based communication represents a de facto standard across current neuromorphic platforms, which aim to inherit the proficiency of their biological archetype by replicating chosen aspects of its structure and dynamics[3–7].

Among the various encoding schemes proposed for spiking neurons, the representation of information within the specific timing of individual spikes is of particular interest[8], as it effectively allows the communication of, ideally, real-valued signals on an energy budget equivalent to only generating and transmitting a single bit. This gives rise to a specific call for SNN training algorithms that exploit the temporal richness of spike timing codes for solving computational tasks efficiently and accurately, while remaining capable of operating under the realistic constraints of the underlying physical substrate, whether biological or artificial.

[1]Kirchhoff-Institute for Physics, Heidelberg University, Heidelberg, Germany. [2]Department of Physiology, University of Bern, Bern, Switzerland. [3]Institute of Neuroinformatics, University of Zurich and ETH Zurich, Zürich, Switzerland. [4]These authors contributed equally: Julian Göltz, Jimmy Weber, Laura Kriener. [5]These authors jointly supervised this work: Melika Payvand, Mihai A. Petrovici. e-mail: julian.goeltz@kip.uni-heidelberg.de; jimmy.weber@ini.uzh.ch; laurak@ini.uzh.ch; melika@ini.uzh.ch; mihai.petrovici@unibe.ch

Recent years have seen an exciting trend in this direction, showing how the performance of SNNs can be improved by optimizing various temporal parameters. Such parameters include neuronal integration time constants[9–15], adaptation time constants[16], and delay variables[17–19].

In particular, spike transmission delays have been predicted to significantly enrich the information processing capabilities of spiking networks[20,21], but specific applications to computationally demanding tasks had remained an open issue. However, recent evidence suggests that a co-optimization of synaptic weights and delays is possible and can significantly reduce the number of training parameters in an SNN, without loss of accuracy[22,23]. This finding is especially important for neuromorphic architectures that target limited on-chip memory.

Nevertheless, from an algorithmic perspective, optimizing delays in SNNs remains an ongoing research problem. Previous literature has largely focused on either exploiting heterogeneity in delay parameters, while limiting gradient-based training to the weights, resulting in selecting the suitable delays[18,23–25], or using evolutionary, not gradient-based algorithms to find delay parameters[26]. Recently, several approaches based on surrogate gradients[27] have been proposed, using convolutional kernels[22] or finite difference methods[19,28]. The underlying surrogate-gradient approach partially addresses, at the expense of both exactness and the conservative property the gradient field[29], the discontinuities of (dis)appearing spikes by smoothing out the spiking threshold. These types of algorithms operate in discrete time, which requires the storage of neuronal activities as binary vectors over the entire history of the SNN.

In addition, from a hardware perspective, there is a growing number of neuromorphic platforms that support the emulation of delays. These implementations require additional memory elements and parameter sets to retain the information of the incoming spike for a controllable amount of time. Previous implementations of on-chip delays using Complementary Metal-Oxide-Semiconductor (CMOS) technology have used digital circuits[23,30–33], active analog circuits[34–37], or mixed-signal solutions[38]. Furthermore, emerging memory technologies such as Resistive Random Access Memory (RRAM) have also been used to realize delay elements, taking advantage of their non-volatile, small three-dimensional footprint, and zero-static-power properties[18,32]. This increasing abundance of neuromorphic substrates offering configurable delays reveals an implicit call for algorithms capable of exploiting these novel capabilities.

In this work, we present DelGrad, which, to the best of our knowledge, is the first exact, analytical solution for gradient-based, hardware-compatible co-learning of delays and weights, using exclusively spike times for the computation of parameter updates. Its hardware compatibility stems from an algorithm-hardware co-design approach: the algorithm was developed with physical hardware systems in mind. It considers the real-valued nature of spike times on analog systems, as well as system-level constraints such as low I/O bandwidth and limited on-chip memory, resulting in a method that is inherently hardware-friendly. Compared to previous approaches, this spike-time-based method simultaneously increases precision and computational efficiency, while also minimizing the required memory footprint of the model. Under DelGrad, we quantitatively study the effect of different types of delays in relation to the size and performance of SNNs. Finally, to experimentally demonstrate the efficacy of our approach, we utilize DelGrad to perform chip-in-the-loop training of a delay-based SNN on a mixed-signal neuromorphic chip.

Results

Training delays in SNNs with DelGrad

While spiking neural networks share their overall structure with the more well-known artificial neural networks, the intrinsic dynamics in the networks are different. In SNNs, each unit in the network is a model of a spiking neuron, i.e., communicating with binary all-or-nothing events called spikes, that carry information in their precise timing (Fig. 1). Here, we employ the leaky integrate-and-fire (LIF) neuron model, which despite its relative simplicity captures the most important properties of biological neurons and therefore often serves as a "standard model" in computational neuroscience and neuromorphic engineering[39,40]. Among these properties are foremost the spiking communication and the leaky-integrator dynamics: all input events are integrated on the membrane potential, which, after an excitation, slowly decays back to its resting state. The precise dynamics of a network of LIF neurons are determined by parameters of the neurons, but also by the inter-neuron connectivity and its parametrization, which includes synaptic weights and transmission delays.

To train the parameters of an ANN, the error backpropagation algorithm[41,42], which optimizes parameters via gradient descent, has become the de facto standard. In contrast, only recently has it become clear that in the case of SNNs the non-differentiability of spikes is, in fact, not an impediment for performing gradient-based optimizations. The developed optimization methods for training SNNs can be roughly split into two groups: approximate, surrogate gradient approaches[19,27,43,44], and methods that employ exact spike time gradients[24,45–48]. In the following, we base our study on the exact gradient methods described in ref. 45.

The subthreshold dynamics of the membrane potential u_{m} of an LIF neuron with exponential current-based synapses are governed by the differential equation

$$\tau_{\mathrm{m}}\dot{u}_{\mathrm{m}}(t) = [E_\ell - u_{\mathrm{m}}(t)] + I_{\mathrm{s}}(t)/g_\ell \tag{1}$$

with neuronal time constant τ_{m}, leak potential E_ℓ, leak conductance g_ℓ and synaptic input current I_{s} defined as

$$I_{\mathrm{s}}(t) = \sum_i \Theta(t - t_i) w_i \exp(-(t - t_i)/\tau_{\mathrm{s}}) \tag{2}$$

where $\Theta(t-t_i)$ is the Heaviside step function, w_i is the weight associated with the synapse receiving a spike at time t_i, and τ_{s} is the synaptic time constant. I_{s} thus outputs a current which is a first-order low pass filter of the input spike train, represented by the exponential kernel $\exp(-(t - t_i)/\tau_{\mathrm{s}})$. I_{s} is itself further leaky integrated by the neuron's membrane, with time constant τ_{m}. Upon crossing the spiking threshold ϑ, the membrane is reset to V_{reset} for a refractory period τ_{ref}, during which the neuron does not react to any further input spikes, and the neuron emits an output spike.

The response function of a neuron, thus, fundamentally, maps a sequence of input spike times t_i to a sequence of output spike times T_i. Here, we focus on a single output spike per neuron for ease-of-notation, but the method can be straightforwardly extended to multi-spike scenarios, as described in Section SI.D. For one such output spike time T, under a parametrization given by the synaptic weights w_i of the incoming connections, we can write T as a function of the set of input weights $\{w_i\}$ and set of input spikes $\{t_i\}$:

$$T(\{t_i\} \cup \{w_i\}) \ . \tag{3}$$

Under certain conditions, depending on the values of the neuronal and synaptic time constants τ_{m} and τ_{s}, the function T becomes analytic, as discussed in ref. 45.

For example, for $\tau_{\mathrm{m}} = \tau_{\mathrm{s}}$

$$T = \tau_{\mathrm{s}}\left\{\frac{b}{a_1} - \mathcal{W}\left[-\frac{g_\ell\vartheta}{a_1}\exp\left(\frac{b}{a_1}\right)\right]\right\} \tag{4}$$

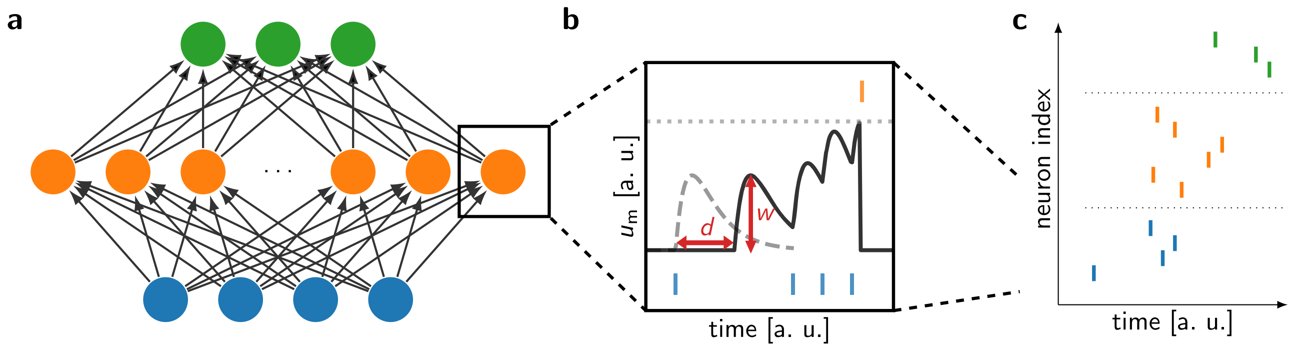

Fig. 1 | Information flow in a (SNN). a Network architecture of a feed-forward SNN with a spiking input layer at the bottom, a hidden layer in the middle and the output layer on top. While the methods described in this manuscript are applicable to many different network architectures, the structure depicted in (**a**), with a variable size of the hidden layer, is used in the following. **b** Zoom in on the information processing in a single leaky integrate-and-fire (LIF) neuron in the hidden layer. Incoming spikes (blue, bottom) are integrated by the neuron's membrane u_m and generate postsynaptic potentials (PSPs), which accumulate additively. Once the membrane potential passes a threshold (gray dashed line), an output spike (orange, top) is generated and passed on to the neurons in the next layer. The PSP amplitudes are modulated by the respective synaptic weights w (vertical red arrow); these are the parameters that are conventionally adapted during learning. Learnable transmission delays d (horizontal red arrow) shift PSPs in time, providing additional temporal processing power to the neuron. **c** Zoom out to a raster plot of the full spiking activity in the network. The information passed between the layers is encoded in the timing of the spikes. As sketched in the raster plot, in the experiments in this manuscript, we employ TTFS coding, i.e., each neuron spikes only once, however our method also generalizes to multi-spike scenarios (Section SI.D) if required by the task.

and for $\tau_m = 2\tau_s$

$$T = 2\tau_s \ln\left[\frac{2a_1}{a_2 + \sqrt{a_2^2 - 4a_1 g_\ell \vartheta}}\right] \tag{5}$$

where a_i and b are explicit functions of w_i, t_i and $\mathcal{W}$ is the Lambert W function (see Eq. SI.5). Writing the spike time in this way enables us both to perform an efficient, event-based forward pass and to train this network, by calculating the exact gradient of the output of the network with respect to the network parameters.

In a multi-spike scenario (Section SI.D), all subsequent spikes after the first spike can be calculated by taking the reset into account and solving the equation for different initial conditions. Ultimately, this results in similar expressions as Eqs. (3–5).

To optimize the network parameters via gradient descent, we base the update of each parameter θ on its influence on the loss $\mathcal{L}$, the gradient $\partial\mathcal{L}/\partial\theta$. Employing the chain rule, this gradient is iteratively composed of $\partial T/\partial\theta$ and $\partial T/\partial t_i$, i.e., derivatives of the above equations.

Specifically for a network with parameters w_i, $\partial T/\partial w_i$ allows us to link a deviation in an output spike time to a change in weight parameters, while $\partial T/\partial t_i$ relates this deviation in the output to a deviation in the input, thereby enabling us to propagate an error in the spike time backwards through the neuron. Crucially, just like the original Eqs. (5 and 4), and in contrast to surrogate-gradient-based approaches, these derivatives only depend on spike times and parameters of the network and can be computed without the calculation or measurement of the membrane potential. This allows us to perform a fully event-based forward and backward pass, without any need for temporal discretization of forward or backwards dynamics.

Transmission delays of spike signals can now simply be introduced as additive parameters d to the original spike times $t^{\not d}$:

$$t_i^d = t_i^{\not d} + d_i \tag{6}$$

These delayed spike times then become the relevant input for the postsynaptic neuron. As above, derivatives of this expression provide the necessary quantities for adapting the delays and for backpropagating the spike timing errors. In this case, the corresponding equations are trivial:

$$\frac{\partial t_i^d}{\partial t_i^{\not d}} = 1 \quad \text{and} \quad \frac{\partial t_i^d}{\partial d_i} = 1. \tag{7}$$

Treating spike times as continuous variables, different from the time-binning performed in other approaches[19,22], allows this natural implementation of full-precision delays as well as the exact and simple training of the delay parameters. We note that these considerations do not depend on a specific network setup and thus apply to any activity pattern in arbitrary spiking networks. In the following, we focus our attention on the particular problem of pattern classification, for which we employ a specific network architecture and spike coding scheme (Fig. 1).

To take advantage of a well-established architectural paradigm, we now consider information propagation in hierarchical feed-forward networks. As also shown in the corresponding computational graph (Fig. 2a, solid black arrow), the input $\boldsymbol{t}^0$ is passed through the sequence of layers until it reaches the output (we use bold symbols to denote non-scalar variables). The gradient of the chosen loss function $\mathcal{L}$ then goes backwards through the network (dashed red arrow) for optimizing the parameters. In the forward pass, the only information that is transmitted is spike times $\boldsymbol{t}^l$; in the backward pass, we transmit the gradient of the loss function $\partial\mathcal{L}/\partial\boldsymbol{t}^l$, but note that it is also only evaluated at the times when neurons spike.

For SNNs with delays, the computational graph differentiates between two types of layer: neuron layers and delay layers (Fig. 2). Both layers receive input spikes $\boldsymbol{t}^{l-1}$ and return output spikes $\boldsymbol{t}^l$, but using different forward transfer functions, as given by Eq. (4)/Eqs. (5 and 6), respectively. In the backward direction, they pass the partial derivative $\partial\mathcal{L}/\partial\boldsymbol{t}^{l-1}$ discussed above.

Figure 2b, c highlight the similarity of the two layer types: both neuron and delay layers take spike trains as an input and produce spike trains as an output in the forward pass, and propagate gradients of the loss with respect to the corresponding spike times in the backward pass. Their respective computations are carried out sequentially, as depicted in Fig. 2a, with delay layers stacked in between neuron layers.

In Fig. 3a we distinguish between different types of delays: axonal delays d_{axo} on a neuron's output, dendritic delays d_{den} on a neuron's input, and synaptic delays d_{syn} that are specific for every connection between pairs of neurons. Their respective natural representations as

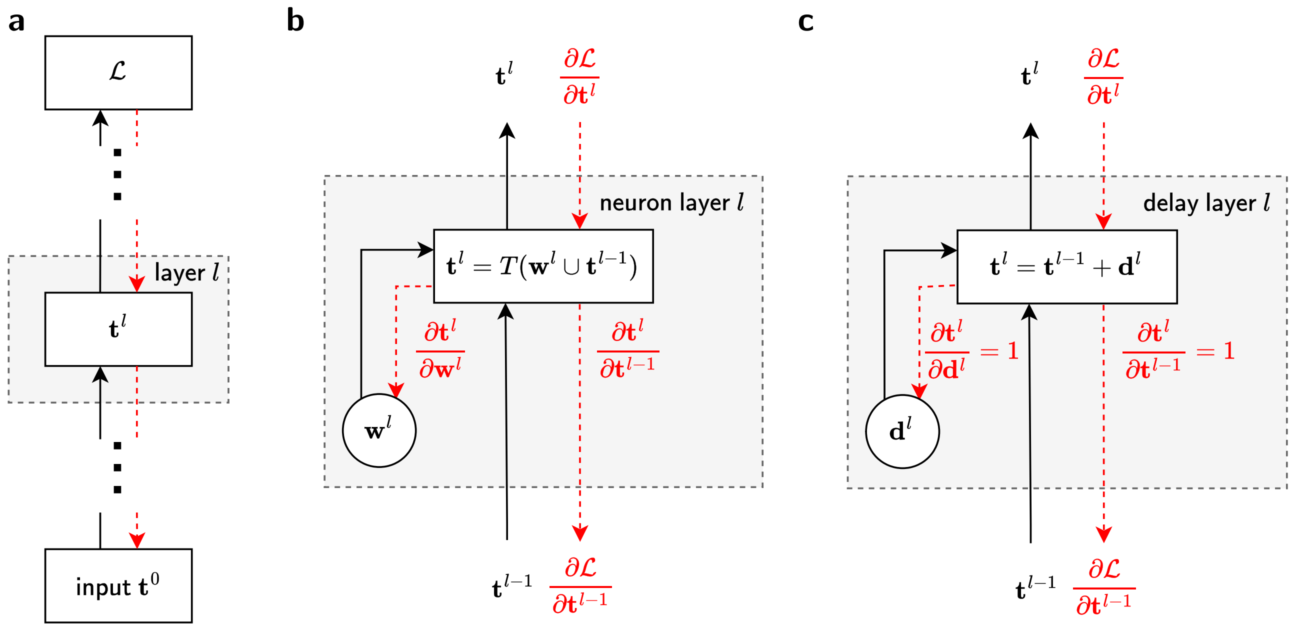
a
$\mathcal{L}$
layer l
$\mathbf{t}^l$
input $\mathbf{t}^0$
b
$\mathbf{t}^l$
$\frac{\partial \mathcal{L}}{\partial \mathbf{t}^l}$
neuron layer l
$\mathbf{t}^l = T(\mathbf{w}^l \cup \mathbf{t}^{l-1})$
$\frac{\partial \mathbf{t}^l}{\partial \mathbf{w}^l}$
$\frac{\partial \mathbf{t}^l}{\partial \mathbf{t}^{l-1}}$
$\mathbf{w}^l$
$\mathbf{t}^{l-1}$
$\frac{\partial \mathcal{L}}{\partial \mathbf{t}^{l-1}}$
c
$\mathbf{t}^l$
$\frac{\partial \mathcal{L}}{\partial \mathbf{t}^l}$
delay layer l
$\mathbf{t}^l = \mathbf{t}^{l-1} + \mathbf{d}^l$
$\frac{\partial \mathbf{t}^l}{\partial \mathbf{d}^l} = 1$
$\frac{\partial \mathbf{t}^l}{\partial \mathbf{t}^{l-1}} = 1$
$\mathbf{d}^l$
$\mathbf{t}^{l-1}$
$\frac{\partial \mathcal{L}}{\partial \mathbf{t}^{l-1}}$

Fig. 2 | Computational graph of a multi-layer SNN with spike-time information encoding and adjustable delay and weight parameters. a Graph for a multi-layer network with spike times $\boldsymbol{t}^0$ injected into the bottom (1st) layer. In the forward pass (black arrows), each layer l takes spike times as inputs and returns spike times as outputs that go into the next layer. The spike times of the topmost layer are used to compute the loss function $\mathcal{L}$. The backward pass (red dashed arrows) starts at the loss and passes the gradients backwards through the layers. We consider two types of layers: neuron layers and delay layers. **b** Neuron layer with parameters $\boldsymbol{w}^l$ (synaptic weights). These are used together with the input spike times $\boldsymbol{t}^{l-1}$ to calculate the output spike times $\boldsymbol{t}^l$ according to the nonlinear relation described in Eqs. (4 and 5). **c** Delay layer with parameters $\boldsymbol{d}^l$ that are added (linearly) to the input spike times $\boldsymbol{t}^{l-1}$ to calculate the output spike times $\boldsymbol{t}^l$ as in Eq. (6).

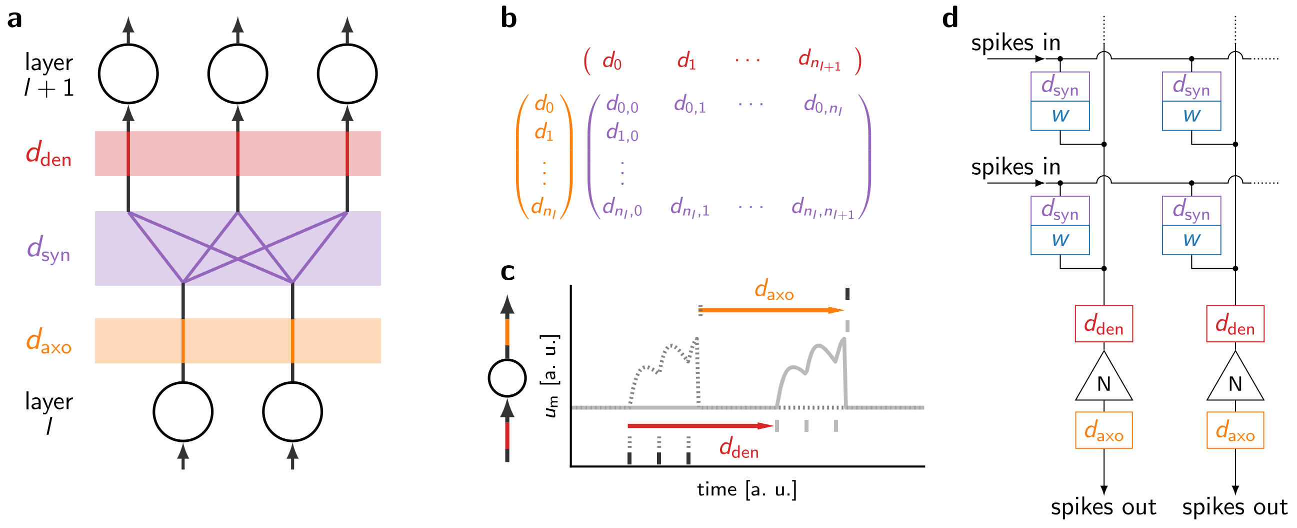

Fig. 3 | Illustrating different types of delays. a From bottom to top: axonal delays shift the timing of the neuron's outgoing spikes by d_{axo} (orange); synaptic delays shift the timing of spikes by a specific value d_{syn} for each pair of pre- and post-synaptic neuron (purple); dendritic delays shift the timing of the incoming spikes into a neuron by d_{den} (red). **b** Vector and matrix representation of the different types of delays and their dimensionality as a function of the number of pre- and post-synaptic neurons. **c** Equivalent effect of the dendritic and axonal delays on the output spike time of a neuron, due to the time-shift invariance of the temporal dynamics of a LIF neuron. **d** Schematic illustration of the location of synaptic, dendritic and axonal delay components in a generic neuromorphic crossbar architecture.

column vectors, row vectors and matrices are shown in Fig. 3b. The memory footprint of axonal and dendritic delays thus scales linearly with the number of neurons in the network, while for synaptic delays, it scales linearly with the network depth and quadratically with its width.

While in principle different types of delays can be simultaneously present in a network and can be combined with each other, it is important to note that, as illustrated in Fig. 3c, combining dendritic and axonal delays for the same neuron is redundant: as neuronal dynamics are invariant to temporal shifts, it is equivalent for the input spikes to arrive with a delay $d_{\text{den}} = d$, resulting in a delayed output spike (red arrow and gray curve), or for the output spike of the neuron to be directly delayed with $d_{\text{axo}} = d$ (orange arrow and membrane dynamics in black).

Given the resource constraints of neuromorphic systems, we investigate the performance benefits incurred by the different delay types, with different requirements on the memory resources. Although a quantitative evaluation of the exact energy consumption, chip area and design complexity of different delay architectures heavily depends on system architecture and the chosen design (e.g., analog vs. digital and circuit topology), some generic statements can be made using the mathematical representation of the delay elements.

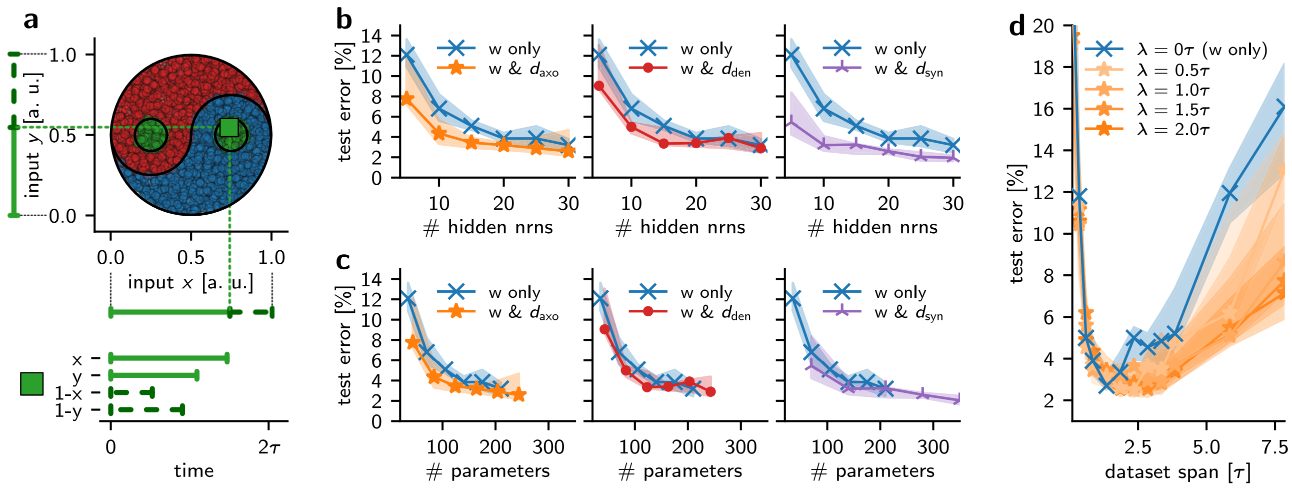

Fig. 4 | Classification task and simulation results. a The Yin-Yang (YY) task[50] consists of the classification of dots based on whether they belong to the Yin (red), Yang (blue), or dot (green) regions, as illustrated in (**a**). The input features are the two-dimensional coordinates (x, y) of the image, along with their mirrored values (1−x, 1−y), totaling four features. These features are encoded into spike times, such that a larger value of x or y coordinate results in a later spike time for x or y and an early spike time for its mirrored version 1−x or 1−y respectively. For more details on the encoding, see the original publication[50]. **b** Test error as a function of the number of hidden neurons in an SNN, using different delay types. The solid lines and markers show the median of the error, and the shaded areas illustrate the interquartile ranges (IQRs) for 10 seeds. **c** Same data as in **b** but as a function of the number of trainable parameters in the networks, i.e., counting the distinct weights and, if applicable, delays. **d** Impact of axonal delays as a function of the temporal scale of the dataset. The trainable delays cover a range λ as indicated by the orange hue. The network performance without delays is shown in blue.

For typical crossbar architectures (Fig. 3d), the synaptic delay mechanisms are often located within the crossbar array and therefore scale with the product of the array's input and output size. In contrast, dendritic and axonal delays can be located in the periphery of the array, and thus their required area scales linearly with the input and output array size, respectively. It is worth noting that an important property of axonal delay mechanisms is that they are located directly after the neurons' output and therefore only need to operate on sparse events. In contrast, dendritic delays are located directly before the neurons' input, and after the input signals have been scaled by the synaptic weight.

Depending on the design choices, in particular on whether the synaptic integration happens in the synapses or in the neurons, this may require more complex circuitry. Note also that neurons usually receive more spikes than they emit, so the required buffering may also increase the corresponding hardware footprint of dendritic delay implementations.

Simulation results

This section evaluates DelGrad's ability to co-train delays and weights, demonstrating improved accuracy and parameter efficiency over weight-only training. By systematically studying the effect of hidden layer size and comparing different delay types, we highlight the advantages of incorporating learnable delays.

We benchmark a PyTorch[49] implementation of the DelGrad method using the Yin-Yang (YY)[50] dataset to evaluate the impact of transmission delays on the SNN performance, and assess how this varies with the network size. This dataset is selected for its advantageous properties—compactness, making it amenable for hardware prototyping, training speed, as well as discriminatory power between network architectures and training paradigms: it leaves ample room for benchmarking above the accuracy achievable with a linear classifier. The task is to classify the region of a Yin-Yang image to which a point in the image plane belongs, as illustrated in Fig. 4a. The coordinates of the point (x, y) and their mirrored values (1−x, 1−y) are encoded into spike times, such that a larger value of the coordinate results in a later spike time, and an early spike time for its mirrored version.

The network architecture as shown in Fig. 1 is a feed-forward multi-layer configuration with four input neurons, followed by a variable-size neuron layer (hidden layer) and finally an output layer, comprising three neurons for the three classes (a study on deeper networks is provided in Section SI.A.1). Delay layers are inserted between neuron layers, as previously illustrated in the computational graph (Fig. 2). The neurons have no configurable biases, and the time constants are configured such that $\tau_m = 2\tau_s$. Thus, we utilize Eq. (5) for training. The refractory period τ_{ref} is set to infinity, such that all neurons only spike once. The output is represented in a time-to-first-spike (TTFS) decoding scheme, where the first output neuron to spike indicates the predicted class for a given input. To avoid negative or excessively large values for the delays, the effective delay d is calculated as a logistic function of a trainable parameter θ_d such that $d = \lambda\,\sigma(\theta_d)$, which ensures that the delays remain bounded between 0 and λ. Further details can be found in Section SI.A.

We have chosen the time-invariant mean squared error (MSE) loss to improve accuracy and stability of training:

$$\mathcal{L}_{\Delta\mathrm{MSE}}[\boldsymbol{t}, n^\star; \Delta_t] = \frac{1}{2}\sum_{n \neq n^\star} \left[(t_n - t_{n^\star}) - \Delta_t\right]^2 \qquad (8)$$

where $n^\star$ and n denote the respective indices of the correct and wrong label neurons and Δ_t is a freely chosen parameter. The purpose of introducing Δ_t into the loss is to achieve a specific separation of Δ_t between the spike times of the correct and incorrect label neurons, instead of providing precise target spike times. To ensure a balance between model accuracy and hardware compatibility, Δ_t is set to $0.2\tau_s$ in our simulations.

We investigate the effects of different types of delay layers on accuracy, compared to configurations without any delays. Fig. 4 reports the performance of our approach on the YY dataset across different network sizes. Fig. 4b shows the percentage of misclassified samples in the test set (test error) of the network as a function of the number of hidden layers. It demonstrates that co-training delays alongside the weights always improves performance, regardless of the specific type of delay. Among the delay-augmented configurations, the variant with synaptic delays outperforms the ones with axonal- or dendritic-only parameters. This is in line with expectations, as synaptic

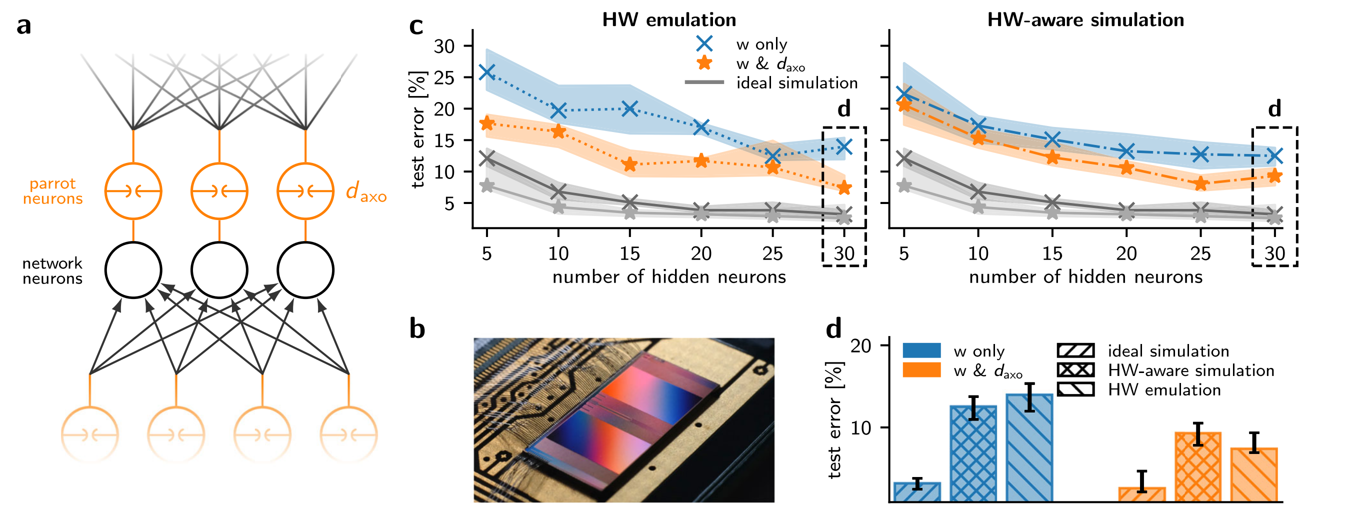

Fig. 5 | In-the-loop training with on-chip axonal delays on BrainScaleS-2. **a** Schematic illustration of the network architecture for on-chip axonal delays; here, we apply this generic approach to the BrainScaleS-2 neuromorphic hardware. Each neuron in the network (black) is paired with a parrot neuron (orange) connected in a one-to-one scheme. The parrot neuron repeats each of its input spikes with a configurable delay. **b** Photograph of the BrainScaleS-2 neuromorphic chip (taken from[65]). **c** Median test errors and IQR on the Yin-Yang dataset when training network weights and axonal delays (orange) or only weights (blue). The dash-dotted lines indicate a hardware-aware simulation (cf. Section SI.A.3) and the dotted lines the hardware emulation results. For comparison, we also show the ideal software simulation results from Fig. 4b in gray. The shaded areas indicate the IQR over 10 runs with different seeds. The values for networks with 30 hidden neurons (highlighted by the dashed box) are shown for a better comparison in (**d**). **d** Detailed comparison of performances at 30 hidden neurons of an ideal simulation, hardware-aware simulation and emulation on neuromorphic hardware.

delays offer the greatest configurable parameter space among the three delay variants.

Figure 4c displays the same test errors, but now as a function of the number of parameters. This representation reveals that, at least for the YY dataset, delay-augmented networks with the same number of parameters perform similarly well, regardless of the type of delay. As before, for a given number of parameters, the co-training of delays always yields at least as good results as the training of synaptic weights alone. In other words, for the same memory footprint, a mix of both weights and delays is better than just synaptic weights.

Notably, the functional benefit of trainable delays depends to a great extent on the temporal structure of the data. In particular, we expect the training of delays to have a larger impact if the input data spans longer time scales. For YY, it is straightforward to change the temporal volume occupied by the dataset by modifying its span—the time difference between the earliest and latest possible input spikes. Fig. 4d shows the effect of trainable delays across these different spans. For small spans, errors are high because the temporal dynamics in the data are too fast for the intrinsic dynamics of the LIF neurons. However, beyond a certain point, we always observe a clear benefit of co-training delays and weights as opposed to weights alone. Furthermore, for a larger dataset span where input spikes can consequently be further apart, the range λ of available delays that are able to push the PSPs together becomes increasingly relevant.

Optimal learning rates are determined through hyperparameter optimization for each configuration of neuron and delay layers. Across all investigated settings, our approach demonstrates robust training convergence (Fig. SI.2a) as well as exploitation of all available resources (Fig. SI.2b). Overall, these results clearly evince the added value of learning delays, as well as the ability of our algorithm to capitalize on this potential.

Hardware results

To calculate the gradients for training weights and delays in SNNs, DelGrad only requires spike time recordings, compared to surrogate-gradient-based approaches, which also require recording membrane potential. Therefore, DelGrad is ideally suited for implementation on a variety of neuromorphic substrates, whose output is spike-based, by design[51]. Here, we demonstrate the flexibility of our method by describing a successful application in silico, on the neuromorphic platform BrainScaleS-2 (BSS-2), that does not natively support delays.

The BSS-2 system (Fig. 5b,[7,52]) is built around a mixed-signal neuromorphic chip with 512 physical neuron circuits. The neuron dynamics are accelerated compared to biological time scales by a factor of 10^3. The neuron circuits emulate the dynamics of the adaptive exponential leaky integrate-and-fire (AdEx) model[53] with individually configurable parameters for each neuron[54]. Neighboring neuron circuits can be connected to form multi-compartment neurons[55]. The connectivity between the neurons on the chip can be configured arbitrarily within the constraints of the two 256×256 synaptic crossbar arrays. The synaptic weights are configured digitally with 6 bit resolution.

Although the current generation of BSS-2 does not natively support on-chip delays, we present an approach that allows us to explore the computational potential of delays for the current substrate. We realize on-chip axonal delays by re-purposing a subset of the available neurons as delay elements. For this, we utilize the adaptation circuitry as well as multi-compartment functionality of the neurons on chip. This allows us to perform in-the-loop training of both synaptic weights and the on-chip axonal delays and illustrate the computational advantage obtained by the inclusion of delays. The details of the delay implementation are provided in Section SI.B.1. We additionally provide a proof-of-concept of a different on-chip realization of axonal delays using LIF neuron dynamics, which is the most widely adopted neuron model for hardware platforms (see Section SI.B.2).

Even without an explicit hardware implementation of delays, an effective axonal delay can be achieved by exploiting the dynamics of the on-chip infrastructure. For that, a “parrot neuron” is connected to the output of a neuron that is part of the actual trained network (Fig. 5a). For any spike that the network neuron produces, the parrot neuron is configured to elicit a spike after the desired delay.

In our implementation on BSS-2, this behavior is achieved via the interplay between the two neuron compartments that form a parrot neuron. The first compartment reacts to each incoming spike with a reset, which clamps the membrane voltage to the reset potential for a refractory period, during which the neuron is not responsive to incoming spikes. After the end of the configurable refractory period, the second compartment becomes active and its adaptation mechanism almost instantly triggers an output spike of the parrot neuron. Therefore, a spike is generated after the configurable refractory period, used here as the delay. For a more detailed description of the mechanism see Section SI.B.1. We use this method, as it allows us to control the delay produced by the parrot neuron via its refractory period, which is digitally controlled on BSS-2 using 8 bits of precision. This results in a more precise and easily configurable delay, compared to using an analog variable, and is likely closer to a future implementation of native delays on a BSS-2-like system.

This delay mechanism allows us to train a network with axonal delays on BSS-2. We use an in-the-loop training approach, which means that we present a batch of inputs to the network on chip and record the spike times. The spike times are sent back to the host computer, where the loss and the backward pass are calculated in software. The resulting updates for weights and delays are then used to reconfigure the chip before the next batch is presented.

With this chip-in-the-loop setup, we train and evaluate networks, with synaptic weights alone, as well as networks that incorporate both adjustable weights and axonal delays (Fig. 5c). Similar to the simulation results presented in Fig. 4, we experimentally confirm an accuracy gain over a range of network sizes, for the networks with additional delay parameters compared to the ones with only weight parameters.

Overall, the final test errors reached in the software simulation are lower than the ones measured on hardware. This is expected, as hardware effects such as trial-to-trial variations, fixed-pattern noise and jitter on the on-chip delays disturb the dynamics. To illustrate and characterize these effects, we measured the magnitude of several noise sources found on the hardware and modeled them in a series of hardware-aware simulations. Fig. 5c shows that, when the various sources of noise are realistically modeled based on hardware measurements, the hardware-aware simulations capture the increase in test error similarly to the actual emulation. This confirms that the gap in accuracy between software simulations and hardware experiments is mostly due to the modeled sources of noise. For an in-depth description of the noise models employed in the hardware-aware simulations and an analysis of the impact of the different noise sources on the network performance, we refer to Section SI.A.3.

For an easier comparison, we focus on the most expressive networks with 30 hidden neurons, highlighted by boxes in Fig. 5c, and collect the achieved test errors in Fig. 5d, which amount to 7.40% with axonal delays and 13.95% in the weight-only case on the hardware. A full report of the achieved test errors and IQR, both in hardware-aware simulation and on chip, can be found in Table SI.1. Additionally, we note that the performance gap between the delay and no-delay setup is significantly wider on hardware than in the ideal software simulations. We hypothesize that this effect arises because the YY classification problem, by design, does not require a large network to solve, leading to few learnable parameters, low redundancy, and consequently a greater sensitivity to noise. Introducing axonal delays increases redundancy, due to the higher parameter count. However, this increase is rather small compared to the number of parameters in a weight-only setup. This suggests that the computational properties of the delays are at least partially responsible for making the network more noise resilient, explaining the larger performance gap between the two networks on noisy hardware.

These results illustrate that our method for training delays is not only applicable in ideal software simulations but can also be applied to mixed-signal neuromorphic systems. Additionally, they demonstrate the benefit of learnable delays for neuromorphic platforms, especially in resource-constrained scenarios, and might encourage the inclusion of delay mechanisms in future generations of neuromorphic systems.

Discussion

We have introduced an exact event-based algorithm for training temporal variables, specifically transmission delays, in conjunction with synaptic weights in SNNs. Additionally, we have experimentally validated its effectiveness through both software simulations and neuromorphic hardware implementations.

Delay parameters were previously demonstrated to increase the representational power of the SNNs, even without optimization, just by training the weight parameters to select the useful delays for spatio-temporal feature detection[18,23,56]. However, this optimization-through-selection approach requires an over-allocation of resources in order to provide a sufficiently diverse set of delay parameters from which the best can be selected. To illustrate this, we compare a network of random fixed delays to a network with trained delays using DelGrad. We show in Fig. SI.3 that for the same number of delay parameters, the network with trained delays and weights has a clear accuracy advantage over the network with randomly initialized delays with weight-only optimization. Therefore, it is advantageous to combine a dedicated learning algorithm for transmission delays with hardware capable of configuring them accordingly.

Algorithms based on surrogate gradients for direct training of delay elements have been explored recently, using temporal convolution kernels[22] or numerical solutions that estimate the delay gradients using finite-difference approximations[19]. However, as pointed out in ref. 22, delay training based on finite-difference approximation[19] appears to not be sufficiently accurate to achieve an improvement over fixed, random delays. Additionally, both approaches use a time-stepped framework for calculating the gradients.

As such, delays are represented implicitly in the number of simulation time steps before transmitting a spike. However, as delay parameters are essentially shifts in individual spike times, we argue that it is more natural to have a framework where the information is explicitly represented by these spike times[32,45,46,48,57], and delays are learned as additive parameters for these times. Furthermore, the objective of building efficient asynchronous neuromorphic systems, where time represents itself, is an additional motivation for representing temporal information in spike times[3]. Such representations are naturally available from event-based sensors, where the change in the signal is encoded into spike times using the delta modulation encoding scheme[58–60].

This work brings together all the aforementioned objectives: DelGrad presents an event-based framework for gradient-based co-training of delay parameters and weights, without any approximations, and which meets the typical demands and constraints of neuromorphic hardware, as demonstrated experimentally on an analog mixed-signal neuromorphic system. As such, it takes an important step towards fully exploiting the temporal nature of SNNs for memory- and power-efficient end-to-end event-based neuromorphic systems.

In this work, we have also compared the effect of dendritic, axonal and synaptic delays on the performance of SNNs on a representative task. The synaptic delays have the highest impact on increasing the expressivity of SNNs, compared to using only dendritic or axonal delays. However, from a hardware perspective, the addition of synaptic delays imposes a quadratic growth on the size and thus the on-chip area of the network, compared to a linear growth in the case of axonal and dendritic delays. In fact, we find that when comparing the performance for equal parameter counts, the gap between different types of delays vanishes while the superiority over weight-only training persists. As memory represents one of the most critical constraints on hardware, reducing on-chip memory is of utmost importance. In particular, this means that a redesign of a chip with fewer neurons but with an intrinsic delay mechanism, based on our findings, will save energy while maintaining expressivity and performance. Therefore, our work suggests that it might be practical to consider using dendritic or axonal delays in future hardware designs, combining favorable scaling and improved processing power.

DelGrad provides an advantage in terms of hardware mappability as it only requires recording the spike times from on-chip neurons.

This is in contrast to other approaches[19,22], which need access to the membrane potential of all neurons for surrogate gradient learning[61,62]. Such voltage-based plasticity requires additional components for voltage readout, communication and potentially analog-to-digital conversion. Furthermore, this information is much denser in time than the spikes themselves, imposing further stress on the overall communication bandwidth. In both chip-in-the-loop and on-chip training scenarios, these additional requirements ultimately translate to additional circuitry; not only does this increase the complexity of the chip design, but it also inherently reduces the maximum implementable network size for a chip of a given area. Moreover, if learning is to be implemented on-chip, this additional circuitry will negatively affect the device's energy efficiency during training.

In this work, the YY dataset was used as a proof of concept and first step to benchmark our approach. The YY dataset provides a problem that can not be solved linearly and where the information can be presented using a TTFS encoding, similar to the previous work[45]. As indicated by our experiments in Fig. 4d, the performance boost provided by the inclusion of learnable delays increases when the temporal features of the data span larger time scales.

Therefore, the natural next step will reside in a more thorough benchmarking on larger datasets, and in particular on data with explicit temporal components, such as[63,64]. Especially for data provided by event-based sensors, longer time scales are required, and TTFS might reach its limits as a feasible coding scheme. Although our current software implementation only takes into account a single spike per neuron during the training, this is not a limitation of our proposed mathematical framework and training scheme (see Section SI.D). Additionally, the extension to more complex spike timing codes can go hand in hand with a shift from a feed-forward to a recurrent network architecture.

Data availability

We used the Yin-Yang data set[50], the code is available at https://github.com/lkriener/yin_yang_data_set.

Code availability

Code for the simulations is available at https://github.com/JulianGoeltz/fastAndDeep.

References

1. Olshausen, B. A. & Field, D. J. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature* **381**, 607–609 (1996).
2. Koch, C. & Segev, I. The role of single neurons in information processing. *Nat. Neurosci.* **3**, 1171–1177 (2000).
3. Mead, C. Neuromorphic electronic systems. In *Proc. IEEE*, Vol. **78**, no. 10, 1629–1636 (1990).
4. Indiveri, G. & Liu, S.-C. Memory and information processing in neuromorphic systems. In *Proc. IEEE*, Vol. **103**, 8, 1379–1397 (2015).
5. Frenkel, C., Bol, D. & Indiveri, G. Bottom-up and top-down approaches for the design of neuromorphic processing systems: tradeoffs and synergies between natural and artificial intelligence. Preprint at https://doi.org/10.1109/JPROC.2023.3273520 (2023).
6. Furber, S. B., Galluppi, F., Temple, S. & Plana, L. A. The spinnaker project. In *Proc. IEEE*, Vol. 102, no. 5, 652–665 (2014).
7. Billaudelle, S. et al. Versatile emulation of spiking neural networks on an accelerated neuromorphic substrate. In *Proc. International Symposium on Circuits and Systems (ISCAS)* (IEEE, 2020).
8. Bohte, S. M. The evidence for neural information processing with precise spike-times: a survey. *Nat. Comput.* **3**, 195–206 (2004).
9. Yin, B., Corradi, F. & Bohté, S. M. Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks. *Nat. Mach. Intell.* **3**, 905–913 (2021).
10. Rao, A., Plank, P., Wild, A. & Maass, W. A long short-term memory for AI applications in spike-based neuromorphic hardware. *Nat. Mach. Intell.* **4**, 467–479 (2022).
11. Nowotny, T., Turner, J. P. & Knight, J. C. Loss shaping enhances exact gradient learning with eventprop in spiking neural networks. *Neuromorphic Comput. Eng.* **5**, 014001 (2025).
12. Bittar, A. & Garner, P. N. A surrogate gradient spiking baseline for speech command recognition. *Front. Neurosci.* **16**, 865897 (2022).
13. Perez-Nieves, N., Leung, V. C. H., Dragotti, P. L. & Goodman, D. F. M. Neural heterogeneity promotes robust learning. *Nat. Commun.* **12,** 5791 (2021).
14. Fang, W. et al. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proc. IEEE/CVF International Conference on Computer Vision*, 2661–2671 (IEEE, 2021).
15. Moro, F., Aceituno, P. V., Kriener, L. & Payvand, M. The role of temporal hierarchy in spiking neural networks. Preprint at https://doi.org/10.48550/arXiv.2407.18838 (2024).
16. Bellec, G., Salaj, D., Subramoney, A., Legenstein, R. & Maass, W. Long short-term memory and learning-to-learn in networks of spiking neurons. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 795–805 (Curran Associates Inc., 2018).
17. Hammouamri, I., Masquelier, T. & Wilson, D. G. Mitigating Catastrophic Forgetting in Spiking Neural Networks through Threshold Modulation. *Transactions on Machine Learning Research* https://openreview.net/forum?id=15SoThZmtU (2022).
18. DAgostino, S. et al. Denram: neuromorphic dendritic architecture with RRAM for efficient temporal processing with delays. *Nat. Commun.* **15**, 3446 (2024).
19. Shrestha, S. B. & Orchard, G. S. LAYER: spike Layer Error Reassignment in Time. In *Proc. 32nd International Conference on Neural Information Processing Systems*. 1419–1428 (Curran Associates, Inc., 2018).
20. Maass, W. & Schmitt, M. On the complexity of learning for spiking neurons with temporal coding. *Inf. Comput.* **153**, 26–46 (1999).
21. Izhikevich, E. M. Polychronization: computation with spikes. *Neural Comput.* **18**, 245–282 (2006).
22. Hammouamri, I., Khalfaoui-Hassani, I. & Masquelier, T. Learning delays in spiking neural networks using dilated convolutions with learnable spacings. *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=4r2ybzJnmN (2024).
23. Patiño-Saucedo, A. et al. Empirical study on the efficiency of spiking neural networks with axonal delays, and algorithm-hardware benchmarking. In *Proc. International Symposium on Circuits and Systems (ISCAS)*, 1–5 (IEEE, 2023).
24. Bohte, S. M., Kok, J. N. & La Poutré, H. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing* **48**, 17–37 (2002).
25. Gerstner, W., Kempter, R., van Hemmen, J. L. & Wagner, H. A neuronal learning rule for sub-millisecond temporal coding. *Nature* **383**, 76–78 (1996).
26. Schuman, C. D., Mitchell, J. P., Patton, R. M., Potok, T. E. & Plank, J. S. Evolutionary optimization for neuromorphic systems. In *Proc. Annual Neuro-Inspired Computational Elements Workshop* (NICE), 1–9 (Association for Computing Machinery, 2020).
27. Neftci, E. O., Mostafa, H. & Zenke, F. Surrogate gradient learning in spiking neural networks: bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Process. Mag.* **36**, 51–63 (2019).
28. Sun, P., Chua, Y., Devos, P. & Botteldooren, D. Learnable axonal delay in spiking neural networks improves spoken word recognition. *Front. Neurosci.* **17**, 1275944 (2023).

29. Gygax, J. & Zenke, F. Elucidating the theoretical underpinnings of surrogate gradient learning in spiking neural networks. *Neural Comput.* **37**, 886–925 (2025).
30. Madhavan, A., Sherwood, T. & Strukov, D. Race logic: a hardware acceleration for dynamic programming algorithms. *ACM SIGARCH Comput. Archit. News* **42**, 517–528 (2014).
31. Davies, M. et al. Loihi: a neuromorphic manycore processor with on-chip learning. *IEEE Micro* **38**, 82–99 (2018).
32. Madhavan, A., Daniels, M. W. & Stiles, M. D. Temporal state machines: using temporal memory to stitch time-based graph computations. *ACM J. Emerg. Technol. Comput. Syst.* **17**, 1–27 (2021).
33. Merolla, P. A. et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* **345**, 668–673 (2014).
34. Sheik, S., Chicca, E. & Indiveri, G. Exploiting device mismatch in neuromorphic VLSI systems to implement axonal delays. In *Proc. International Joint Conference on Neural Networks (IJCNN)*, 1–6 (IEEE, 2012).
35. Wang, R., Jin, C., McEwan, A. & van Schaik, A. A programmable axonal propagation delay circuit for time-delay spiking neural networks. In *Proc. International Symposium of Circuits and Systems (ISCAS)*, 869–872 (IEEE, 2011).
36. Huayaney, F. L. M., Nease, S. & Chicca, E. Learning in silicon beyond STDP: a neuromorphic implementation of multi-factor synaptic plasticity with calcium-based dynamics. *IEEE Trans. Circuits Syst. I* **63**, 2189–2199 (2016).
37. Gerber, S., Steiner, M., Indiveri, G. & Donati, E. et al. Neuromorphic implementation of ecg anomaly detection using delay chains. In *Proc. Biomedical Circuits and Systems Conference (BioCAS)*, 369–373 (IEEE, 2022).
38. Richter, O. et al. DYNAP-SE2: a scalable multi-core dynamic neuromorphic asynchronous spiking neural network processor. *Neuromorphic Comput. Eng.* **4**, 014003 (2024).
39. Lapicque, L. Recherches quantitatives sur l'excitation electrique des nerfs traitee comme une polarization. *J. Physiol. Pathol.* **9**, 620–635 (1907).
40. Abbott, L. Lapicque's introduction of the integrate-and-fire model neuron (1907). *Brain Res. Bull.* **50**, 303–304 (1999).
41. Linnainmaa, S. *The Representation of the Cumulative Rounding Error of an Algorithm as a Taylor Expansion of the Local Rounding Errors*. MSc Thesis (in Finnish) 6–7 (University of Helsinki, 1970).
42. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning representations by back-propagating errors. *Nature* **323**, 533–536 (1986).
43. Zenke, F. & Ganguli, S. Superspike: supervised learning in multilayer spiking neural networks. *Neural Comput.* **30**, 1514–1541 (2018).
44. Renner, A., Sheldon, F., Zlotnik, A., Tao, L. & Sornborger, A. The backpropagation algorithm implemented on spiking neuromorphic hardware. *Nat. Commun.* **15**, 9691 (2024).
45. Göltz, J. et al. Fast and energy-efficient neuromorphic deep learning with first-spike times. *Nat. Mach. Intell.* **3**, 823–835 (2021).
46. Wunderlich, T. C. & Pehle, C. Event-based backpropagation can compute exact gradients for spiking neural networks. *Sci. Rep.* **11**, 12829 (2021).
47. Klos, C. & Memmesheimer, R.-M. Smooth exact gradient descent learning in spiking neural networks. *Phys. Rev. Lett.* **134**, 027301 (2025).
48. Stanojevic, A. et al. High-performance deep spiking neural networks with 0.3 spikes per neuron. *Nat. Commun.* **15**, 6793 (2024).
49. Paszke, A. et al. Pytorch: an imperative style, high-performance deep learning library. *Adv. Neural Inf. Process. Syst.* **32**, 8026–8037 (2019).
50. Kriener, L., Göltz, J. & Petrovici, M. A. The Yin-Yang dataset. In *Proc. Neuro-Inspired Computational Elements Conference*, NICE 2022, 107–111 (Association for Computing Machinery, 2022).
51. Boahen, K. A. *Communicating Neuronal Ensembles between Neuromorphic Chips*. 229–259 (Kluwer Academic Publishers, 1998).
52. Pehle, C. et al. The BrainScaleS-2 accelerated neuromorphic system with hybrid plasticity. *Front. Neurosci.* **16** https://www.frontiersin.org/articles/10.3389/fnins.2022.795876 (2022).
53. Brette, R. & Gerstner, W. Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *J. Neurophysiol.* **94**, 3637–3642 (2005).
54. Billaudelle, S., Weis, J., Dauer, P. & Schemmel, J. An accurate and flexible analog emulation of AdEx neuron dynamics in silicon. In *Proc. 29th International Conference on Electronics, Circuits and Systems (ICECS)*, 1–4 (IEEE, 2022).
55. Schemmel, J., Kriener, L., Müller, P. & Meier, K. An accelerated analog neuromorphic hardware system emulating nmda-and calcium-based non-linear dendrites. In *Proc. International Joint Conference on Neural Networks (IJCNN)*, 2217–2226 (IEEE, 2017).
56. Habashy, K. G., Evans, B. D., Goodman, D. F. M. & Bowers, J. S. Adapting to time: Why nature evolved a diverse set of neurons. *PLoS Comput. Biol.* https://doi.org/10.1371/journal.pcbi.1012673 (2024).
57. Schrauwen, B. & Van Campenhout, J. Extending spikeprop. In *Proc. International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)*, IJCNN-04 (IEEE, 2002).
58. Gallego, G. et al. Event-based vision: a survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **44**, 154–180 (2020).
59. van Schaik, A. & Liu, S.-C. Aer ear: a matched silicon cochlea pair with address event representation interface. In *Proc. International Symposium on Circuits and Systems (ISCAS)*, 4213–4216 (IEEE, 2005).
60. Bartolozzi, C., Glover, A. & Donati, E. Neuromorphic sensing, perception and control for robotics. in *Handbook of Neuroengineering*, 1–31 (Springer, 2021).
61. Cramer, B. et al. Surrogate gradients for analog neuromorphic computing. *Proc. Natl. Acad. Sci. USA.* **119,** e2109194119 (2022).
62. Göltz, J. et al. Gradient-based methods for spiking physical systems. In *Proc. International Conference on Neuromorphic, Natural and Physical Computing (NNPC)* (NNPC, 2023).
63. Cramer, B., Stradmann, Y., Schemmel, J. & Zenke, F. The heidelberg spiking data sets for the systematic evaluation of spiking neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **33**, 2744–2757 (2022).
64. Warden, P. Speech commands: a dataset for limited-vocabulary speech recognition. 1804.03209. Preprint at https://doi.org/10.48550/arXiv.1804.03209 (2018).
65. Müller, E. et al. Extending BrainScaleS OS for BrainScaleS-2. 2003.13750. Preprint at https://doi.org/10.48550/arXiv.2003.13750 (2020).

Acknowledgements

We want to thank the EIS-Lab, NeuroTMA, CompNeuro, and ElectronicVision(s) groups, in particular Yannik Stradmann, Robin Heinemann, Joscha Ilmberger and Eric Müller, for the continuing support. Additionally, we are grateful to the NNPC conference 2023, where this fruitful collaboration was initialized, as well as the CapoCaccia workshop, where this work was first presented and received much helpful feedback, especially from Paolo Gibertini and Maryada. We also thank Guillaume Bellec for critical comments on the preprint and Florent Draye for providing feedback on the mathematical proofs. The presented work has received funding from the Manfred Stärk Foundation (JG, MAP), the EC Horizon 2020 Framework Programme under grant agreement 945539 (HBP; JG, LK, SB, PL, JS, MAP) and Horizon Europe grant agreement 101147319 (EBRAINS 2.0; MAP), the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy EXC 2181/1-390900948 (Heidelberg STRUCTURES Excellence Cluster; JG, MAP), Swiss National Science Foundation Starting Grant

Project UNITE (TMSGI2-211461; JW, LK, SB, MP), and the Volkswagen-Stiftung under grant number 9C840 (SB).

Author contributions

JG, JW, and LK jointly developed the theory, designed the experiments, implemented the code, executed simulation and hardware experiments; SB, PL, and JS contributed to the hardware experiments; MP and MAP supervised the project, contributed to the experiment design, and provided helpful guidance throughout; JG, JW, LK, MP, and MAP wrote and revised the 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/s41467-025-63120-y.

Correspondence and requests for materials should be addressed to Julian Göltz, Jimmy Weber, Laura Kriener, Melika Payvand or Mihai A. Petrovici.

Peer review information *Nature Communications* thanks the anonymous reviewers for their contribution to the peer review of this work. A peer review file is available.

DelGrad: Exact event-based gradients for training delays and weights on spiking neuromorphic hardware

Supplementary Information

SI.A Additional simulation results

SI.A.1 Deeper networks

DelGrad maintains its performance when scaling to deeper networks. Figure SI.1 shows a series of networks including axonal delays with increasing depth (from 1 to 5 hidden layers) and fixed width (7 neurons per hidden layer), demonstrating a consistent decrease in test error. The baseline configuration with 1 hidden layer of 30 neurons falls between the 4-layer and 5-layer networks, both in terms of test error and number of parameters.

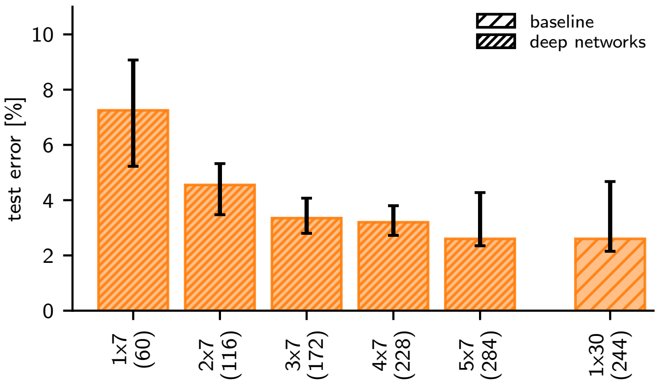

Figure SI.1: **Test error for networks of varying depth.** Networks of the form $l \times n$, with l the number of hidden layers of n neurons. The number of parameters is indicated in parentheses. The test error decreases from a depth $l = 1$ to $l = 5$ and constant width of $n = 7$. The 1×30 baseline lies between the 4×7 and 5×7 configurations.

SI.A.2 Ablation studies

In addition to the results presented in the main text we performed several ablation studies that illustrate the effect of trainable delays in our networks. Figure SI.2b demonstrates that the training makes use of all available resources to improve the task performance by comparing fully trained networks with networks where weight and delay training is disabled either for the input-to-hidden layer connections or for the hidden-to-output ones. The fixed parameters are initialized by sampling from a Gaussian distribution that approximates the empirical distribution of weights and delays observed in a fully trained and optimized baseline network. This ensures that the parameters lie in an appropriate range to solve the task. On the one hand, this shows that the training of all layers is required to obtain optimal performance on the task. On the other hand, it also proves that in the full training, useful gradients are provided to all parameters in all layers.

Finally, we compare the performance achieved when we train weights and delays to an approach similar to the one employed in [1], where weights are trained, but the delays are fixed and random (Fig. SI.3). The mean and standard deviation for configuring the delays are obtained from a hyperparameter search on a wide range of values. We see that for this task, training the delays compared to just providing a random selection that is not adjustable provides a clear advantage for all delay types and especially in smaller networks.

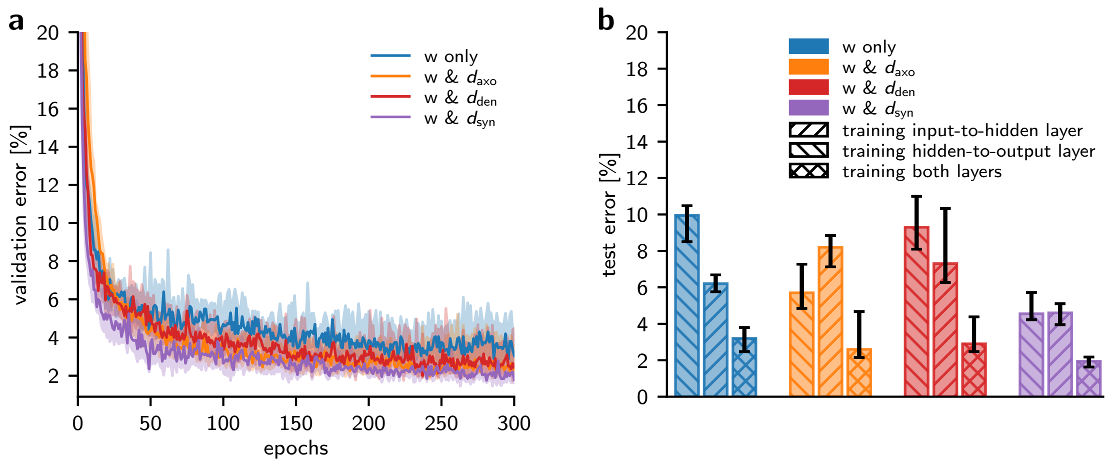

Figure SI.2: **Extended simulation results. a)** Comparison of the validation error during training for the different delay types and a network without delays. All networks have one hidden layer with 30 neurons. **b)** Ablation study (for networks with a hidden layer size of 30 neurons) on the effect of only training the connections between input and hidden layer or between hidden and output layer.

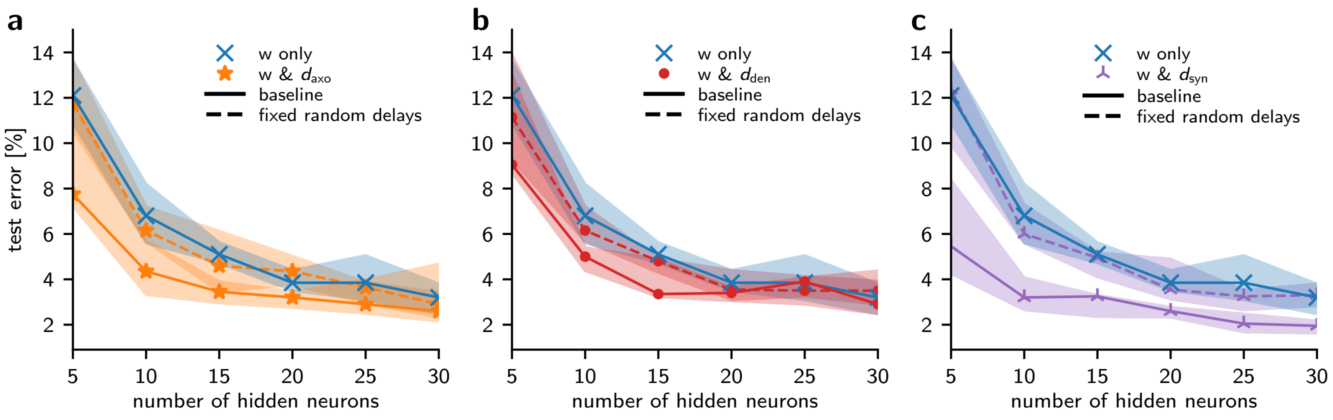

Figure SI.3: **Comparison to random but fixed delays.** For a setup of axonal (a), dendritic (b) or synaptic (c) delays the performance of fully trained networks (i.e. learning weights and delays) are compared to networks with random but fixed delays (dashed lines) and networks without delays (blue).

SI.A.3 Hardware-aware software simulations

The training of spiking neural networks (SNNs) on mixed-signal neuromorphic hardware brings several additional challenges compared to an ideal software simulation. These challenges include, among others, limited resolution and ranges on parameters such as the synaptic weights, fixed-pattern noise and trial-to-trial variability. The impact of these factors on the final outcome of the training difficult to predict and disentangle. To nevertheless get an impression of this, we attempt to mimic these effects in a software simulation. We call this hardware-aware training. With the hardware-aware simulation we can perform an ablation-study that allows us to step-by-step include more levels of hardware-realism and observe the effect on performance. We ensure that the noise levels and restrictions that we include in the simulation are of similar magnitude than what is encountered on a mixed-signal platform. This of course strongly depends on the choice of neuromorphic system and we show it here for our configuration of BSS-2.

Weight ranges and quantization The synaptic weights on BSS-2 have a 6 bit-resolution. To model this in software we use the same approach as described in detail in the methods of [2]: We match the available maximal postsynaptic potential (PSP) height in simulation and on hardware and then train with quantization-aware training within the available range.

Due to the limited weight range, the maximum achievable weight is constrained, requiring, for our chosen neuron

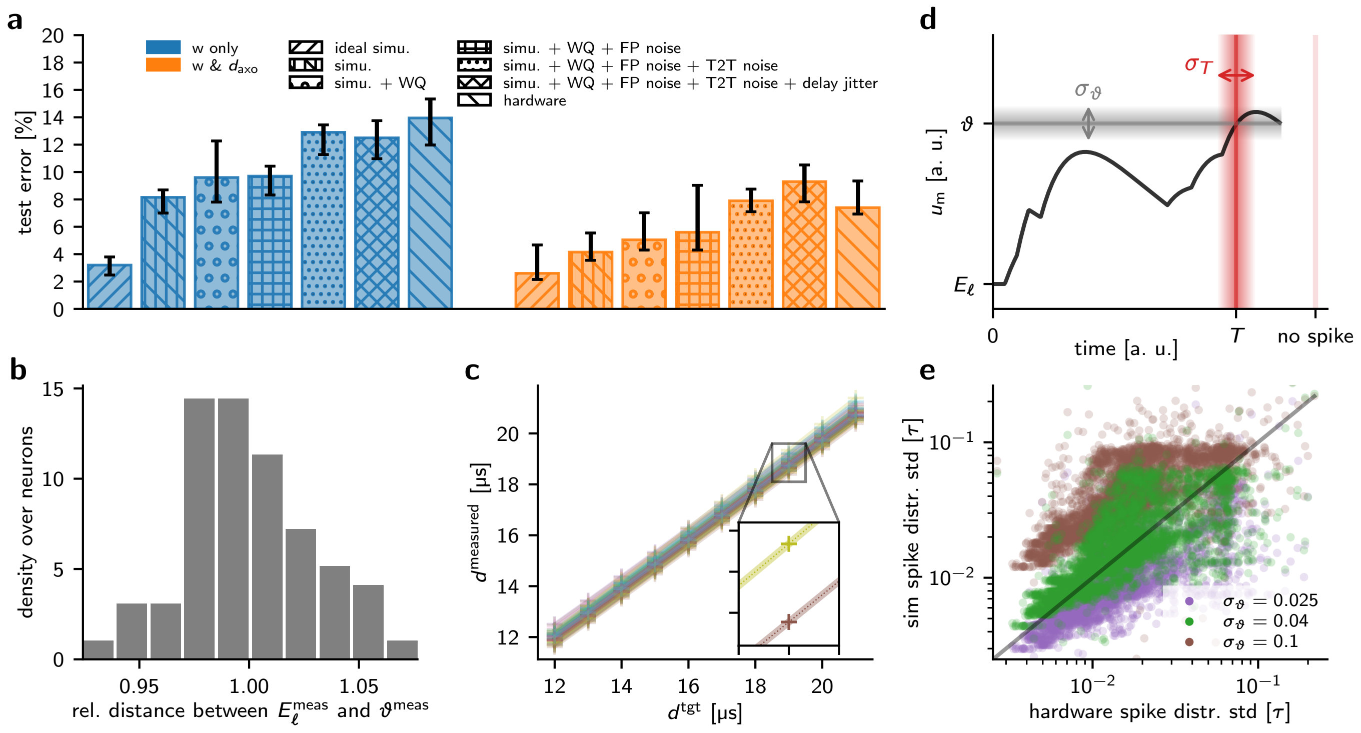

Figure SI.4: **Hardware-aware simulation and noise measurements a)** Ablation study to disentangle the impact of different hardware effects on training performance. In addition to the ideal software simulation (leftmost bar) and the actual hardware result (rightmost bar), it shows hardware-aware simulation results with progressively more hardware effects (from left to right): weight quantization (WQ), fixed-pattern noise (FP), trial-to-trial noise (T2T) and delay jitter. In principle, for each setting slightly different hyperparameters would be optimal; however, for a more direct comparison within reasonable simulation time, we've selected the set of hyperparameters that have proven reliable in the actual hardware emulation for all experiments starting from the second bar (simu.). Bar height indicates median test error and the error bars show the interquartile range (IQR). The values shown correspond to Table SI.1. **b)** Histogram of the normalized relative distance between the measured threshold ϑ and the leak potential E_ℓ. This distribution is used to estimate the fixed-pattern noise between different neuron circuits on the hardware. **c)** Measurements of delays produced by parrot neurons on BrainScaleS-2 (BSS-2) over the corresponding target values used to configure the parrots. **d)** Sketch of the non-linear relationship between variations in the threshold (gray area) and the resulting variations in the output spike timing (red area). **e)** Comparison of variations on the output spike timing on hardware and in the hardware-aware simulation for different assumed trial-to-trial noise magnitudes σ_ϑ. Each point represents the variations on the spike time for one neuron that was presented multiple times with the same input pattern. The plot combines data for multiple neurons and different input patterns.

parametrization, multiple input spikes to reliably trigger an output spike in a neuron. To address this, as described in [2], we replicate each input spike across five channels. This approach effectively increases the maximum achievable weight by a factor of five, a technique we refer to as "channel multiplexing". While these multiplexed channels introduce additional parameters to the network, in the weight-only training they do not enhance its computational capacity. This is because all multiplexed channels share the same input and hence, for each input spike the actual input into the hidden layer is simply the sum of the weights across all channels. In contrast, when delays are available, each channel can have a unique delay, resulting in input spikes arriving at different times. These staggered spike timings separate the influence of each channel, allowing the individual weights to have a distinct effect and thereby increasing the computational capacity of the delay-based network compared to the weight-only ones. However, since this computational advantage arises solely from hardware limitations, we enforce shared delays across multiplexed channels in the delay-based network to ensure a fair comparison.

For the small networks with a hidden layer size of only 5 neurons we encounter the same problem of having too few spiking neurons in the hidden layer to reliably activate the output layer. On BSS-2 we can solve this by using two synapse circuits instead of one for a connection, effectively doubling the weight (the delays here are shared automatically, because the spikes are received from a parrot neuron). This requires more chip resources per connection, which is why typically this is avoided, but for the smallest networks it proved to be necessary. We mimic this in the hardware-aware simulation, for the hidden layer size of 5, by increasing the available weight range by a factor of 2.

Fixed-pattern noise Fixed-pattern noise, or sometimes called frozen noise, is caused by imperfections in the chip fabrication process and causes slight differences between the neuron circuits. This results in the dynamics of each neuron on chip to differ slightly. These differences between neurons are static over time. The effects of fixed-pattern noise can partially be mitigated using calibration procedures, which we employ, but some variability between the neurons remains. Neuron parameters that are affected by fixed-pattern noise are for example the time constants, the strength of the synaptic input, the resting potential and the threshold. As a simplification for our simulations we do not model all sources of fixed-pattern noise individually, but summarize them all into one source. The variation between neurons of the difference between resting potential and threshold is easy to measure on chip (Fig. SI.4b). Therefore, in our simulations, we model the fixed-pattern noise based on this parameter. In the hardware-aware simulations, at network initialization, a random offset to the threshold of each neuron is drawn from a Gaussian distribution and applied for the whole training procedure. For an estimate on the variance of the Gaussian, we use the variance observed in Fig. SI.4b and increase it slightly to account for the other sources of fixed-pattern noise.

Trial-to-trial variability In addition to fixed-pattern noise, which is static over time, we also observe trial-to-trial variability on the hardware. Electronic circuits are affected by temporal noise on all timescales. High frequency components become apparent as visible jitter on top of the underlying signal, for example on membrane traces. Lower frequency components, in contrast, occur also on timescales often greater than individual observation periods and can thus manifest themselves as pseudo-static offsets of a signal on a trial-to-trial basis. These low frequency components are particularly strong due to the fact that the noise characteristics of electronic devices are typically dominated by flicker noise with a spectral density, or "amplitude", proportional to $1/f$.

The resulting trial-to-trial variability can be interpreted as random fluctuations of neuron parameters on comparatively long time scales: We model it by varying the neuron parameters between experiments (i.e. different batches) but assume them to stay constant within experiments. Most noise sources – such as the leak and threshold terms but also the synaptic integrators and input circuits – eventually affect the neuronal membrane state in form of a random offset and we thus subsume all of them in a variation of the distance between leak and threshold potentials. Therefore, if the same neuron on the same chip is presented with the same input in different batches, its output spike time will differ slightly. The relationship between a variation on the threshold and the resulting variations on the spike times of the neuron is highly non-linear (Fig. SI.4d). This makes the trial-to-trial variability hard to model directly on the spike times, even though this is where it is observed. Instead, we choose to add random offsets to each threshold of every neuron for every batch, which automatically then approximates the trial-to-trial noise observed on the spike times on the hardware. To confirm this, we repeatedly present the same batches of samples to the hardware and record all occurring spikes. Then we present the same samples repeatedly to the hardware-aware simulation, where for each batch and neuron, we add a random sample as the offset to the threshold, drawn from a Gaussian centered around zero and with width σ. For the right choice of σ we observe that the variances, observed for each sample over many repetitions, match well between hardware-aware simulation and experiments (Fig. SI.4e).

Delay effects Figure SI.4c provides an estimate of how accurately our parrot neuron setup reproduces a target delay. While the variations across multiple trials are minimal, we account for them in the hardware-aware simulations. Specifically, we model the variability of the delay circuits by introducing Gaussian noise ($\sigma = 0.01\,\tau$) to the output spike times of the delay layers. In general, spike signal communication on BSS-2 is highly reliable, and spike loss only becomes a concern at very high firing rates within the network. However, the inclusion of parrot neurons for delay implementation introduces a potential new source of spike loss. We measured the average rate of spike loss caused by the parrot neuron setup and found it to be negligibly small. Consequently, this effect is not incorporated into our simulations.

Results With the hardware-aware simulation setup described above we perform an ablation study and compare the results to the actual hardware emulation as well as the ideal software simulation (Fig. SI.4a and Table SI.1).

Some increase in error is introduced when transitioning from the hyperparameters optimized for the ideal simulations to those used on our hardware. This adjustment was necessary because the optimal parameters identified by software-based Hyperparameter Optimization (HPO) did not perform well on the hardware. In software simulations, hyperparameters were optimized independently for each network size and delay configuration (e.g., axonal or synaptic). However, such extensive HPO is impractical on hardware due to the inability to parallelize runs on the chip, making the process prohibitively time-consuming. To address this, we optimized a single set of hyperparameters that performs reasonably well across both weight-only and axonal delay scenarios and for all network sizes. While this compromise set does not achieve the same performance as the highly optimized software case, it is a good trade-off between the feasibility and performance. More importantly, as shown in Fig. SI.4a and Table SI.1, all modeled noise sources contribute to the increased error, and the final error in hardware-aware training matches closely with the results observed on the chip. This demonstrates that, despite significant simplifications in modeling hardware effects, our measurement-based estimation of noise sources effectively captures the chip's general behavior.

Table SI.1: **Estimation of the impact of different hardware phenomena on training results using the hardware-aware simulation framework.** The values given are the median test errors with the IQR in parentheses.

	ideal simulation	simulation (HW params)	simulation (HW params) + weight quant.	simulation (HW params) + weight quant. + FP noise	simulation (HW params) + weight quant. + FP noise + trial-to-trial	simulation (HW params) + weight quant. + FP noise + trial-to-trial + delay jitter	hardware
weights + axonal delays	$2.60^{4.67}_{2.15}$ %	$4.15^{5.55}_{3.55}$ %	$5.05^{7.03}_{4.30}$ %	$5.60^{9.03}_{4.30}$ %	$7.90^{8.75}_{7.10}$ %	$9.30^{10.52}_{7.83}$ %	$7.40^{9.35}_{6.93}$ %
weights	$3.20^{3.80}_{2.48}$ %	$8.15^{8.70}_{7.00}$ %	$9.60^{12.27}_{7.80}$ %	$9.70^{10.42}_{8.33}$ %	$12.50^{13.75}_{10.97}$ %		$13.95^{15.34}_{11.97}$ %

SI.A.4 Comparison to literature

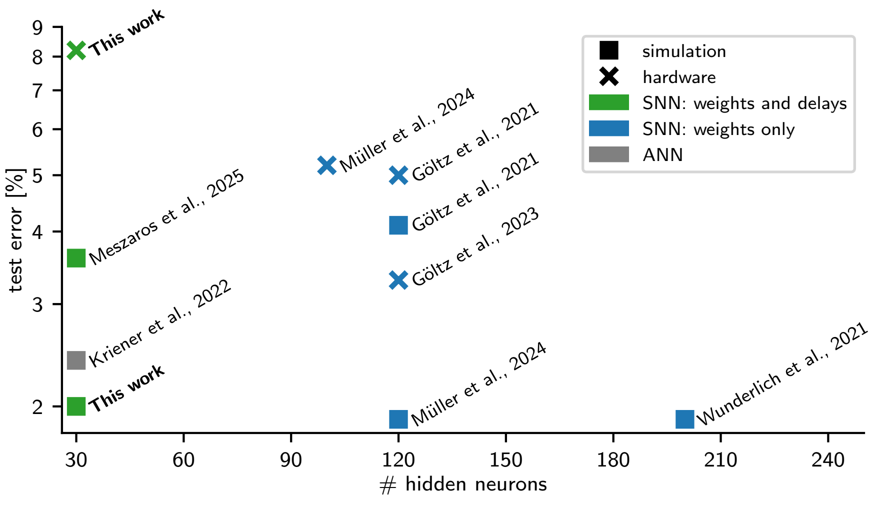

Figure SI.5: **Comparison to other results on the Yin-Yang (YY) dataset.** Each data point represents the mean test accuracy achieved on the YY dataset for a certain size of the hidden layer. Simulation results are marked as squares while hardware results are plotted as crosses. Results employing delays are colored in green, while weight-only networks are blue. The data is collected from the following publications: Meszaros et al., 2025 [3]; Kriener et al., 2022 [4]; Müller et al., 2024 [5]; Göltz et al., 2021 [2]; Göltz et al., 2023 [6] and Wunderlich et al., 2021 [7].

SI.B Hardware implementation and additional results

As BSS-2 does not include dedicated circuitry for emulating delays, we re-purpose neuron circuits to act as delay elements (parrot neurons, see Fig. 5a). For any incoming spike the parrot neuron is configured to produce an output spike with a certain controllable delay, as described below.

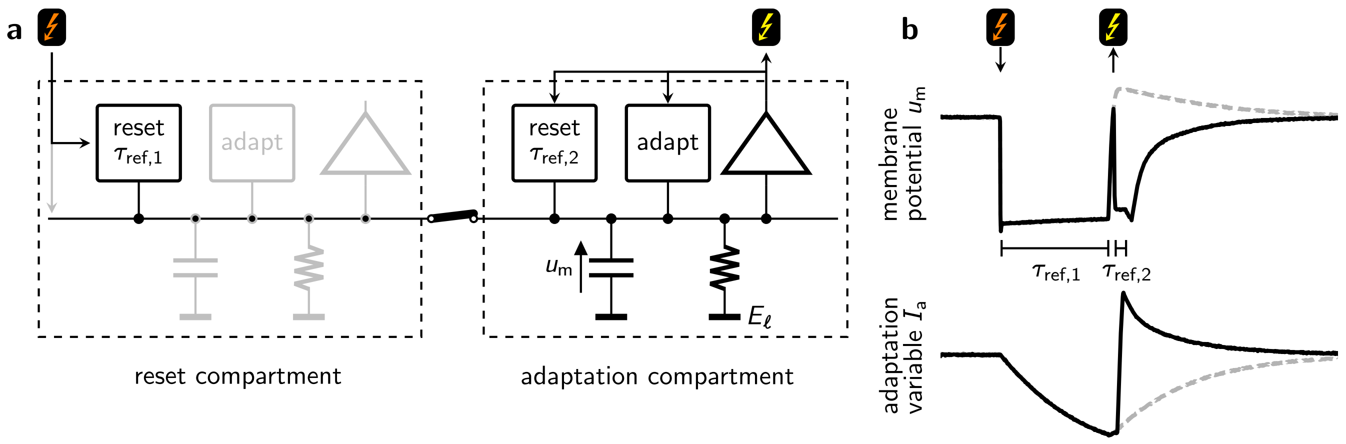

Figure SI.6: **Schematic illustration of the parrot neuron used to implement delays on BrainScaleS-2. a)** Sketch of the two neuron compartments (dashed boxes) used for the delay mechanism. The reset and adaptation circuits are drawn as rectangles, the threshold comparator as triangles. Disabled components are drawn in gray. Lightning bolts indicate incoming (orange) and outgoing (yellow) spikes. **b)** Recorded traces of membrane voltage and adaptation when the parrot neuron receives an input spike and produces a delayed output spike. The dashed gray traces show the circuit with disabled threshold comparator in the second compartment.

SI.B.1 Axonal delays using AdEx and multi-compartment functionality

To obtain the results shown in the main text the delays are implemented using the multi-compartment and the adaptation functionality [8] of the BSS-2 hardware [9, 10]. Figure SI.6 illustrates the mechanism that consists of two separate neuron compartments, from here on called reset and adaptation compartment, corresponding to their respective role. The membrane capacitance of the reset compartment is disabled and the two compartments are short-circuited such that they share a membrane voltage. The dynamics of the adaptation compartment are described by the equations of the adaptive exponential leaky integrate-and-fire (AdEx) model with disabled exponential component:

$$\tau_\text{m}\dot{u}_\text{m}(t) = [E_\ell - u_\text{m}(t)] + I_\text{s}(t)/g_\ell - I_\text{a}(t)/g_\ell \tag{SI.1}$$

$$\tau_\text{a}\dot{I}_\text{a}(t) = a(u - E_\ell) - I_\text{a}(t) \tag{SI.2}$$

where the adaptation variable I_a is a leaky integrator that is driven by the difference between the membrane voltage and the leak potential. Note that no synaptic input is connected to this compartment, and thus $I_\text{s}(t)$ is zero at all times. When $u_\text{m}(t)$ crosses the spiking threshold at t_spike, the membrane voltage is reset and clamped to V_reset for the duration of the refractory period $\tau_{\text{ref},2}$ and spike-triggered adaptation causes a jump on the adaptation variable

$$u(t) = V_\text{reset} \quad \forall t \in (t_\text{spike}, t_\text{spike} + \tau_{\text{ref},2}] \tag{SI.3}$$

$$I_\text{a} \to I_\text{a} + b \tag{SI.4}$$

where b is the parameter controlling the magnitude of the spike-triggered adaptation.

An input spike arriving at the reset compartment of the parrot neuron triggers an immediate reset which clamps the membrane potential (shared between both compartments) to the low reset potential V_reset for the duration of $\tau_{\text{ref},1}$. Since V_reset is lower than E_ℓ, the magnitude of the adaptation current I_a in the adaptation compartment builds up during the refractory period. Once the refractory period $\tau_{\text{ref},1}$ ends, the membrane voltage, now driven by the adaptation variable, can evolve freely away from V_reset and the accumulated strong adaptation current I_a causes the membrane voltage to rapidly increase towards the threshold. This phenomenon, although commonly caused not by a low V_reset but by inhibitory input, is known as inhibitory rebound: The membrane potential overshoots the leak potential and becomes high enough to reach the spiking threshold of the adaptation compartment and produce an output spike, triggering the reset mechanism of this compartment. Additionally, the spike triggers the spike-triggered adaptation mechanism of the AdEx model, which causes a jump on I_a by b, bringing back the adaptation I_a close to zero. Finally, the refractory period of the adaptation compartment $\tau_{\text{ref},2}$ is configured to be very short and the membrane can almost instantly relax to the leak potential again. After this, the parrot neuron is ready to receive and delay the next input spike.

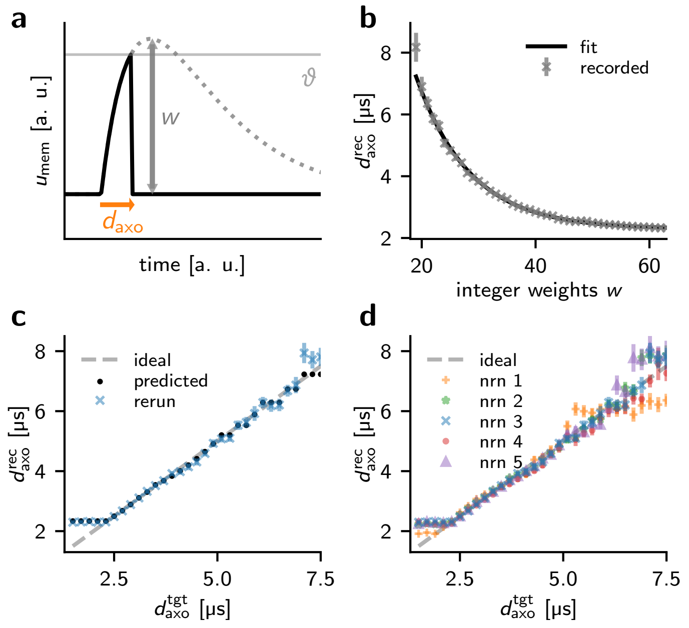

Figure SI.7: **Proof of concept for implementing on-chip axonal delays using only LIF dynamics on BrainScaleS-2.** **a)** Example membrane trace of a parrot neuron where the rise time of the PSP causes a delay of its output spike with respect to the output spike time of its afferent network neuron. This delay can be configured through an appropriate choice of the synaptic weight between network neuron and parrot neuron. **b)** Example BrainScaleS-2 recording of the relationship between the measured input-output delay of a parrot neuron d_{rec} and its afferent synaptic weight (neuron 3 in d)). Mean and standard deviation are shown over 10 runs with 50 spike pairs each. An exponential fit (black) yields the calibration curve for the weight-delay relationship. **c)** Test of the calibration for the same parrot neuron as in b). The calibration curve from the fit in d) is used across a range of target delays $d_{\mathrm{axo}}^{\mathrm{tgt}}$ to determine the corresponding optimal synaptic weights. With this weight the delay is re-measured for 5 runs with 50 spike pairs each, checking the deviation between the predicted delay (black) and the actual recorded delay d_{rec} (mean and standard deviation in blue). **d)** Same as c) but for 5 different parrot neurons on the chip to illustrate the variability between different neuron circuits.

As the inhibitory rebound causes an output spike immediately after the end of the refractory period of the reset compartment, the delay of this parrot neuron can be configured via $\tau_{\mathrm{ref},1}$. This is advantageous, as τ_{ref} is an 8 bit digital parameter on BSS-2 and can be configured to a large range of possible refractory periods without fixed pattern noise. As shown in Fig. SI.4c, the delays can be reliably configured in the range of 12 µs to 21 µs. In principle, the available delay range is larger (approximately 8 µs to 35 µs), but since we do not require such a large range of delays for our experiments and the values in the middle of the available range are the most stable, we restrict ourselves to what is shown in Fig. SI.4c. Note that having the smallest available delay unequal to zero is not computationally relevant, as long as this minimal value is equal for all neurons.

SI.B.2 Proof-of-concept for axonal delays using only LIF dynamics

Since leaky integrate-and-fire (LIF) neuron models are more widely available on various neuromorphic platforms compared to the multi-compartment AdEx model, we also present a proof-of-concept of how LIF models can be adapted for implementing analog on-chip delays, ensuring our results are more easily reproducible.

Setup The network setup in this method is similar to the previous section in that each network neuron has a corresponding parrot implementing the delay; However, the method of creating this delay is different: We leverage the fact that due to the finite rise time of the PSP on the parrot's membrane voltage, its spike is delayed compared to the one of the network neuron (Fig. SI.7a). The magnitude of this delay, which emulates the axonal delay of the network neuron, depends on several parameters, such as the synaptic weight w of the connection between the network and parrot neurons, the time constants $\tau_{\mathrm{s}}, \tau_{\mathrm{m}}$ and the difference between threshold and leak potential of the parrot neuron. For a theoretical description of the relationship between the parrot's delay and the synaptic weight see Section SI.F.

For our implementation of this scheme on BSS-2, we control the delay solely via the synaptic weight w, keeping the time constants and potentials fixed: while those parameters can be individually tuned, that (analog) configuration is slower compared to the (digital) weight setting. Since in our trained networks on BSS-2 ([2] and Fig. 5) we use neuron time constants of $\tau_\mathrm{s} = \tau_\mathrm{m} \approx 6\,\mu\mathrm{s}$, we aim to reach delays of the same magnitude here. To achieve this, we configure the parrot neurons to have a synaptic time constant of $\tau_\mathrm{s} = 10\,\mu\mathrm{s}$ and a membrane time constant of $\tau_\mathrm{m} = 15\,\mu\mathrm{s}$. A long refractory time of $16\,\mu\mathrm{s}$ ensures that each input to the parrot only triggers one output spike.

During the training of a network, it is required to reconfigure the parrot neurons on the chip to produce the correct axonal delays $d^\mathrm{tgt}_\mathrm{axo}$. For this, the relationship between the synaptic weight w and the produced delay $d^\mathrm{rec}_\mathrm{axo}$ needs to be measured. Due to the usual variations in the manufacturing process (fixed-pattern noise), the on-chip analog neuron circuits are not exactly identical to each other. Therefore, for every parrot neuron, we perform a separate calibration measurement in order to determine the precise mapping between the weight parameter and the recorded delay individually. To this end, we configure a range of different weights and record the resulting delays $d^\mathrm{rec}_\mathrm{axo}$ (Fig. SI.7b for an example neuron). The full available weight range from 0 to 63 is not used, as for the lower weights, the parrot neuron does not reliably produce output spikes. To include both temporal drift during one trial and trial-to-trial variations, we record delays during 10 runs with 50 pairs of input and output spikes and average the results.

We fit an exponential function $d(w) = \alpha + \beta \exp(\gamma(w + \delta))$ to the measured data. The inverse of the fit function thus determines the relation between the target delay $d^\mathrm{tgt}_\mathrm{axo}$, and the optimal integer weight to configure on the chip. To test the quality of the weight-delay fit, it is used to configure the chip for a whole range of target delays $d^\mathrm{tgt}_\mathrm{axo}$, while measuring the actual value of the delays produced on the chip $d^\mathrm{rec}_\mathrm{axo}$ (Fig. SI.7c). The above process is repeated for 5 different neuron circuits on the chip, and the results are compared to illustrate the impact of fixed-pattern noise in Fig. SI.7d.

Results Figure SI.7c shows the desired correspondence between the target, $d^\mathrm{tgt}_\mathrm{axo}$, and the recorded, $d^\mathrm{rec}_\mathrm{axo}$, on-chip delay values, especially in the intermediate delay range. This underpins the feasibility of our proposed approach. We note a plateau in the recorded delay values for lower targets, caused by the maximum possible on-chip weight value, corresponding to the shortest possible delay. Additionally, a larger deviation and more instability is observed for larger target delays; this effect has two causes: a worse quality of the exponential fit and an increased instability of the threshold crossing in the region where the PSP plateaus. Such increasing instability is unavoidable in analog neurons, as any amount of noise on the membrane voltage results in increased trial-to-trial variability when the peak of the PSP is close to the threshold.

Although each neuron can be configured individually to produce the desired behavior, there is still some variability between the neurons. However, we do not expect these variations to be harmful in practice; in fact, some heterogeneity between neurons behavior might even be beneficial, as has been shown in [11].

While the presented idea can be feasible for small networks and tasks, it is clearly suboptimal, as it requires a portion of the available neuron circuits to be used as delay elements instead of their usual role in the network. Additionally, several practical considerations have to be taken into account when this setup is included in the training of a full network. First, the range of achievable delays is limited by the time constants of the parrot neurons. In our experiments we targeted a delay range of approximately $6\,\mu\mathrm{s}$ which corresponds to the delay range used in the simulation results. Second, for a correct delay on an incoming spike, the parrot neuron's membrane voltage and synaptic currents need to be at their resting values. Therefore, the interval between spikes arriving at the parrot neuron needs to be large enough, which can be ensured by increasing the refractory time of the neurons in the network. However, for tasks where the neurons need short refractory periods to process their input correctly, this method of producing axonal delays is not suitable. Nevertheless, these results provide a proof of concept that axonal delays can be implemented by repurposing resources and circuits that are universally available on most neuromorphic substrate.

SI.C Complete equation for the time of the first spike

Given a sequence of input spikes $\{t_i\}$ and corresponding weights $\{w_i\}$, we define

$$a_n := \sum_{i \in C} w_i \exp\left(\frac{t_i}{n\tau_\mathrm{s}}\right) \quad \text{and} \quad b := \sum_{i \in C} w_i \frac{t_i}{\tau_\mathrm{s}} \exp\left(\frac{t_i}{\tau_\mathrm{s}}\right) \,. \tag{SI.5}$$

These definitions use the causal set $C = \{i \mid t_i < T\}$ of input spikes before the output. With those definitions, the spike time of a neuron with identical membrane and synaptic time constant $\tau_\mathrm{m} = \tau_\mathrm{s}$ is (Eq. (4) in the main text)

$$T = \tau_\mathrm{s} \left\{ \frac{b}{a_1} - \mathcal{W}\left[-\frac{g_\ell \vartheta}{a_1} \exp\left(\frac{b}{a_1} \right) \right] \right\} , \tag{SI.6}$$

and for $\tau_\mathrm{m} = 2\tau_\mathrm{s}$ (Eq. (5) in the main text)

$$T = 2\tau_\mathrm{s} \ln \left[\frac{2a_1}{a_2 + \sqrt{a_2^2 - 4a_1 g_\ell \vartheta}} \right] . \tag{SI.7}$$

The Lambert W function is defined as the solution $h = \mathcal{W}(z)$ to the equation $z = h \exp(h)$.

SI.D Extending the formalism for multiple spikes

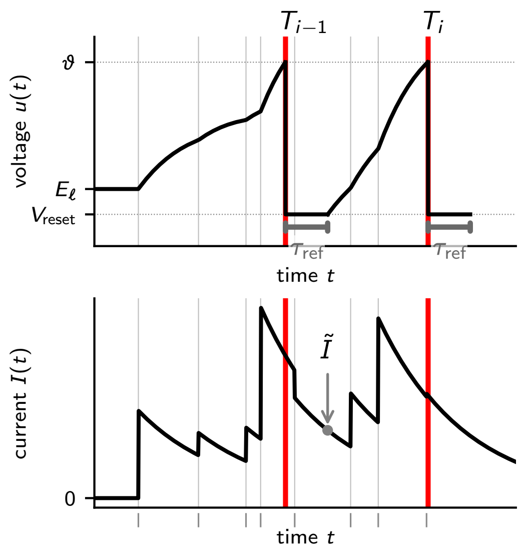

Figure SI.8: **Sketch of the voltage (top) and current (bottom) dynamics of an LIF neuron that is spiking multiple times.** The gray vertical lines indicate the input spikes, the red vertical lines the output spikes. After each spike, the voltage is clamped to the reset potential V_reset for a time τ_ref, highlighted by the gray horizontal bar in the top plot. After this refractory period, the voltage evolves freely again, from $T_{i-1} + \tau_\mathrm{ref}$ on driven by the residual current $\tilde{I}$ and additional input spikes.

In the main text, we have derived our equations in a simplified scenario in which each neuron only spikes once. As shown in the present and earlier work [2], single spikes can already be sufficient for many problems, however, for scenarios in which multiple spikes per neuron are necessary, the extended equations are derived below.

For the first spike of a neuron, Equations (4) and (5) are derived by integrating the ordinary differential equation (ODE) in Eq. (1) to get voltage dynamics $u(t)$, and afterward solving for the time T of the spike, defined by $u(T) = \vartheta$. Recalling the dynamics of the LIF model, after a spike the voltage is fixed at the reset voltage V_reset for a time τ_ref (Fig. SI.8, top panel), and afterward continues to follow the dynamics laid out in the ODE (1). For the time of the second spike T_2 and all further spikes this implies that the same procedure can be followed when adhering to different initial conditions.

For the first spike, the integration of the ODE is performed with initial vanishing current $I(0) = 0$ and voltage $u(0) = 0$ (w.l.o.g. we have chosen leakage $E_\ell = 0$), resulting in voltage dynamics

$$u(t) = \sum_{i \in C} \Theta(t - t_i) \frac{w_i}{g_\ell} \frac{\tau_\mathrm{s}}{\tau_\mathrm{m} - \tau_\mathrm{s}} \left[\exp\left(-\frac{t - t_i}{\tau_\mathrm{m}} \right) - \exp\left(-\frac{t - t_i}{\tau_\mathrm{s}} \right) \right] . \tag{SI.8}$$

Assuming a spike at T_{i-1}, the new initial condition for the voltage can be written down at time $\tilde{t} := T_{i-1} + \tau_\mathrm{ref}$ as

$$u(\tilde{t}) = \tilde{u} := V_\mathrm{reset} , \tag{SI.9}$$

while the current is not affected by the reset, and keeps following Eq. (2). To simplify the notation, we split up the spikes of the causal set $t_i \in C$ into one set arriving before $\tilde{t}$, $C^< = \{t_i \in C | t_i < \tilde{t}\}$, and one set with spikes afterward $C^\geq = \{t_i \in C | t_i \geq \tilde{t}\}$. For times $t \geq \tilde{t}$, we can write the current as

$$I(t) = \tilde{I} \exp\left(-\frac{t - \tilde{t}}{\tau_\mathrm{s}} \right) + \sum_{i \in C^\geq} \Theta(t - t_i) w_i \exp\left(-\frac{t - t_i}{\tau_\mathrm{s}} \right) , \tag{SI.10}$$

with $\tilde{I} = \sum_{i \in C^<} w_i \exp(-\frac{\tilde{t} - t_i}{\tau_\mathrm{s}})$ the (residual) current at time $\tilde{t}$ (see Fig. SI.8). Integration of the ODE under these initial conditions results in an equation for the dynamics of the voltage:

$$\begin{aligned} u(t) = &\sum_{i \in C^\geq} \Theta(t - t_i) \frac{w_i}{g_\ell} \frac{\tau_\mathrm{s}}{\tau_\mathrm{m} - \tau_\mathrm{s}} \left[\exp\left(-\frac{t - t_i}{\tau_\mathrm{m}} \right) - \exp\left(-\frac{t - t_i}{\tau_\mathrm{s}} \right) \right] \\ &+ \tilde{u} \exp\left(-\frac{t - \tilde{t}}{\tau_\mathrm{m}} \right) + \frac{\tilde{I}}{g_\ell} \frac{\tau_\mathrm{s}}{\tau_\mathrm{m} - \tau_\mathrm{s}} \left[\exp\left(-\frac{t - \tilde{t}}{\tau_\mathrm{m}} \right) - \exp\left(-\frac{t - \tilde{t}}{\tau_\mathrm{s}} \right) \right] . \end{aligned} \tag{SI.11}$$

The first term encapsulates the effect of the incoming spikes as before (compare Eq. (SI.8)), while the second line is an exponential decay from the reset to the leak, while accounting for the effect of the residual current. Interestingly, this form of the equation shows that the residual current can be modeled as a virtual spike with weight $\tilde{I}$ at time $\tilde{t}$. Crucially, when starting at leak voltage $\tilde{u} = 0$ and without initial current $\tilde{I} = 0$, the above voltage dynamics Eq. (SI.8) are recovered.

With this equation, the derivation performed in [2] can be followed, i.e., assuming a spike $T_i > \tilde{t}$ defines a causal set $\tilde{C} = \{t_i \in C^{\geq} | t_i < T_i\}$, and using $u(T_i) = \vartheta$ we can write

$$\begin{aligned} 0 = &- \underbrace{\left[\tilde{I}\frac{\tau_\mathrm{s}}{\tau_\mathrm{m}-\tau_\mathrm{s}}\exp\left(\frac{\tilde{t}}{\tau_\mathrm{s}}\right) + \sum_{i\in\tilde{C}} w_i \frac{\tau_\mathrm{s}}{\tau_\mathrm{m}-\tau_\mathrm{s}}\exp\left(\frac{t_i}{\tau_\mathrm{s}}\right)\right]}_{a_1'} \cdot \exp\left(-\frac{T_i}{\tau_\mathrm{s}}\right) \\ &+ \underbrace{\left[\tilde{u}\exp\left(\frac{\tilde{t}}{\tau_\mathrm{m}}\right) + \tilde{I}\frac{\tau_\mathrm{s}}{\tau_\mathrm{m}-\tau_\mathrm{s}}\exp\left(\frac{\tilde{t}}{\tau_\mathrm{m}}\right) + \sum_{i\in\tilde{C}} w_i \frac{\tau_\mathrm{s}}{\tau_\mathrm{m}-\tau_\mathrm{s}}\exp\left(\frac{t_i}{\tau_\mathrm{m}}\right)\right]}_{a_2'} \cdot \exp\left(-\frac{T_i}{\tau_\mathrm{m}}\right) \\ &- g_\ell\vartheta \\ = &- a_1'\exp\left(-\frac{T_i}{\tau_\mathrm{s}}\right) + a_2'\exp\left(-\frac{T_i}{\tau_\mathrm{m}}\right) - g_\ell\vartheta\ . \end{aligned} \tag{SI.12}$$

This equation follows the same structure as in the original derivation [2, Eq. (26-27)] with differently defined parameters a_1' and a_2': comparing with Eq. (SI.5), in the case of $\tau_\mathrm{m} = 2\tau_\mathrm{s}$ we find

$$\begin{aligned} a_1 \to a_1' &:= a_1 + \tilde{I}\exp\left(\frac{\tilde{t}}{\tau_\mathrm{s}}\right) \\ a_2 \to a_2' &:= a_2 + \tilde{I}\exp\left(\frac{\tilde{t}}{\tau_\mathrm{m}}\right) + \tilde{u}\exp\left(\frac{\tilde{t}}{\tau_\mathrm{m}}\right)\ . \end{aligned} \tag{SI.13}$$

Notably, this implies that one can solve for T_i to get a function

$$T_i\left(\{t_i\} \cup \{w_i\}, \tilde{I}, \tilde{t}\right)\ . \tag{SI.14}$$

This function is at the heart of implementing exact, event-based forward dynamics and, more importantly, its differentiability enables error backpropagation through multiple layers of such neurons. Through $\tilde{I}$ there is a dependence of the output spike times T_i on earlier input spike times $t_i < \tilde{t}$, and through $\tilde{t}$ a dependence on the previous spike of the same neuron. For the case of $\tau_\mathrm{m} = \tau_\mathrm{s}$, one can use l'Hôpital's rule and the Lambert W function to write down the solution for T_i.

SI.E Explicit, event-based gradients of a voltage-max-over-time loss

Typically, when using an event-based framework, information in a network is encoded in spike times. For some tasks, losses based on the voltage of (a subset) of neurons have been proposed and used, especially the max-over-time loss (for a selection, see [6, 7, 12–14]). For this, the corresponding loss function depends on the correct class $n^\star$ and the voltage $\mathbf{u}(t)$ of the (non-spiking) label neurons together with a scale factor a_scale like

$$\mathcal{L}_{\mathrm{MOT}}[\mathbf{u}(t), n^\star; a_\mathrm{scale}] = -\log\left[\mathrm{softmax}_{n^\star}(a_\mathrm{scale}\cdot \max_t \mathbf{u}(t))\right]\ . \tag{SI.15}$$

Because this loss has been predominantly used with surrogate gradients and depends on the membrane voltage of the label neurons, it is not typically associated with exact and event-based training schemes, with [7, 14] being the exceptions. However, the loss is compatible with a purely spike-based formulation and in the following the relevant gradient will be derived:

$$\frac{\partial u_\mathrm{max}}{\partial\theta} = \frac{\partial u(t|\{\theta\})}{\partial\theta}\bigg|_{t=\tilde{t}} + \frac{\partial u(t|\{\theta\})}{\partial t}\bigg|_{t=\tilde{t}} \cdot \frac{\partial\tilde{t}}{\partial\theta}\ . \tag{SI.16}$$

In addition to the natural first term, the second term occurs when an (inhibitory) input spike determines the maximum of the voltage.

For the derivation, we assume the voltage u is a function of the parameters input spikes $\{t_i\}$ and weights $\{w_i\}$, here shortened as $\{\theta\}$

$$u = u(t|\{t_i\} \cup \{w_i\}) = u(t|\{\theta\}) \,, \tag{SI.17}$$

the maximum voltage $u_{\max}$ at time $\tilde{t}$ is defined by

$$\begin{aligned} \tilde{t} :=& \quad \underset{t\in S}{\arg\max} \quad u(t|\{\theta\}) \\ u_{\max} =& \quad \max_{t\in S} \quad u(t|\{\theta\}) = u(\tilde{t}|\{\theta\}) \,, \end{aligned} \tag{SI.18}$$

where $t \in S$ runs in the integration domain S. This definition makes $u_{\max}$ an implicit function[1] of the parameters $\{\theta\}$. Because the voltage of these (non-spiking) neurons is continuous, the maximization can be written in an integral form using the Dirac δ distribution

$$u_{\max} = \max_t u(t|\{\theta\}) = u(\tilde{t}|\{\theta\}) \tag{SI.19}$$

$$= \int_S \mathrm{d}t\, u(t|\{\theta\})\delta(t-\tilde{t}) \,, \tag{SI.20}$$

which will allow the reformulation in simple, event-based terms.

We split up the voltage along the spike times t_i, cf. Fig. SI.9. Here, w.l.o.g. we assume the input spikes $\{t_i | i \in [1, N]\}$ to be ordered $t_i < t_{i+1} \forall i$ and define $t_0 = -\infty$ and $t_{N+1} = \infty$ as well as the intervals $S_i = (t_i, t_{i+1}]$. Further, we want to define the shorthand $\tilde{S}$ for the interval $\tilde{\imath}$ that contains the maximum $\tilde{S} := S_{\tilde{\imath}} \ni \tilde{t}$, therefore $t_{\tilde{\imath}}$ is the last input spike causal to the dynamics in this interval and there is no new spike within the interval. Employing as a shorthand the synaptic interaction kernel

$$\kappa(t) = \Theta(t)\frac{\tau_\mathrm{s}}{\tau_\mathrm{m} - \tau_\mathrm{s}}\left[\exp(-t/\tau_\mathrm{m}) - \exp(-t/\tau_\mathrm{s})\right] \,, \tag{SI.21}$$

the voltage behaves like $u(t|\{\theta\}) = \frac{1}{g_\ell}\sum_i^N w_i\kappa(t-t_i)$, and one can write

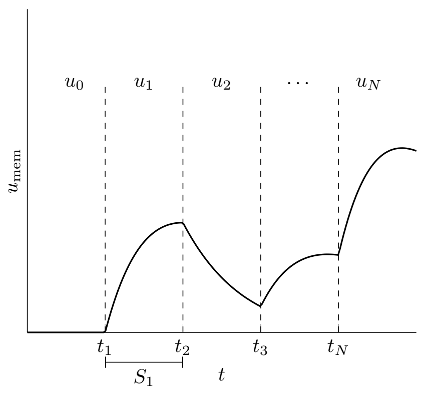

Figure SI.9: Separating u into distinct, smooth u_i at the input spike times t_i.

$$u(t|\{\theta\}) = \frac{1}{g_\ell} \cdot \begin{cases} 0 & \text{if } t \le t_1 \\ w_1\kappa(t-t_1) & \text{if } t_1 < t \le t_2 \\ w_1\kappa(t-t_1) + w_2\kappa(t-t_2) & \text{if } t_2 < t \le t_3 \\ \vdots & \vdots \\ u_n(t|\{\theta\}) & \text{if } t_n < t \le t_{n+1} \end{cases} \,, \tag{SI.22}$$

with $u_n(t|\{\theta\}) = \frac{1}{g_\ell}\sum_i^n w_i\kappa(t-t_i)$. This function $u_n(t)$ is time-differentiable everywhere on its domain S_n. The separation into u_n can be carried over to all integrals of the voltage with any function f

$$\int_S \mathrm{d}t\, u(t|\{\theta\})\left[f(t)\right] = \sum_i \int_{S_i} \mathrm{d}t\, u_i(t|\{\theta\})\left[f(t)\right] \,. \tag{SI.23}$$

Specifically, this can be done for the integral formulation of $u_{\max}$ Eq. (SI.20). For the next step, a requirement is the derivative of an integral with varying boundaries:

$$\frac{\partial}{\partial y}\int_{a(y)}^{b(y)} \mathrm{d}x\, f(x,y) = \left[f(x,y)\frac{\partial}{\partial y}x\right]_{x=a(y)}^{x=b(y)} + \int_{a(y)}^{b(y)} \mathrm{d}x\,\frac{\partial}{\partial y}f(x,y) \,. \tag{SI.24}$$

[1]In this section, we assume the maximum is uniquely defined in a neighborhood of the current parameters, i.e., there is no sudden jump of the maximum to another time. Cases in which this assumption is not satisfied have to be treated differently, e.g., by adding up gradients coming from these different maxima of equal value.

Differentiating the integral form of the maximum voltage leads to:

$$
\begin{aligned}
\frac{\partial u_{\max}}{\partial \theta} &= \frac{\partial}{\partial \theta} \int_S \mathrm{d}t\, u(t|\{\theta\}) \cdot \delta(t-\tilde{t}) && \text{(SI.25)}\\
&= \frac{\partial}{\partial \theta} \sum_i \underbrace{\int_{S_i} \mathrm{d}t\, u_i(t|\{\theta\}) \cdot \delta(t-\tilde{t})}_{\text{vanishes due to } \delta \text{ except if } i=\tilde{\imath}} && \text{(SI.26)}\\
&= \frac{\partial}{\partial \theta} \int_{t_{\tilde{\imath}}}^{t_{\tilde{\imath}+1}} \mathrm{d}t\, u_{\tilde{\imath}}(t|\{\theta\}) \cdot \delta(t-\tilde{t}) && \text{(SI.27)}\\
&= \left[u_{\tilde{\imath}}(t|\{\theta\}) \cdot \delta(t-\tilde{t}) \frac{\partial t}{\partial \theta} \right]_{t_{\tilde{\imath}}}^{t_{\tilde{\imath}+1}} + \int_{t_{\tilde{\imath}}}^{t_{\tilde{\imath}+1}} \mathrm{d}t\, \frac{\partial}{\partial \theta} \left[u_{\tilde{\imath}}(t|\{\theta\}) \cdot \delta(t-\tilde{t}) \right] && \text{(SI.28)}\\
&= \left[u_{\tilde{\imath}}(t|\{\theta\}) \cdot \delta(t-\tilde{t}) \frac{\partial t}{\partial \theta} \right]_{t_{\tilde{\imath}}}^{t_{\tilde{\imath}+1}} + \int_{t_{\tilde{\imath}}}^{t_{\tilde{\imath}+1}} \mathrm{d}t\, \left[\frac{\partial u_{\tilde{\imath}}(t|\{\theta\})}{\partial \theta} \cdot \delta(t-\tilde{t}) + u_{\tilde{\imath}}(t|\{\theta\}) \cdot \frac{\partial}{\partial \theta} \delta(t-\tilde{t}) \right] && \text{(SI.29)}\\
&= \left[u_{\tilde{\imath}}(t|\{\theta\}) \cdot \delta(t-\tilde{t}) \frac{\partial t}{\partial \theta} \right]_{t_{\tilde{\imath}}}^{t_{\tilde{\imath}+1}} + \frac{\partial u(t|\{\theta\})}{\partial \theta}\Big|_{t=\tilde{t}} + \int_{t_{\tilde{\imath}}}^{t_{\tilde{\imath}+1}} \mathrm{d}t\, u_{\tilde{\imath}}(t|\{\theta\}) \cdot \frac{\partial}{\partial \theta} \delta(t-\tilde{t})\,. && \text{(SI.30)}
\end{aligned}
$$

For the last term, the derivative of the delta distribution $\frac{\partial}{\partial\theta}\delta(t-\tilde{t})$ can be reformulated:

$$
\begin{aligned}
\frac{\partial \delta(t-\tilde{t})}{\partial \theta} &= \frac{\partial \delta(t-\tilde{t})}{\partial (t-\tilde{t})} \frac{\partial (t-\tilde{t})}{\partial \theta} && \text{(SI.31)}\\
&= -\frac{\partial \delta(t-\tilde{t})}{\partial (t-\tilde{t})} \frac{\partial \tilde{t}}{\partial \theta} && \text{(SI.32)}\\
&= -\frac{\partial \delta(t-\tilde{t})}{\partial (t-\tilde{t})} \overbrace{\frac{\partial (t-\tilde{t})}{\partial t}}^{\text{inserting } 1} \frac{\partial \tilde{t}}{\partial \theta} && \text{(SI.33)}\\
&= -\frac{\partial \delta(t-\tilde{t})}{\partial t} \frac{\partial \tilde{t}}{\partial \theta}\,. && \text{(SI.34)}
\end{aligned}
$$

Inserting this above in Eq. (SI.30) and integrating by parts[2] allows removing the derivative of the δ distribution:

$$
\begin{aligned}
\int_{t_{\tilde{\imath}}}^{t_{\tilde{\imath}+1}} \mathrm{d}t\, u_{\tilde{\imath}}(t|\{\theta\}) \cdot \frac{\partial}{\partial \theta} \delta(t-\tilde{t}) &= -\int_{t_{\tilde{\imath}}}^{t_{\tilde{\imath}+1}} \mathrm{d}t\, u_{\tilde{\imath}}(t|\{\theta\}) \frac{\partial \delta(t-\tilde{t})}{\partial t} \frac{\partial \tilde{t}}{\partial \theta} && \text{(SI.35)}\\
&= -\left[u_{\tilde{\imath}}(t|\{\theta\}) \delta(t-\tilde{t}) \frac{\partial \tilde{t}}{\partial \theta} \right]_{t_{\tilde{\imath}}}^{t_{\tilde{\imath}+1}} + \int_{t_{\tilde{\imath}}}^{t_{\tilde{\imath}+1}} \mathrm{d}t\, \delta(t-\tilde{t}) \frac{\partial}{\partial t} \Big(u_{\tilde{\imath}}(t|\{\theta\}) \cdot \frac{\partial \tilde{t}}{\partial \theta} \Big) && \text{(SI.36)}\\
&= -\left[u_{\tilde{\imath}}(t|\{\theta\}) \delta(t-\tilde{t}) \frac{\partial \tilde{t}}{\partial \theta} \right]_{t_{\tilde{\imath}}}^{t_{\tilde{\imath}+1}} + \int_{t_{\tilde{\imath}}}^{t_{\tilde{\imath}+1}} \mathrm{d}t\, \delta(t-\tilde{t}) \cdot \frac{\partial u_{\tilde{\imath}}(t|\{\theta\})}{\partial t} \cdot \frac{\partial \tilde{t}}{\partial \theta} && \text{(SI.37)}\\
&= -\left[u_{\tilde{\imath}}(t|\{\theta\}) \delta(t-\tilde{t}) \frac{\partial \tilde{t}}{\partial \theta} \right]_{t_{\tilde{\imath}}}^{t_{\tilde{\imath}+1}} + \frac{\partial u_{\tilde{\imath}}(t|\{\theta\})}{\partial t}\Big|_{t=\tilde{t}} \cdot \frac{\partial \tilde{t}}{\partial \theta} && \text{(SI.38)}\\
&= -\left[u_{\tilde{\imath}}(t|\{\theta\}) \delta(t-\tilde{t}) \frac{\partial \tilde{t}}{\partial \theta} \right]_{t_{\tilde{\imath}}}^{t_{\tilde{\imath}+1}} + \frac{\partial u(t|\{\theta\})}{\partial t}\Big|_{t=\tilde{t}} \cdot \frac{\partial \tilde{t}}{\partial \theta}\,. && \text{(SI.39)}
\end{aligned}
$$

The time derivative only acts on $u_{\tilde{\imath}}$ because $\tilde{t}$ (and its derivative) are independent of the integration variable t. Furthermore, at time $\tilde{t}$ the value of $u_{\tilde{\imath}}$ is identical to the one of u, so we can substitute the regular voltage back in.

With Eq. (SI.39) inserted into Eq. (SI.30), reordering of the terms yields

$$
\frac{\partial u_{\max}}{\partial \theta} = \left[u(t|\{\theta\}) \cdot \delta(t-\tilde{t}) \frac{\partial t}{\partial \theta} \right]_{t_{\tilde{\imath}}}^{t_{\tilde{\imath}+1}} - \left[u(t|\{\theta\}) \cdot \delta(t-\tilde{t}) \frac{\partial \tilde{t}}{\partial \theta} \right]_{t_{\tilde{\imath}}}^{t_{\tilde{\imath}+1}} + \frac{\partial u(t|\{\theta\})}{\partial \theta}\Big|_{t=\tilde{t}} + \frac{\partial u(t|\{\theta\})}{\partial t}\Big|_{t=\tilde{t}} \frac{\partial \tilde{t}}{\partial \theta}\,. \qquad \text{(SI.40)}
$$

[2] $\int \mathrm{d}x\, f(x) \frac{\partial g(x)}{\partial x} = f(x)g(x)\big|_{\text{boundary}} - \int \mathrm{d}x\, \frac{\partial f(x)}{\partial x} g(x)$

From $\tilde{t} \in S_{\tilde{\imath}} = (t_{\tilde{\imath}}, t_{\tilde{\imath}+1}]$ follows $\tilde{t} \neq t_{\tilde{\imath}}$, therefore the first two terms evaluated at the lower boundary vanish. The remaining, upper boundary terms $t = t_{\tilde{\imath}+1}$ are only nonzero if the maximum happens at that boundary $\tilde{t} = t_{\tilde{\imath}+1}$, in which case the two terms cancel each other, yielding the final, concise result

$$\frac{\partial u_{\max}}{\partial \theta} = \frac{\partial u(t|\{\theta\})}{\partial \theta}\Big|_{t=\tilde{t}} + \frac{\partial u(t|\{\theta\})}{\partial t}\Big|_{t=\tilde{t}} \cdot \frac{\partial \tilde{t}}{\partial \theta} \,. \tag{SI.41}$$

There are two distinct cases how a maximum of the voltage can be reached: The more common one is a maximum due to the decay of the voltage back to the leakage. In a neighborhood around this maximum at $\tilde{t}$, the voltage is a smooth function with time derivative $\frac{\partial u}{\partial t}|_{t=\tilde{t}} = \dot{u}(\tilde{t}) = 0$. Therefore, the second term in Eq. (SI.41) vanishes.

The other possibility is in the event of a sufficiently strong inhibitory spike: as a consequence, the voltage can decrease immediately and the time of maximal voltage is identical to the time of this inhibitory input $\tilde{t} = t_{\tilde{\imath}+1}$. In this case, the second term will be nonzero but can be calculated because both $\dot{u}$ and $\frac{\partial \tilde{t}}{\partial \theta} = \frac{\partial t_{\tilde{\imath}+1}}{\partial \theta}$ of the inhibitory input spike $t_{\tilde{\imath}+1}$ are known. This contribution is proportional to $\dot{u}$ (the left derivative at time of input spike, i.e., how much the membrane changes in free dynamics) as well as Δt (how much a change in parameter θ influences the relevant input spike time $t_{\tilde{\imath}+1}$).

Now, we investigate $\frac{\partial u(t)}{\partial \ldots}\Big|_{t=\tilde{t}}$ in two different settings, starting with the more peculiar one.

Equal time constants $\tau_\mathrm{s} = \tau_\mathrm{m}$ In this regime the voltage behaves as

$$u(t) = \sum_i \Theta(t - t_i) \frac{w_i}{g_\ell} \frac{t - t_i}{\tau_\mathrm{s}} \exp\left(-\frac{t - t_i}{\tau_\mathrm{s}}\right) \,. \tag{SI.42}$$

We can calculate the derivative to be

$$\frac{\partial u(t)}{\partial w_j}\Big|_{t=\tilde{t}} = \frac{1}{g_\ell} \sum_{i \in \{i | t_i < \tilde{t}\}} \underbrace{\frac{\partial w_i}{\partial w_j}}_{\delta_{ij}} \frac{\tilde{t} - t_i}{\tau_\mathrm{s}} \exp\left(-\frac{\tilde{t} - t_i}{\tau_\mathrm{s}}\right) \tag{SI.43}$$

$$= \frac{1}{g_\ell} \mathbb{1}_{t_j < \tilde{t}} \frac{\tilde{t} - t_j}{\tau_\mathrm{s}} \exp\left(-\frac{\tilde{t} - t_j}{\tau_\mathrm{s}}\right) \,. \tag{SI.44}$$

Similarly, we get

$$\frac{\partial u(t)}{\partial t_j}\Big|_{t=\tilde{t}} = \frac{w_j}{g_\ell} \mathbb{1}_{t_j < \tilde{t}} \frac{\tilde{t} - t_j - \tau_\mathrm{s}}{\tau_\mathrm{s}^2} \exp\left(-\frac{\tilde{t} - t_j}{\tau_\mathrm{s}}\right) \,. \tag{SI.45}$$

Unmatched time constants $\tau_\mathrm{s} \neq \tau_\mathrm{m}$ While the voltage dynamics is slightly different

$$u(t) = \sum_i \Theta(t - t_i) \frac{w_i}{g_\ell} \frac{\tau_\mathrm{s}}{\tau_\mathrm{m} - \tau_\mathrm{s}} \left[\exp\left(-\frac{t - t_i}{\tau_\mathrm{m}}\right) - \exp\left(-\frac{t - t_i}{\tau_\mathrm{s}}\right)\right] \,, \tag{SI.46}$$

the calculation is similar and results in

$$\frac{\partial u(t)}{\partial w_j}\Big|_{t=\tilde{t}} = \frac{\tau_\mathrm{s}}{\tau_\mathrm{m} - \tau_\mathrm{s}} \frac{1}{g_\ell} \mathbb{1}_{t_j < \tilde{t}} \left[\exp\left(-\frac{\tilde{t} - t_j}{\tau_\mathrm{m}}\right) - \exp\left(-\frac{\tilde{t} - t_j}{\tau_\mathrm{s}}\right)\right] \tag{SI.47}$$

$$\frac{\partial u(t)}{\partial t_j}\Big|_{t=\tilde{t}} = \frac{\tau_\mathrm{s}}{\tau_\mathrm{m} - \tau_\mathrm{s}} \frac{w_j}{g_\ell} \mathbb{1}_{t_j < \tilde{t}} \left[\frac{1}{\tau_\mathrm{m}} \cdot \exp\left(-\frac{\tilde{t} - t_j}{\tau_\mathrm{m}}\right) - \frac{1}{\tau_\mathrm{s}} \cdot \exp\left(-\frac{\tilde{t} - t_j}{\tau_\mathrm{s}}\right)\right] \,. \tag{SI.48}$$

SI.F Relationship of weight and delay for LIF-based parrot neuron

With the LIF dynamics from above (Eq. (SI.46)), we can proceed to get the desired relationship of a weight and the resulting delay. To calculate the delay of a parrot neuron that has one input spike time t associated with a weight w, one needs to compute its time of spiking (i.e., $u(T) = \vartheta$). Assuming w.l.o.g. $t = 0$ and using $T = t + d$ for a delay d yields

$$\vartheta = \frac{\tau_\mathrm{s}}{g_\ell(\tau_\mathrm{m} - \tau_\mathrm{s})} w \left[\exp\left(-\frac{d}{\tau_\mathrm{m}}\right) - \exp\left(-\frac{d}{\tau_\mathrm{s}}\right)\right] \,. \tag{SI.49}$$

Solving for the weight w returns

$$w = \frac{g_\ell \vartheta (\tau_\mathrm{m} - \tau_\mathrm{s})}{\tau_\mathrm{s}} \frac{1}{\exp\left(-\frac{d}{\tau_\mathrm{m}}\right) - \exp\left(-\frac{d}{\tau_\mathrm{s}}\right)} \ . \tag{SI.50}$$

However, this holds only if the neuron is in fact spiking. This can happen in the interval $[0; \tilde{t}]$ with $\tilde{t}$ the time at which the membrane voltage is maximal:

$$\tilde{t} = \frac{\tau_\mathrm{m}\tau_\mathrm{s}}{\tau_\mathrm{m} - \tau_\mathrm{s}} \log \frac{\tau_\mathrm{m}}{\tau_\mathrm{s}} \ . \tag{SI.51}$$

For the specific case of $\tau_\mathrm{s} = \tau_\mathrm{m}$, l'Hôpital's rule in the limit $\tau_\mathrm{m} \to \tau_\mathrm{s}$ can be applied to Eq. (SI.50) and Eq. (SI.51):

$$w = \frac{g_\ell \vartheta \tau_\mathrm{s}}{d} \exp\left(\frac{d}{\tau_\mathrm{s}}\right) \quad \text{and} \quad \tilde{t} = \tau_\mathrm{s} \ . \tag{SI.52}$$

SI.G Simulation parameters

Table SI.2: **Dataset and training parameters**. Used to produce the results in Fig. 4, Fig. 5, Fig. SI.1, Fig. SI.2, Fig. SI.3, Fig. SI.4, Fig. SI.5 and Table SI.1.

parameter name	**ideal simulation**	**hardware-aware simulation/ hardware emulation**
dataset parameters		
input size	4	4
t_{early}	0.15	0.15
t_{late}	2.0	2.0
training parameters		
training epochs	300	300
batch size	150	40
adam parameter β	$(0.9, 0.999)$	$(0.9, 0.999)$
adam parameter ϵ	10^{-8}	10^{-8}
lr-scheduler	StepLR	StepLR
lr-scheduler step size	20	20
lr-scheduler γ	0.95	0.95
delay-lr[1]	$[0.1, 0.3, 0.5, 1, 1.5, 2] \times 10^{-2}$	2×10^{-3}
weight-lr[1]	$[0.1, 0.3, 0.5, 1, 1.5, 2] \times 10^{-2}$	2×10^{-3}
input noise σ	no noise	no noise
max allowed Δw	0.2	0.2
weight bump value	0.0005	0.0005
loss Δ_t	0.2	0.3

[1] For hyperparameter optimization, a grid search was performed over the range of values in brackets.

Table SI.3: **Network parameters.** Used to produce the results in Fig. 4, Fig. 5, Fig. SI.1, Fig. SI.2, Fig. SI.3, Fig. SI.4, Fig. SI.5 and Table SI.1.

parameter name	**ideal simulation**	**hardware-aware simulation/ hardware emulation**
neuron parameters		
g_ℓ	0.5	1.0
E_ℓ	0.0	0.0
ϑ	1.0	2.6
τ_m	2.0	1.0
τ_s	1.0	1.0
network parameters		
layer 0[1]	[*broadcast, axonal, dendritic, synaptic*]	
delay init mean	0.0	0.0
delay init std	0.25	0.5
scale λ	1.0	1.5
shift	0.0	2.0
layer 1	*neuron*	
size[1]	[5, 10, 15, 20, 25, 30]	
max ratio missing spikes	0.3	0.05
weight init mean	1.0	1.0
weight init std	1.0	0.12
layer 2[1]	[*broadcast, axonal, dendritic, synaptic*]	
delay init mean	0.0	0.0
delay init std	0.25	0.5
scale λ	1.0	1.5
shift	0.0	2.0
layer 3	*neuron*	
size	3	3
max ratio missing spikes	0.0	0.05
weight init mean	1.0	0.075
weight init std	1.0	0.15

[1] Parameters over which a sweep was performed are in brackets.

Table SI.4: **Weight and delay configurations.** Used to produce the results in Fig. SI.2. These values are extracted from trained networks and averaged from 10 different seeds. They were not trained after the initialization, but rather fixed. The rest of the parameters were kept the same as in Table SI.3.

parameter name	**ablation study**			
layer 0	*broadcast*	*axonal*	*dendritic*	*synaptic*
delay mean		0.0	0.0	0.0
delay std		0.50	1.13	0.92
layer 1	*neuron*			
weight mean	0.68	0.87	0.91	0.90
weight std	1.14	1.72	1.47	1.17
layer 2	*broadcast*	*axonal*	*dendritic*	*synaptic*
delay mean		0.0	0.0	0.0
delay std		0.85	0.21	0.75
layer 3	*neuron*			
weight mean	0.56	1.71	1.64	1.26
weight std	1.53	4.55	3.88	2.12

Table SI.5: **Specific delay configurations.** Used to produce the results in Fig. SI.3. Those values were not trained after the initialization, but rather fixed. The rest of the parameters were kept the same as in Table SI.3.

parameter name	**random but fixed delay**
Layer 0	*broadcast, axonal, dendritic, synaptic*
delay mean	0.0
delay std[1]	[0.0, 0.4375, 0.875, 1.3125, 1.75]
Layer 2	*broadcast, axonal, dendritic, synaptic*
delay mean	0.0
delay std[1]	[0.0, 0.4375, 0.875, 1.3125, 1.75]

[1] For hyperparameter optimization, a grid search was performed over the range of values in brackets.

Table SI.6: **Hardware parameters.** Used to produce the results in Fig. 5, Fig. SI.4, Fig. SI.5 and Table SI.1. These values were obtained from the study made in Section SI.A.3.

parameter name	**hw-aware**
weight quant. (max range)[1]	2.1
weight quant. (precision)	1/30
FP noise (mean)	0.13
FP noise (std)	0.08
trial-to-trial (std)	0.04
delay jitter	0.01

[1] To ensure sufficient drive at the input, the maximum range of the weight was multiplied by 5 for networks with a hidden layer of 5 neurons, for both hardware-aware simulation and hardware emulation.

References

1. DAgostino, S., Moro, F., *et al.* DenRAM: Neuromorphic Dendritic Architecture with RRAM for Efficient Temporal Processing with Delays. *Nature Communications* **15,** 3446 (2024).
2. Göltz, J., Kriener, L., *et al.* Fast and energy-efficient neuromorphic deep learning with first-spike times. *Nature Machine Intelligence* **3,** 823–835 (2021).
3. Mészáros, B., Knight, J. C. & Nowotny, T. Efficient Event-based Delay Learning in Spiking Neural Networks. *arXiv preprint arXiv:2501.07331* (2025).
4. Kriener, L., Göltz, J. & Petrovici, M. A. *The Yin-Yang Dataset* in *Neuro-Inspired Computational Elements Conference* (Association for Computing Machinery, Virtual Event, USA, 2022), 107–111.
5. Müller, E., Althaus, M., *et al. jaxsnn: Event-driven Gradient Estimation for Analog Neuromorphic Hardware* in *2024 Neuro Inspired Computational Elements Conference (NICE)* (2024), 1–6.
6. Göltz, J., Billaudelle, S., *et al.* Gradient-based methods for spiking physical systems. *International conference on neuromorphic, natural and physical computing (NNPC).* arXiv: `2309.10823 [q-bio.NC]` (2023).
7. Wunderlich, T. C. & Pehle, C. Event-based backpropagation can compute exact gradients for spiking neural networks. *Scientific Reports* **11** (June 2021).
8. Brette, R. & Gerstner, W. Adaptive Exponential Integrate-and-Fire Model as an Effective Description of Neuronal Activity. *Journal of Neurophysiology* **94,** 3637–3642 (Nov. 2005).
9. Pehle, C., Billaudelle, S., *et al.* The BrainScaleS-2 Accelerated Neuromorphic System with Hybrid Plasticity. *Front. Neurosci.* **16.** arXiv: `2201.11063` (2022).
10. Billaudelle, S., Weis, J., *et al. An accurate and flexible analog emulation of AdEx neuron dynamics in silicon* in *2022 29th IEEE International Conference on Electronics, Circuits and Systems (ICECS)* (2022), 1–4.
11. Perez-Nieves, N., Leung, V. C. H., *et al.* Neural heterogeneity promotes robust learning. *Nature Communications* **12** (Oct. 2021).
12. Cramer, B., Billaudelle, S., *et al.* Surrogate gradients for analog neuromorphic computing. *Proceedings of the National Academy of Sciences* **119** (2022).
13. Bittar, A. & Garner, P. N. A surrogate gradient spiking baseline for speech command recognition. *Frontiers in Neuroscience* **16,** 865897 (2022).
14. Nowotny, T., Turner, J. P. & Knight, J. C. Loss shaping enhances exact gradient learning with Eventprop in spiking neural networks. *Neuromorphic Computing and Engineering* **5,** 014001 (Jan. 2025).

8. Summary and discussion

This thesis makes the attempt to study sparsely communicating neural networks, with a focus on the exact prediction of their dynamics and their gradient-based optimisation. Our work bridges neuromorphic engineering and neuroscience, studying the computational efficiency of spike-based Artificial Intelligence, linking to experimental evidence from biology – and even creating a new educational neuromorphic system showcasing the crucial neuronal properties at the foundation of this thesis.

The new neuromorphic hardware as described in Chapter 3 has proved transformative for explaining concepts from neuroscience to the public as well as researchers in adjacent fields. Both tangible and flexible, Lu.i has engaged a global audience, at a number of universities, schools and companies. The PCBs implement fundamental properties of biological neurons, especially spiking communication and integration of inputs across space and time.

To help replicate the spatio-temporal correlation of neurons in artificial networks, we have devised the Yin-Yang (YY) dataset (Chapter 4): it enables fine-grained characterisation of new algorithms and hardware, addressing restrictions of existing benchmarks such as logic problems or the MNIST dataset. Convincing metrics need to be the cornerstone for meaningful evaluations, for they provide an understanding for the underlying processes and existing limitations. The YY dataset replaces complexity-by-size with genuine difficulty, and has proven a valuable addition for the community interested in unlocking spatio-temporal learning.

In Chapter 5, we described a solution to exploiting the event-based nature of a spiking neural network (SNN): exact gradients exist, can be used to train networks, and are robust enough for usage with imperfect analogue hardware. Enabled by the YY dataset, we performed systematic robustness studies, and the realisation on BrainScaleS-2 is the convincing statement that the algorithm successfully handles inaccuracies, at the same time empowering fast and energy-efficient classification. This implementation is also used to assess the system health of the neuromorphic infrastructure in Heidelberg, to ensure the operation of BrainScaleS-2 within the *EBRAINS* cloud platform. Our approach has established itself as an innovative method (Frenkel 2021), expanding the frontiers of both neuroscience and neuromorphic engineering.

We have used the method ourselves for studying activities in SNNs to find relations to established models of information propagation in the brain (Chapter 6). It enabled the training of deep, biological architectures, facilitating the separation of inhibitory and excitatory effects, and to identify sub-networks that form as a result of training, both reminiscent of biological equivalents.

Furthermore, the synergy of our training framework with the concept of transmission delays has allowed us to extend the set of trainable parameters, drastically enriching the temporal dynamics (Chapter 7). The training of delays appears naturally and with no additional requirements, enabling a systematic study underpinning their significance and, in addition, an implementation on the BrainScaleS-2 hardware.

8.1. Analytical spike times

Guided by rigorous mathematical treatment we found an analytical relation for the spike time of leaky integrate-and-fire (LIF) neurons: in Chapter 5 we described the solution for the first spike time, given certain proportions of time constants: when the synaptic τ_{syn} and membrane time constant τ_{mem} are identical, the relation for the time of the spike is found with the help of the differentiable Lambert $\mathcal{W}$ function (Chapter 5, Eq. 2).[21] Furthermore, if the membrane time constant is twice as large as the synaptic, $\tau_{\text{mem}} = 2\tau_{\text{syn}}$, the resulting quadratic equation can also be solved (Chapter 5, Eq. 3). While not used in our studies, solutions exists also for other ratios of time constants.[22] Together with the generalisation to multiple spikes (Chapter 7, Section SI.D), this enables precise calculation of activity in SNNs.

Compared to established methods, this is a radically new approach: usually, simulators for neuronal states are numerically integrated. Even in cases where the subthreshold dynamics can be calculated exactly, the simulation and spike times are confined to a time grid (Rotter and Diesmann 1999). To guarantee faithful reproduction of more complex neural dynamics or effects like synchronisation, a small enough simulation time step is required (Hansel et al. 1998; Valadez-Godínez et al. 2020), with detrimental effects on simulation times. A sub-timestep resolution of the spike time can still be achieved, but typical mechanisms for that purpose rely on interpolation (Morrison et al. 2007) or iterative root-finding (Hanuschkin et al. 2010). Brette (2007) argued for the pursuit of exact spike times for more biological neuron models, and in fact our work extends the argument: aiming not only to find methods for exact spike times, but also to produce these results analytically, or even with a closed-form expression.[23]

This is also possible for simpler models, for example, neurons with delta synapses (Mattia and Giudice 2000): the membrane voltage jumps at each input spike and decays from there, i.e., the postsynaptic potential (PSP) is monotonically decreasing. This requires the spikes of a neuron to coincide with an input, severely limiting expressivity. When including delays the activity is non-trivial, but essentially ruled by combinatorics. The far more interesting scenario allows the PSPs to accumulate, enabling the neuron to perform spatio-temporal integration. Additionally, for biological realism as well as computational advantage, it is desirable for inputs to have temporally limited effects and the neuron to forget its excitation over time. The most treatable model fulfilling these requirements is the LIF neuron with exponential, current-based (CuBa) synapses. Not entirely coincidentally, this model serves as the basis of many studies in computational neuroscience (Abbott 1999; Brunel and van Rossum 2007a) and is featured on a lot of neuromorphic systems.

Our approach allows the realisation of interesting neuron models in their essential form, purely event-driven, transforming a set of input spikes into output spikes, and provides a

[21]This result was published simultaneously and independently by a team from Google (Comşa et al. 2020; Comşa et al. 2022).

[22]The existence of a solution for a specific ratio implies the solution of the inverse ratio, because the voltage dynamic Equation (2.12) is symmetric under exchange of τ_{mem} and τ_{syn}. This means that $\tau_{\text{mem}} = \tau_{\text{syn}}/2$ can be treated similarly, by solving a quadratic equation. In general, for a ratio $n = \tau_{\text{mem}}/\tau_{\text{syn}} \neq 1$, the relation for the time of a spike can be brought into the form of the generic equation $0 = ax^n + bx + c$. This equation can be solved, e.g., for $n = 3$ by finding the root of the depressed cubic (discovered by Scipione del Ferro and Niccolò Fontana Tartaglia, published by Cardano 1545) and for $n = 4$ with Ferrari's method (named after Lodovico Ferrari, published as well by Cardano 1545). However, the calculations get more challenging and there is no apparent benefit from ratios different from 1 or 2.

[23]In contrast to the relation for identical time constants which contains the analytical Lambert $\mathcal{W}$ function, the spike time for $\tau_{\text{mem}} = 2\tau_{\text{syn}}$ can be calculated using only simple operations: sums, products, exponentials, a square root and a logarithm (Chapter 5, Eqs. 3 and 11).

ground truth for spiking activity. Crucially, as the computation does not depend on voltages, this also cements the insight that the relevant information *are* the spike times. Without additional computational effort, we also allow the information to be represented precisely, with the sub-millisecond precision found in biological neurons (Mainen and Sejnowski 1995).

8.2. Sparse and exact gradients

The accurate study of the dynamics reveals that the essential vessel of propagated information are the spikes. A profound insight results from the existence and properties of the analytical relations: from their differentiability it follows that the spike times are differentiable with respect to the parameters. Looking specifically at the synaptic weights, this allows us to exactly quantify the influence of each weight on the time of the spike, i.e., how a change in the weight changes the spike time. In addition, the differentiability also holds for other parameters, notably the input spike times: by relating a deviation of an output spike to a shift in the input spike, this derivative unlocks error backpropagation. With consecutive applications of the local derivatives, we can propagate an error in the output layer through a deep network. At the same time, the propagation is sparse because neurons only interact with spikes. Together, this enables the exact and event-based calculation of the gradient of any loss function, with respect to any parameter in a deep spiking network. Incidentally, for our usage with a time-to-first-spike coding, the algorithm can be brought close to the traditional formulation of error backpropagation (Eq. (2.28)), see Chapter 5, Eq. 39.

Representing a network's state in spike times has another benefit besides the apparent sparsity. In the form of Eq. (2.10), transmission delays appear as a natural parameter of the neuronal computation, alongside the input spike times and weights. Delays have been established as an effective computational paradigm for SNNs (Maass and Schmitt 1999; Patiño-Saucedo et al. 2023; D'Agostino et al. 2024), however, performing gradient-based training in a time-stepped framework requires workarounds: either finite-difference methods (Shrestha and Orchard 2018) or a surrogate for the time shift associated with the delay (Hammouamri et al. 2024; Queant et al. 2025). As worked out in Section SI.D, delays are straight-forwardly integrated in our approach. From a computational view, they can be seen as an operation on spike times separate from the neuronal dynamics, enabling an efficient implementation both in software and on neuromorphic hardware (Chapter 7, Figs. 2 and 5). In addition, this separation has allowed us to systematically study the effects of different types of delays: as a reference to biology, we distinguish between axonal, synaptic, and dendritic delays (Chapter 7, Fig. 3). In Chapter 7, Figure 4, their respective impact on training performance is shown. While synaptic delays demonstrate superior performance, taking the quadratic parameter scaling into account levels their benefit relative to the other delay types, while all delay networks still outperform networks lacking delays.

An issue that is often raised in conjunction specifically with spiking networks is a difficulty to train deep networks. This challenge originates from artificial neural networks (ANNs) where it is caused by the vanishing gradient problem (Hochreiter 1991; Hochreiter et al. 2001). Some papers claim that it is aggravated in SNNs (Stanojevic et al. 2024). However, we have not encountered this issue: Chapter 6 demonstrates networks of up to seven layers depth with no problem of error propagation (see Fig. SI.B.2 for a network of ten layers). Furthermore, in Chapter 7 we show that increasing depth increases performance (Chapter 7, Fig. SI.1). Even more strikingly, at equal parameter counts a deep-and-narrow network performs as good as a shallow-and-wide network: this confirms that the algorithm can make use of every parameter in the network and that the exact gradient propagates deep into the deep network.

Decoding the network output In our experiments, we have investigated SNNs where the decision of the network is determined by spikes in the last layer of neurons. The nonlinearity associated with spikes can complicate learning: in ANN architectures – in order to facilitate training – the readout layer typically has a 'softer' activation function like a softmax function, as opposed to rectifying linear unit (ReLU) activations in hidden layers (Krizhevsky et al. 2012; He et al. 2016). Similarly, for our systems we have the freedom to choose internal variables like the membrane voltage as an output.[24] This is a possible for simulations as well as for neuromorphic substrates with support for accessing this observable. Voltage-based decoding typically involves a fixed observation period, requiring the knowledge of the voltage for the entire period before a decision can be reached. In contrast, time-to-first-spike (TTFS) decoding facilitates fast classifications that happen at the first spike, strikingly highlighted in the raster plots in Chapter 6. There, the classification occurs well before the majority of spikes happen in the network, even before most of the input spikes (Chapter 6, Fig. 1).

When using voltage-based decoding, interestingly, only little additional information is required: because the output population is a small fraction of all neurons, there is no significant increase of memory or I/O costs. It is important to note that the usage of membrane voltages as an output is not synonymous with a departure from event-based gradients: as we have shown in Chapter 7, Section SI.E, certain voltage-based losses can be optimised in an exact and event-based fashion. Using spike times grants access to a high-precision coding, but in contrast, the smoothing effect of the voltage is generally assumed to provide increased stability (Göltz et al. 2023; Nowotny et al. 2025). A promising middle ground could be to base the decisions purely on spikes while including voltage-based regularisation for a more stable training.

8.3. Fast and energy-efficient classification on a custom accelerator

A critical test for robustness is the implementation on an analogue neuromorphic system, where mismatch in the underlying circuits on the chip leads to variabilities, i.e., parameters of different circuits are not identical and they exhibit inhomogeneous dynamics. BrainScaleS-2 has the capacity of tuning parameters to homogenise the emulated behaviour. After this calibration, the standard deviations of, e.g., the time constants τ_{mem} and τ_{syn} are close to 5 % (Billaudelle 2022). To adjust to the remaining imperfections, we adopt an in-the-loop approach: we calculate the parameter updates using observables from a forward pass on the substrate (Fig. 2.6). Crucially, we have shown that an analysis of the mathematical model makes a difference: when considering the naïve derivative of the spike time, we have identified and inserted the formula for that spike time (Chapter 5, from Eq. 22 to 24). In this way, we can integrate another measure of the 'operating point' of the substrate into the gradient, fully utilising the measured spike times for the parameter updates. In simulation studies of the robustness, this stabilised the training to a wider range of synthetical impurities (Chapter 5, Fig. 5). Also on BrainScaleS-2, the full algorithm deals well with the existing inaccuracies (Chapter 5, Fig. 4)

[24]The loss function uses this output to quantify the performance in a differentiable manner, thereby guiding the training. We want to note that the choice of this loss function also has a direct impact on the performance, but ultimately it is a secondary topic.

and incurs only a small performance loss when comparing simulation and emulation results (Chapter 5, Table 1).

Examined for the MNIST dataset, the inference with a trained network on BrainScaleS-2 has remarkable characteristics: even including communication time, 10 000 test samples are sequentially classified within 1 s. The streamed rate is even higher, above 20 kHz: as the classification is performed fully sequentially, this means that the samples are separated by less than 50 µs. The values are measurements in an experiment where the BrainScaleS-2 chip is only consuming 175 mW, which comes down to an energy per classification of 8.4 µJ (see Chapter 5, Section SI.E for details). The power consumption is lower than for competitive approaches because we can power down a lot of infrastructure on the chip including the plasticity processing units needed for voltage-based classifications (Cramer, Billaudelle et al. 2022). In addition to the stated metrics, our approach also performs well in terms of accuracy compared to other approaches (Chapter 5, Table 2), and is fully surpassed only by a method based on surrogate gradients that has since been published (Cramer, Billaudelle et al. 2022).

Notably, the time-to-classification in the network is actually much shorter than the 50 µs experiment duration, approximately 10 µs. The disparity arises because the residual activity needs to decay back to rest state for the network to be able to classify the next input. A speed-up can be achieved straightforwardly by triggering a reset. With this, Cramer, Billaudelle et al. (2022) increase the rate by more than a factor of 4 while the additional circuits only draw approximately 20 mW more power.

Recently, we have boosted our method by including an on-chip implementation of transmission delays. While we have recovered the expected performance enhancement due to the inclusion of delays, we have observed an increased benefit: Chapter 7, Figure 5 shows a substantial improvement of the outcome of a network by including delays, probably by stabilising the activity against noise. This is especially visible in these experiments because they are conducted with small networks, where hardware inhomogeneities have exacerbated repercussions. The stabilising effect, however, is expected to also improve performance in tasks requiring larger networks.

8.4. Relations to existing methods

Beyond our approach of analytic, event-based processing and exact gradients, alternative methods have been used to train SNNs. Though these methods aim for the same objective, they vary significantly in terms of applicability, generality, exactness, and compatibility with event-based computation.

Surrogate gradients Since their introduction by Zenke and Ganguli (2018) and Neftci et al. (2019), surrogate gradients have become the de facto standard to train SNNs due to the broad dissemination, their ease-of-use, and independence from specific neuron models. The approach is based on assuming the spiking output of a neuron purely as a function of its membrane voltage u_{mem} and, crucially, for the gradient calculation defining a differentiable approximation to this function, the surrogate (Eq. (2.29)). Essentially, this amounts to changing the problem: the proper question of the influence of parameters (like the input spike time) on the output of the neuron, $\partial T / \partial t$, is replaced with combining the influence of parameters on the membrane potential $\partial u_{\mathrm{mem}} / \partial t$ with the influence of the membrane potential on the spike train, $\partial S / \partial u_{\mathrm{mem}}$. This change of question is directly related to the common misconception about a supposed lack of differentiability of spikes: while the membrane voltage is not differentiable at the time of the spike when it is reset, the spike time itself is still differentiable as we showed in Chapter 5.

Incidentally, the modified question is the direct transfer of the common formulation of error backpropagation, Equation (2.28), from classical deep learning (DL). With the surrogate in place, the training essentially becomes backpropagation through time (BPTT) for an equivalent recurrent neural network (RNN) (Neftci et al. 2019). Because BPTT is an established method to train ANNs on time sequence tasks in DL (Werbos 1990), its highly-developed machinery becomes fully applicable, combined with the associated known downsides like costly scaling of memory and compute (Neftci et al. 2019; Marschall et al. 2020).

More specifically, introducing the surrogate requires the membrane voltage for the calculation of the gradient. As opposed to the sparseness of spikes, the voltage is a dense quantity, defined at all times, requiring large amounts of memory and compute even for simulations. There are also direct consequences for neuromorphic implementations: on analogue hardware, measuring the voltages of all neurons in parallel demands a fast analogue-to-digital converter (ADC). For the update calculation, both analogue and digital implementations require access to numerical values, resulting in correspondingly high bandwidth demands. In general, the approach follows the tradition of DL to resort to algorithmically simple solutions, replacing algorithm complexity with increasing computational cost.

Additionally, since the derivation fundamentally rests on an approximation, one consistently operates within this approximation. In contrast to, e.g., the simplification a model introduces where abstraction can render a task feasible, the approximation at hand inherently changes the information flow in the forward and backward path: the optimised system is different from the actual SNN. This can be helpful, because for quiescent networks the surrogate operates as an implicit regularisation to create suitable activity. Yet, it also distorts the updates, to the point that the surrogate gradient can aim in the opposite direction of the correct gradient (Gygax and Zenke 2025). This paper also shows that, mathematically, surrogate gradients are a non-conservative vector field, i.e., there is no modified loss function which one is guaranteed to optimise. Although it is not clear how the observation of opposing gradients generalises to the high-dimensional environments encountered in practise, it proves that following the surrogate gradient can lead to deteriorating performance.

EventProp An intrinsically exact alternative is the EventProp algorithm, described by Wunderlich and Pehle (2021). Based on the adjoint method (Galán et al. 1999; Serban and Recuero 2019), it is an extension of existing applications for neural networks (Chen et al. 2018; Jia and Benson 2019). It also builds on work proving the applicability of the implicit function theorem for neuronal dynamics at the time of a spike (Yang et al. 2014). The algorithm provides a general method to calculate exact gradients in an SNN. In simulations, for the forward path it relies on explicit equations like ours, or time-stepped numerical integration (e.g., *GeNN* by Nowotny et al. 2025). For models for which the evolution of the state variables is known as an explicit expression, a root solver can be employed to find the time the voltage reaches the threshold (Brette 2007).

Given spike times from a forward pass as described above or from a neuromorphic substrate, EventProp describes a way to arrive at the desired gradients of the loss. The respective differential equations can either be integrated numerically, or analytically for an event-based computation. In particular, for $\tau_{\mathrm{mem}} = \tau_{\mathrm{syn}}$ and $\tau_{\mathrm{mem}} = 2\tau_{\mathrm{syn}}$ one recovers our results (Blessing 2023). The absence of explicit equations for the gradient calculation limits the insights into the low-level dynamics of learning, such as the interpretation of the spike-timing-dependent plasticity (STDP) term, but achieves an algorithmically more straightforward formulation.

8.5. Activity and dynamics in biological brains

Our networks are synthetically optimised for machine intelligence – and yet claim to be inspired by biology. This of course raises the question of how close they actually are to biological circuits. For an answer, we focus first on the mode of information coding, second on the combined analysis of network dynamics and structure as shaped by training, and finally on a connection to delay plasticity in biological neurons.

Spike time coding in the brain As described in Section 2.1, information encoding based on individual spikes and their precise timing has been observed in a variety of brain areas and species. This might seem to contradict the stochasticity that is commonly associated with brain activity. In their seminal paper, however, Mainen and Sejnowski (1995) show stochastic behaviour as a consequence of synthetic input, while realistic synaptic stimuli lead to highly reproducible spikes with little variation.

Notable among the observations of precise spikes in the brain is their role in tactile sensing, e.g., in encoding of whisker localisation in rats (Panzeri et al. 2001; Petersen et al. 2001) and encoding tactile information in fingertips in humans (Johansson and Birznieks 2004). Another example is the auditory system, where first spikes are crucial (Carr and Konishi 1990; Heil 2004). Few-spike codes also play an important role in vision: specifically, in a variety of animals the retina seems to use precisely timed spikes (Berry et al. 1997), and in monkeys, intricate patterns have been observed in higher cortical areas, too (Ayzenshtat et al. 2010). Also in human vision, timing of spikes is considered to be important (Thorpe et al. 2001; Reich et al. 2001). Being able to operate in the limit of only few spikes enables the short timescales observed in human vision, which are measured to be on the order of 150 ms (Antal et al. 2000; Thorpe et al. 1996). Lamme and Roelfsema (2000) present evidence that simple visual processing is driven by a first wave of activity that passes in a fast, feedforward manner through the visual hierarchy, carried by single spikes per neuron. What all of these experimental observations have in common is the need for fast information propagation to enable rapid reactions based on sensory input, relying on the precise timing of individual spikes. Together, they on the one hand provide a biological motivation for the TTFS encoding employed in this thesis, that unlocks fast and energy-efficient classifications similar to how an animal needs to be able to perform quick reactions in a resourceful manner. On the other hand, it also moves our networks closer to biological approaches for information processing, opening the possibility for a more detailed comparison of the respective network dynamics.

Emergence of synchronised activity along distinct pathways We explored the structure and activity of our trained networks in Chapter 6. The investigated networks use TTFS encoding and are particularly deep to study the propagation over multiple layers. In addition, the networks adhere to Dale's law, i.e., in each layer the neurons are separated into an excitatory and an inhibitory population. In the studied networks, this generally does not impair the training but allows the clear distinction of excitatory and inhibitory effects.

The networks were trained to classify images of handwritten digits, the MNIST dataset. The size of the layers is not designed for efficiency (i.e., drastically more parameters compared to Chapter 5) but to allow the emergence of population effects by many participating neurons.

Following a stimulation, a wave of activity propagates through the network. In early layers, the temporal distribution of this wave spreads, while it condenses towards the later layers (Chapter 6, Figs. 1 and 4; Fig. SI.B.2), effectively synchronising while travelling through the network. Packets of pulses have been the focus of theoretical studies before (Diesmann et al. 1999; van Rossum et al. 2002; Vogels and Abbott 2005; Shimizu and Toyoizumi 2025), however, the previously studied networks have been randomly initialised as opposed to being trained on a task. Examining activity propagation across layers of neurons, synchronisation was observed in both in vitro (Reyes 2003)[25] and in vivo studies (Gray et al. 1989; Gray and Singer 1989). It is speculated to enable complex visual processing via the connection of concepts related to each other but encoded in spatially distant locations in the brain (Singer and Gray 1995).

When training our networks, we observed the formation of distinct pathways for the excitatory activity (Chapter 6, Fig. 4). In addition, the populations of inhibitory neurons develop a broad dampening effect on the next layer that is akin to the blanket of inhibition described for the cortex (Fino and Yuste 2011; Karnani et al. 2014). Both effects together, the selectivity of excitatory neurons and unspecific inhibition, have also been observed in the visual cortex (Sohya et al. 2007; Niell and Stryker 2008). The formation of distinct chains leads to a semantisation of activity. As highlighted by Chapter 6, Figure 4a, a neuron in a chain being active becomes indicative of the respective class, similar to the idea of grandmother neurons in the brain: few and even single neurons encoding high-level concepts (Kobatake and Tanaka 1994; Quiroga et al. 2005; Rust and DiCarlo 2010). Importantly, while our networks were initialised stochastically and with a dense, all-to-all connectivity between layers, the training forges distinct regions with little overlap. With semantisation and synchronisation, our training method recreates two key features of the visual hierarchy in our networks, strengthening the tie between synthetic models and experimental observations.

Adaptation of transmission delays in animals While the main point of our study of transmission delays in Chapter 7 was to explore their relevance in a DL context, reviewing studies of delay learning in biology proves insightful. In the brain, the delay along an axon is determined by the axonal length and the conduction speed which in turn depends on the axon's morphology and extent of myelination. Recent evidence suggests the importance of myelination change for learning: McKenzie et al. (2014) study specifically novel myelination through newly-generated oligodendrocytes (OLs), and examine the learning of a new motor task in mice.[26] Without active myelination, mice can still perform previously encountered tasks and improve their performance on these tasks. However, they are significantly impaired in learning a new task, and McKenzie et al. conclude that 'motor skill learning requires active central

[25]In a newer study, Barral et al. (2019) indicate that synchronisation does, in fact, not occur. However, their setup differs in several central aspects from ours, among them network architecture, stimulus strength, and preparation of networks: their cultures are unconditioned while our networks are extensively trained to evoke responses given a specific type of input. Two questions arise naturally, one, how activity in our network shifts while the network goes from untrained to trained state, and two, how the in vitro neurons change their activity when they are conditioned on the stimuli.

[26]OLs are responsible for myelination in the central nervous system. Crucially, McKenzie et al. inactivate the Myelin regulatory factor (Myrf) precisely in the precursor cells of OLs. Because Myrf is essential for generating myelination, the lack of Myrf in new OL cells makes it possible to study the effect of blocking novel myelination, without affecting existing OLs and the present myelination.

myelination'. Other studies have recently corroborated this new idea of the indispensability of myelination for learning tasks (Xin and Chan 2020; Steadman et al. 2020; Pan et al. 2020).[27]

Given the significance of plasticity of delays for spatio-temporal tasks in biology, it seems appropriate to study this behaviour in more detail, possibly identifying the local purpose of changing delays on a circuit level. This might allow the reevaluation of the ties between our delay learning and biology: by knowing about the direct computational effect of a delay change, our learning rules could offer guidance for experimental design as well as interpretation of experimental observations. Another insight concerns the OLs: because they control the myelination of many different axons, they have access to relative spike times between these axons. This would otherwise not be 'local' information, but OLs allow it to be part of a biologically plausible learning rule.

One possibility for such a study of adaptation of biological delays is with isolated cultures that grant a high level of control. Experiments of clusters of human-induced pluripotent stem cells, sometimes called *organoids*, have recently gained attention (Kagan et al. 2022). In particular, this new technique allows imposing lengths of axons by guiding their growth through specifically structured PDMS substrates (Amos et al. 2024). With the help of microelectrode arrays, such experiments provide precise control over stimulation as well as measurement (Küchler et al. 2025), and thus enable studies of transmission delays (Amos et al. 2025). Augmenting these experiments to include OLs would, for example, permit the precise study of information exchange between axons across the OL cell.

8.6. Improved functionality towards biologically plausible online learning

Our work provides many possible future directions, some of which we outline in the following. The most apparent is the extension to networks of more complex dynamics as well as architectures, while retaining efficiency and interpretability. In addition, it is desirable to continue our analyses of activity in the network, to further stabilise the training, and to investigate biologically plausible variants.

Exact training of more complex networks In Chapters 5 to 7 we restricted both our input stimuli and the network activity to a TTFS coding. With it, we were able to reach high accuracies with efficient simulations, describe a connection to biology, and achieve fast and accurate classification on a neuromorphic substrate. However, for more general applications on longer timescales, like processing of audio (Cramer, Stradmann et al. 2022; Warden 2018), flexibility in spiking activity is desirable, especially flexibility in the number of spikes (Bacho and Chu 2023). While multiple input spikes can be handled trivially, the capability of supporting multiple spikes within the network requires the adaptation described in Chapter 7, Section SI.D. In the case of single spikes, we could make use of efficient data structures for the spike times (Chapter 5, Eq. 39), but multiple spikes per neuron compromise this efficiency. It constitutes

[27]Changes in myelination are also connected to several diseases: severe mental illnesses are correlated with altered OLs, specifically modifications in genes related to myelination (Sokolov 2007). Makinodan et al. (2012) observed changes in myelination following social isolation during weaning that prevent normal cognitive function. In adult mice, too, isolation leads to (recoverable) behaviour-affecting alterations in myelination (Liu et al. 2012). There is also a hypothesised mechanism for the reduction of myelination associated with ageing (Shen et al. 2008). Altogether, it is a field of active study with many open questions, e.g., how OLs can be assisted and subsequent myelination facilitated (Chamberlain et al. 2017; Rosko et al. 2023).

not a conceptual but a practical problem, and a solution might be to adopt existing software. Besides the time-stepped software *GeNN*[28] (Knight and Nowotny 2023) an event-based solution (*jaxsnn*, Müller et al. 2024) has only recently undergone significant improvements. Incidentally, devising meaningful benchmarks for comparisons of time-stepped and event-based approaches is an ongoing endeavour.

When attempting to solve tasks that exhibit long intrinsic time constants, a typical solution is to include recursion: networks with this architecture, RNNs, allow neurons to stimulate themselves via feedback, naturally extending the effective timescales of the network dynamics. This feedback effectively requires multiple spikes. Consequently, the inclusion of multiple spikes facilitates the consideration of more general architectures. For specific tasks, transmission delays can reduce the significance of the increased computational complexity of RNNs (Chapter 7, Fig. 4; D'Agostino et al. 2024), but for a general training framework flexibility in the architecture is certainly preferable. While RNNs introduce no additional mathematical complication, they might require a more elaborate execution of the computation to be efficient, but also bring the networks closer to biological structures.

Studying mathematical properties of networks Successful training of a network amounts to finding a trajectory through the loss landscape to a minimum. The high-dimensional nature of this landscape, unfortunately, renders any intuition difficult. Any change to a network's topology or the choice of output decoding has considerable effects, complicating the comparison of competing approaches (Göltz et al. 2023). Complimenting our analysis in Chapter 6, a new avenue to understand the conditions for the computational complexity of networks is an analysis inspired by linear pieces in ANNs (Montúfar et al. 2014). For SNNs, Dold and Petersen (2025) extend the idea of causal sets (c.f., Chapter 7, Section SI.C) to causal pieces. Each such piece is a connected subset of the space of inputs and parameters for which the output is generated by an identical chain of activity, that is, a region in the input for which the computational graph does not change. Both network expressivity and training success has been shown to positively correlate with the number of causal pieces (Dold and Petersen 2025). However, the analysis so far is limited to non-leaky integrate-and-fire (nLIF) neurons, and the extension to the more complicated LIF model which was described in this thesis is both straight-forward and promising. The goal, then, is to determine factual backing for existing intuition, e.g., concerning initial values and the effectiveness of ad hoc techniques, ultimately leading to improvements in the performance.

Regularising activity Our method of training SNNs requires the neurons to be active: in order for the derivative of the spike time to exist, a spike must have occurred. To ensure sufficient activity, we employ homeostatic weight bumping (Mostafa et al. 2017; Nowotny et al. 2025). For all neurons with insufficient activity, it indiscriminately increases weights: the exact gradients do not have information on a change in the loss upon neither creation nor deletion of spikes. The necessity of spiking activity for well-defined gradients is a natural requirement of all spike-based approaches. In contrast, surrogate gradients inherently act as a regularisation of the activity, treating suboptimal initialisations gracefully by driving the network to an active state even when initially quiescent (Zenke and Ganguli 2018). This is a motivation to

[28] *GeNN* has enriched research (e.g., Max et al. 2024; Nowotny et al. 2025; Mészáros et al. 2025), but as its training of SNNs is based on EventProp it deprives us of explicit equations. Additionally, compared to surrogate-gradient-based approaches, the communication is event-based, but the software flow is inherently time-stepped (Shoesmith et al. 2025).

incorporate voltage-based terms into the loss, e.g., by mixing surrogate and exact training, the former to settle into a good position in the loss landscape independent of initialisation, the latter to efficiently and exactly find the optimum. Another promising direction lies in stepping away from the coarse weight bumping and, for all neurons i without a spike, using a term proportional to their maximum voltage $\alpha \sum_i u_{i,\max}$ as a regularisation. In essence, this finds the path to a spike requiring the smallest weight change. As shown in Chapter 7, Section SI.E, the respective gradient can be calculated in an event-based fashion and only depends on the time of the maximum (Chapter 7, Eq. SI.41). This could be a possibility to unite many advantages: stabilisation of learning by including voltages, data minimisation through event-based training, and efficient inference only relying on spikes.

Biologically plausible online learning Among neuroscientists, it used to be a commonplace that the brain does not learn in supervised fashion with error backpropagation (Grossberg 1987; Crick 1989). At first glance, learning by gradient descent (Eq. (2.22)) requires access to a global loss function, which is in apparent conflict to biological constraints such as locality in space and time. However, more recently the debate has opened up again (Lillicrap and Santoro 2019; Lillicrap et al. 2020). A key insight is that optimising performance with small changes, maybe inadvertently, correlates with gradients of a loss (Richards and Kording 2023). In addition, a variety of biologically plausible mechanisms for transporting the error signals through networks have been described.[29] For the most part, these proposals resort to rate coding in one form or another. In rate codes, an error usually means that a neuron should be either *more* or *less* active. In spike time coding, however, an error indicates the need for an *earlier* or *later* spike, and no natural transfer between the two descriptions is currently known.

However, once the error of the spike is locally available, the parameter update itself is local according to our learning rule.[30] Additionally, the simplified version (Chapter 5, Section SI.D) amounts to simple STDP with the error as a third factor. This implies that the update is biologically plausible and amenable to a hardware implementation, *if* the error is made available locally. Deciphering how the brain blends the required gradients together with the spiking implementation of communication and plasticity is one of the central enigmas in computational neuroscience, and fully resolving it will also lead the way to online learning of streaming data even in a hardware implementation.

[29]Some proposals mainly target the implementation of error backpropagation in cortical microcircuits (Sacramento et al. 2018). Other, more performant approaches are geared towards fast inference and reliable learning in the presence of intrinsically slow neurons (Haider et al. 2021; Senn et al. 2024), or dedicated to finding an efficient and timely distribution of errors throughout the network to solve the spatio-temporal credit assignment problem (Ellenberger et al. 2025).

[30]The prefactor in Chapter 5, Equations 4 and 5, seems to depend on nonlocal quantities, as a synapse would not be expected to be aware of the spike times of *other* presynaptic neurons. However, as shown in Section SI.C, the prefactor is proportional to the time derivative of the membrane potential, which is a local variable.

8.7. Conclusion

The brain's efficiency in solving complex spatio-temporal tasks served as the central inspiration throughout this work. It led us to establish a new benchmark that fundamentally requires the detection of the intrinsic spatio-temporal fingerprint (Chapter 4). We furthermore solved the challenge of training deep spiking networks (Chapters 5 and 7), and related these trained networks to their biological archetype, recovering key principles of neuronal sensory integration (Chapter 6).

In addition, we have put forward a tangible demonstrator for education and outreach (Chapter 3): the dissemination of our knowledge and insights is a central justification of public investment, and Lu.i is a prime example for successful science communication. It has been used in schools, academic lectures, and by industry, has appeared at a multitude of public science events, and it is set to appear in a long-term exhibition in the *Senckenberg Naturmuseum* in Frankfurt.

The central scientific contribution of the thesis is the exact and event-based formalism to train spiking neural networks and its application on a neuromorphic substrate for fast and efficient classification. The approach has been picked up by other groups, notably by Bacho and Chu (2023) and Klos and Memmesheimer (2025). In addition, the realisation that information flows through the network as spikes is also at the foundation of *EventProp*. Our explicit equations are the workhorse for the respective software framework, *jaxsnn*, and serve as a ground truth for comparisons. Our work marks, to our knowledge, the first high-performance, energy-efficient solution for classification on a mixed-signal neuromorphic substrate, setting a new standard for fast and robust processing.

The era that was governed by Moore's law advanced at a steady pace with predictable gains in speed and efficiency. In the future, progress is expected to be a lot more unpredictable and will likely yield increasingly specialised computing infrastructure. Drawing targeted insights from neural mechanisms and translating them into optimised algorithms and hardware can yet unlock the next wave of transformative innovation.

List of publications

Peer-reviewed publications as first author[31]

Sebastian Billaudelle*, Yannik Stradmann*, Korbinian Schreiber*, Benjamin Cramer*, Andreas Baumbach*, Domnik Dold*, **Julian Göltz***, Ákos F. Kungl*, Timo C. Wunderlich*, Andreas Hartel, Eric Müller, Oliver J. Breitwieser, Christian Mauch, Mitja Kleider, Andreas Grübl, David Stöckel, Christian Pehle, Arthur Heimbrecht, Philipp Spilger, Gerd Kiene, Vitali Karasenko, Walter Senn, Mihai A. Petrovici*, Johannes Schemmel and Karlheinz Meier (Oct. 2020). 'Versatile emulation of spiking neural networks on an accelerated neuromorphic substrate'. In: *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, pp. 1–5. DOI: `10.1109/ISCAS45731.2020.9180741`

Julian Göltz*, Laura Kriener*, Andreas Baumbach, Sebastian Billaudelle, Oliver Breitwieser, Benjamin Cramer, Dominik Dold, Akos Ferenc Kungl, Walter Senn, Johannes Schemmel, Karlheinz Meier and Mihai A Petrovici (Sept. 2021). 'Fast and energy-efficient neuromorphic deep learning with first-spike times'. In: *Nature Machine Intelligence* 3.9, pp. 823–835. DOI: `10.1038/s42256-021-00388-x`

Yannik Stradmann*, **Julian Göltz***, Mihai A. Petrovici, Johannes Schemmel and Sebastian Billaudelle (2025). 'Lu.i – A low-cost electronic neuron for education and outreach'. In: *Trends in Neuroscience and Education* 38, p. 100248. DOI: `10.1016/j.tine.2025.100248`

Julian Göltz*, Jimmy Weber*, Laura Kriener*, Sebastian Billaudelle, Peter Lake, Johannes Schemmel, Melika Payvand and Mihai A. Petrovici (Sept. 2025). 'DelGrad: Exact Event-Based Gradients for Training Delays and Weights on Spiking Neuromorphic Hardware'. In: *Nature Communications* 16.1. DOI: `10.1038/s41467-025-63120-y`

Peer-reviewed publications as co-author

Laura Kriener, **Julian Göltz** and Mihai A. Petrovici (Mar. 2022). 'The Yin-Yang Dataset'. In: *Neuro-Inspired Computational Elements Conference*. NICE 2022. Virtual Event, USA: Association for Computing Machinery, pp. 107–111. DOI: `10.1145/3517343.3517380`

Friedemann Zenke, Sander M. Bohté, Claudia Clopath, Iulia M. Comşa, **Julian Göltz**, Wolfgang Maass, Timothée Masquelier, Richard Naud, Emre O. Neftci, Mihai A. Petrovici, Franz Scherr and Dan F.M. Goodman (Feb. 2021). 'Visualizing a joint future of neuroscience and neuromorphic engineering'. In: *Neuron* 109.4, pp. 571–575. DOI: `10.1016/j.neuron.2021.01.009`

[31]First authors marked with $*$.

Eric Müller, Elias Arnold, Oliver Breitwieser, Milena Czierlinski, Arne Emmel, Jakob Kaiser, Christian Mauch, Sebastian Schmitt, Philipp Spilger, Raphael Stock, Yannik Stradmann, Johannes Weis, Andreas Baumbach, Sebastian Billaudelle, Benjamin Cramer, Falk Ebert, **Julian Göltz**, Joscha Ilmberger, Vitali Karasenko, Mitja Kleider, Aron Leibfried, Christian Pehle and Johannes Schemmel (May 2022). 'A Scalable Approach to Modeling on Accelerated Neuromorphic Hardware'. In: *Frontiers in Neuroscience* 16. DOI: 10.3389/fnins.2022.884128

Preprints

Julian Göltz*, Sebastian Billaudelle*, Laura Kriener*, Luca Blessing, Christian Pehle, Eric Müller, Johannes Schemmel and Mihai A. Petrovici (2023). *Gradient-based methods for spiking physical systems.* arXiv: 2309.10823 [q-bio.NC]. DOI: 10.48550/arXiv.2309.10823

Jonas Oberste-Frielinghaus, Anno C. Kurth, **Julian Göltz**, Laura Kriener, Junji Ito, Mihai A. Petrovici and Sonja Grün (2025). *Synchronization and semantization in deep spiking networks.* arXiv: 2508.12975 [q-bio.NC]. DOI: 10.48550/arXiv.2508.12975

Conferences

In the following, the presentations at conferences are grouped by publications.

Göltz et al. (2021) was presented as a poster at the *Neuroscience to Artificially Intelligent Systems (NaiSys) 2020* and at the *Cosyne 2020* in Colorado (in absence of the author). Furthermore, we presented it in talks at the *Spiking neural networks as universal function approximators (SNUFA): Learning algorithms and applications* 2020, at the *Neuro Inspired Computational Elements (NICE) Conference 2021*, at the *Advanced Course and Symposium on Artificial Intelligence & Neuroscience (ACAIN 2021)* in Grasmere, at the *Bernstein Satellite Workshop: Neuroscience and Artificial Intelligence 2021*, at the *virtual NEST Conference 2022* in Jülich, at the *YRC Structures Conference 2022* in Heidelberg, at the *CNRS AISSAI workshop 'Energetics of computation in artificial and natural networks' 2022* in Paris, and at the *Human Brain Project Summit 2023* in Marseille.

Göltz et al. (2023) was presented as a poster at the workshop *Frontiers of Neuromorphic Computing 2023* in Erlangen, at the International conference on *neuromorphic, natural and physical computing (NNPC) 2023* in Hanover, and at the *Cosyne 2024* in Lisbon.

Göltz et al. (2025) was presented as a poster at the *Bernstein conference 2024* in Frankfurt, at the *Spiking neural networks as universal function approximators (SNUFA 2024)*, at the *International Conference on Artificial Neural Networks (ICANN) 2025* in Kaunas (in absence of the author), as well as in a talk at the *Annual Computational Neuroscience Meeting (CNS) 2025* in Florence in the workshop *Brains and AI*.

Bibliography

Abbott, Larry F. (Nov. 1999). 'Lapicque's Introduction of the Integrate-and-Fire Model Neuron (1907)'. In: *Brain Research Bulletin* 50.5. DOI: 10.1016/S0361-9230(99)00161-6.

Abbott, Larry F. (Nov. 2008). 'Theoretical Neuroscience Rising'. In: *Neuron* 60.3. DOI: 10.1016/j.neuron.2008.10.019.

Abeles, Moshe (1982). *Local Cortical Circuits: An Electrophysiological Study.* Springer Berlin Heidelberg. DOI: 10.1007/978-3-642-81708-3.

Abeles, Moshe (Feb. 1991). *Corticonics: Neural Circuits of the Cerebral Cortex.* Cambridge University Press. DOI: 10.1017/cbo9780511574566.

Abidin, Shafiq (Nov. 2024). *Solar coating, in-motor brakes: here's some of the tech Mercedes wants in its cars by 2040.* Accessed: 2025-11-16. URL: https://www.topgear.com/car-news/future-tech/solar-coating-motor-brakes-heres-some-tech-mercedes-wants-its-cars-2040.

Adrian, Edgar Douglas (1928). *The basis of sensation.* WW Norton & Co.

Adrian, Edgar Douglas and Yngve Zotterman (Apr. 1926). 'The Impulses Produced by Sensory Nerve-Endings'. In: *The Journal of Physiology* 61.2. DOI: 10.1113/jphysiol.1926.sp002281.

Aldrich, Richard W. (June 2001). 'Fifty years of inactivation'. In: *Nature* 411.6838. DOI: 10.1038/35079705.

Amos, Giulia, Stephan J. Ihle et al. (May 2024). 'Engineering an in vitro retinothalamic nerve model'. In: *Frontiers in Neuroscience* 18. DOI: 10.3389/fnins.2024.1396966.

Amos, Giulia, Vaiva Vasiliauskaitė et al. (June 2025). *An Integrated In Vitro Platform and Biophysical Modeling Approach for Studying Synaptic Transmission in Isolated Neuronal Pairs.* bioRxiv: 2025.06.04.657933. DOI: 10.1101/2025.06.04.657933.

Andrade-Talavera, Yuniesky, André Fisahn and Antonio Rodríguez-Moreno (2023). 'Timing to be precise? An overview of spike timing-dependent plasticity, brain rhythmicity, and glial cells interplay within neuronal circuits'. In: *Molecular Psychiatry* 28.6. DOI: 10.1038/s41380-023-02027-w.

Antal, Andrea, Szabolcs Kéri et al. (Jan. 2000). 'Early and late components of visual categorization: an event-related potential study'. In: *Cognitive Brain Research* 9.1. DOI: 10.1016/s0926-6410(99)00053-1.

Antic, Srdjan D., Wen-Liang Zhou et al. (June 2010). 'The decade of the dendritic NMDA spike'. In: *Journal of Neuroscience Research* 88.14. DOI: 10.1002/jnr.22444.

Armstrong, Clay M. and Francisco Bezanilla (Nov. 1977). 'Inactivation of the sodium channel. II. Gating current experiments.' In: *The Journal of general physiology* 70.5. DOI: 10.1085/jgp.70.5.567.

Arnaud Yarga, Sidi Yaya and Sean U. N. Wood (Mar. 2025). 'Accelerating spiking neural networks with parallelizable leaky integrate-and-fire neurons*'. In: *Neuromorphic Computing and Engineering* 5.1. DOI: 10.1088/2634-4386/adb7fe.

Arnold, Elias, Eike-Manuel Edelmann et al. (Mar. 2025). ‘Short-reach Optical Communications: A Real-world Task for Neuromorphic Hardware’. In: *2025 Neuro Inspired Computational Elements (NICE)*. IEEE. DOI: 10.1109/nice65350.2025.11065780.

Aru, Jaan, Mototaka Suzuki and Matthew E. Larkum (Oct. 2020). ‘Cellular Mechanisms of Conscious Processing’. In: *Trends in Cognitive Sciences* 24.10. DOI: 10.1016/j.tics.2020.07.006.

Aswani, Nishant Suresh and Saif Eddin Jabari (2025). *Koopman Autoencoders Learn Neural Representation Dynamics*. URL: https://arxiv.org/abs/2505.12809. arXiv: 2505.12809 [cs.LG].

Attneave, Fred (1954). ‘Some informational aspects of visual perception.’ In: *Psychological Review* 61.3. DOI: 10.1037/h0054663.

Ayzenshtat, Inbal, Elhanan Meirovithz et al. (Aug. 2010). ‘Precise Spatiotemporal Patterns among Visual Cortical Areas and Their Relation to Visual Stimulus Processing’. In: *The Journal of Neuroscience* 30.33. DOI: 10.1523/jneurosci.5177-09.2010.

Bacho, Florian and Dominique Chu (Sept. 2023). ‘Exploring Trade-Offs in Spiking Neural Networks’. In: *Neural Computation* 35.10. DOI: 10.1162/neco_a_01609.

Bahdanau, Dzmitry, Kyunghyun Cho and Yoshua Bengio (2016). *Neural Machine Translation by Jointly Learning to Align and Translate*. URL: https://arxiv.org/abs/1409.0473. arXiv: 1409.0473 [cs.CL].

Barlow, Horace Basil (1961). ‘Possible principles underlying the transformation of sensory messages’. In: *Sensory Communication*. MIT Press. DOI: 10.7551/mitpress/9780262518420.003.0013.

Barral, Jérémie, Xiao-Jing Wang and Alex D. Reyes (Sept. 2019). ‘Propagation of temporal and rate signals in cultured multilayer networks’. In: *Nature Communications* 10.1. DOI: 10.1038/s41467-019-11851-0.

Basu, Arindam, Lei Deng et al. (Apr. 2022). ‘Spiking Neural Network Integrated Circuits: A Review of Trends and Future Directions’. In: *2022 IEEE Custom Integrated Circuits Conference (CICC)*. IEEE. DOI: 10.1109/cicc53496.2022.9772783.

Bekkers, John M. (Dec. 2011). ‘Pyramidal Neurons’. In: *Current Biology* 21.24. DOI: 10.1016/j.cub.2011.10.037.

Bellec, Guillaume, David Kappel et al. (2018). ‘Deep Rewiring: Training very sparse deep networks’. In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=BJ_wN01C-. arXiv: 1711.05136 [cs.NE].

Bellec, Guillaume, Franz Scherr et al. (July 2020). ‘A Solution to the Learning Dilemma for Recurrent Networks of Spiking Neurons’. In: *Nature Communications* 11.1. DOI: 10.1038/s41467-020-17236-y.

Béna, Gabriel, Timo Wunderlich et al. (Mar. 2025). ‘Event-based backpropagation on the neuromorphic platform SpiNNaker2’. In: *2025 Neuro Inspired Computational Elements (NICE)*. IEEE. DOI: 10.1109/nice65350.2025.11065716.

Bengtsson, Sara L., Zoltán Nagy et al. (Aug. 2005). ‘Extensive piano practicing has regionally specific effects on white matter development’. In: *Nature Neuroscience* 8.9. DOI: 10.1038/nn1516.

Beniaguev, David, Idan Segev and Michael London (Sept. 2021). ‘Single cortical neurons as deep artificial neural networks’. In: *Neuron* 109.17. DOI: 10.1016/j.neuron.2021.07.002.

Benjamin, Ben Varkey, Peiran Gao et al. (May 2014). ‘Neurogrid: A Mixed-Analog-Digital Multichip System for Large-Scale Neural Simulations’. In: *Proceedings of the IEEE* 102.5. DOI: 10.1109/jproc.2014.2313565.

Bentrop, Detlef, Michael Beyermann et al. (Nov. 2001). ‘NMR Structure of the “Ball-and-chain” Domain of KCNMB2, the β2-Subunit of Large Conductance Ca2+- and Voltage-activated Potassium Channels’. In: *Journal of Biological Chemistry* 276.45. DOI: 10.1074/jbc.m107118200.

Berry, Michael J., David K. Warland and Markus Meister (May 1997). ‘The structure and precision of retinal spike trains’. In: *Proceedings of the National Academy of Sciences* 94.10. DOI: 10.1073/pnas.94.10.5411.

Bi, Guo-qiang and Mu-ming Poo (Dec. 1998). ‘Synaptic Modifications in Cultured Hippocampal Neurons: Dependence on Spike Timing, Synaptic Strength, and Postsynaptic Cell Type’. In: *The Journal of Neuroscience* 18.24. DOI: 10.1523/jneurosci.18-24-10464.1998.

Bienenstock, Élie L., Leon N. Cooper and Paul W. Munro (Jan. 1982). ‘Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex’. In: *The Journal of Neuroscience* 2.1. DOI: 10.1523/jneurosci.02-01-00032.1982.

Billaudelle, Sebastian (2022). ‘From transistors to learning systems: Circuits and algorithms for brain-inspired computing’. Dissertation. Heidelberg, Germany: Heidelberg University.

Billaudelle, Sebastian, Benjamin Cramer et al. (Jan. 2021). ‘Structural Plasticity on an Accelerated Analog Neuromorphic Hardware System’. In: *Neural Networks* 133. DOI: 10.1016/j.neunet.2020.09.024.

Billaudelle, Sebastian, Yannik Stradmann et al. (Oct. 2020). ‘Versatile Emulation of Spiking Neural Networks on an Accelerated Neuromorphic Substrate’. In: *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. DOI: 10.1109/ISCAS45731.2020.9180741.

Billaudelle, Sebastian, Johannes Weis et al. (Oct. 2022). ‘An Accurate and Flexible Analog Emulation of AdEx Neuron Dynamics in Silicon’. In: DOI: 10.1109/icecs202256217.2022.9971058.

Billeh, Yazan N., Binghuang Cai et al. (May 2020). ‘Systematic Integration of Structural and Functional Data into Multi-scale Models of Mouse Primary Visual Cortex’. In: *Neuron* 106.3. DOI: 10.1016/j.neuron.2020.01.040.

Blessing, Luca (2023). ‘Gradient Estimation With Sparse Observations for Analog Neuromorphic Hardware’. Master’s thesis. Universität Heidelberg.

Boahen, Kwabena (Mar. 2017). ‘A neuromorph’s prospectus’. In: *Computing in Science &amp; Engineering* 19.2. DOI: 10.1109/mcse.2017.33.

Boccato, Tommaso, Dmitrii Zendrikov et al. (May 2025). ‘Genetic Motifs as a Blueprint for Mismatch-Tolerant Neuromorphic Computing’. In: *2025 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE. DOI: 10.1109/iscas56072.2025.11043755.

Bohr, Mark (2007). ‘A 30 Year Retrospective on Dennard’s MOSFET Scaling Paper’. In: *IEEE Solid-State Circuits Newsletter* 12.1. DOI: 10.1109/n-ssc.2007.4785534.

Bohte, Sander M., Joost N. Kok and Han La Poutré (Oct. 2002). ‘Error-backpropagation in temporally encoded networks of spiking neurons’. In: *Neurocomputing* 48.1–4. DOI: 10.1016/s0925-2312(01)00658-0.

Bohte, Sander M., Joost N. Kok and Johannes A. La Poutré (2000). ‘SpikeProp: backpropagation for networks of spiking neurons.’ In: *ESANN*.

Bolte, Jérôme and Edouard Pauwels (2020). ‘A mathematical model for automatic differentiation in machine learning’. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato et al. Vol. 33. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/7a674153c63cff1ad7f0e261c369ab2c-Paper.pdf.

Borges, Jorge Luis (1975). ‘On Exactitude in Science’. In: *A Universal History of Infamy*. Trans. by Norman Thomas di Giovanni. London: Penguin Books. ISBN: 0-14-003959-7.

BRAIN Initiative Cell Census Network (BICCN), Edward M. Callaway et al. (Oct. 2021). ‘A multimodal cell census and atlas of the mammalian primary motor cortex’. In: *Nature* 598.7879. DOI: 10.1038/s41586-021-03950-0.

Branco, Tiago and Kevin Staras (May 2009). ‘The probability of neurotransmitter release: variability and feedback control at single synapses’. In: *Nature Reviews Neuroscience* 10.5. DOI: 10.1038/nrn2634.

Brette, Romain (Oct. 2007). ‘Exact Simulation of Integrate-and-Fire Models with Exponential Currents’. In: *Neural Computation* 19.10. DOI: 10.1162/neco.2007.19.10.2604.

Brette, Romain and Wulfram Gerstner (Nov. 2005). ‘Adaptive Exponential Integrate-and-Fire Model as an Effective Description of Neuronal Activity’. In: *Journal of Neurophysiology* 94.5. DOI: 10.1152/jn.00686.2005.

Brody, Carlos D. and John J. Hopfield (Mar. 2003). ‘Simple Networks for Spike-Timing-Based Computation, with Application to Olfactory Processing’. In: *Neuron* 37.5. DOI: 10.1016/s0896-6273(03)00120-x.

Brunel, Nicolas and Mark C. W. van Rossum (Dec. 2007a). ‘Lapicque’s 1907 Paper: From Frogs to Integrate-and-Fire’. In: *Biological Cybernetics* 97.5. DOI: 10.1007/s00422-007-0190-0.

Brunel, Nicolas and Mark C. W. van Rossum (Dec. 2007b). ‘Quantitative Investigations of Electrical Nerve Excitation Treated as Polarization’. In: *Biological Cybernetics* 97.5. DOI: 10.1007/s00422-007-0189-6.

Buhler, Fred N., Peter Brown et al. (June 2017). ‘A 3.43TOPS/W 48.9pJ/pixel 50.1nJ/classification 512 analog neuron sparse coding neural network with on-chip learning and classification in 40nm CMOS’. In: *2017 Symposium on VLSI Circuits*. IEEE. DOI: 10.23919/vlsic.2017.8008536.

Burke, Sara N. and Carol A. Barnes (Jan. 2006). ‘Neural plasticity in the ageing brain’. In: *Nature Reviews Neuroscience* 7.1. DOI: 10.1038/nrn1809.

Cao, Yongqiang, Yang Chen and Deepak Khosla (Nov. 2014). ‘Spiking Deep Convolutional Neural Networks for Energy-Efficient Object Recognition’. In: *International Journal of Computer Vision* 113.1. DOI: 10.1007/s11263-014-0788-3.

Cardano, Girolamo (1545). *Artis Magnae, Sive de Regulis Algebraicis, Lib. unus*. Reprinted and translated in *Cardano: Great art or rules of algebra*. Trans. by T. Richard Witmer. London, England: MIT Press, Dec. 1968. ISBN: 9780262030250.

Carr, Catherine E. and Masakazu Konishi (Oct. 1990). ‘A circuit for detection of interaural time differences in the brain stem of the barn owl’. In: *The Journal of Neuroscience* 10.10. DOI: 10.1523/jneurosci.10-10-03227.1990.

Chamberlain, Kelly A., Kristen S. Chapey et al. (Jan. 2017). ‘Creatine Enhances Mitochondrial-Mediated Oligodendrocyte Survival After Demyelinating Injury’. In: *The Journal of Neuroscience* 37.6. DOI: 10.1523/jneurosci.1941-16.2016.

Chen, Ricky T. Q., Yulia Rubanova et al. (2018). ‘Neural Ordinary Differential Equations’. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach et al. Vol. 31. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf.

Cohen, Ariel (May 2024). ‘AI Is Pushing The World Toward An Energy Crisis’. In: *Forbes*. URL: https://www.forbes.com/sites/arielcohen/2024/05/23/ai-is-pushing-the-world-towards-an-energy-crisis/ (visited on 20/10/2025).

Comşa, Iulia-Maria, Thomas Fischbacher et al. (May 2020). 'Temporal coding in spiking neural networks with alpha synaptic function'. In: IEEE. DOI: 10.1109/icassp40776.2020.9053856.

Comşa, Iulia-Maria, Krzysztof Potempa et al. (Oct. 2022). 'Temporal Coding in Spiking Neural Networks With Alpha Synaptic Function: Learning With Backpropagation'. In: *IEEE Transactions on Neural Networks and Learning Systems* 33.10. DOI: 10.1109/tnnls.2021.3071976.

Cramer, Benjamin, Sebastian Billaudelle et al. (Jan. 2022). 'Surrogate Gradients for Analog Neuromorphic Computing'. In: *Proceedings of the National Academy of Sciences* 119.4. DOI: 10.1073/pnas.2109194119.

Cramer, Benjamin, Yannik Stradmann et al. (July 2022). 'The Heidelberg Spiking Data Sets for the Systematic Evaluation of Spiking Neural Networks'. In: *IEEE Transactions on Neural Networks and Learning Systems* 33.7. DOI: 10.1109/TNNLS.2020.3044364.

Crick, Francis (Jan. 1989). 'The Recent Excitement about Neural Networks'. In: *Nature* 337.6203. DOI: 10.1038/337129a0.

D'Agostino, Simone, Filippo Moro et al. (Apr. 2024). 'DenRAM: Neuromorphic Dendritic Architecture with RRAM for Efficient Temporal Processing with Delays'. In: *Nature Communications* 15.1. DOI: 10.1038/s41467-024-47764-w.

Dale, Henry (Jan. 1935). 'Pharmacology and Nerve-Endings'. In: *Proceedings of the Royal Society of Medicine* 28.3. DOI: 10.1177/003591573502800330.

Davies, Mike (Sept. 2019). 'Benchmarks for progress in neuromorphic computing'. In: *Nature Machine Intelligence* 1.9. DOI: 10.1038/s42256-019-0097-1.

Davies, Mike, Narayan Srinivasa et al. (Jan. 2018). 'Loihi: A neuromorphic manycore processor with on-chip learning'. In: *IEEE Micro* 38.1. DOI: 10.1109/mm.2018.112130359.

Davies, Mike, Andreas Wild et al. (May 2021). 'Advancing Neuromorphic Computing With Loihi: A Survey of Results and Outlook'. In: *Proceedings of the IEEE* 109.5. DOI: 10.1109/jproc.2021.3067593.

Davison, Andrew P. (2008). 'PyNN: a common interface for neuronal network simulators'. In: *Frontiers in Neuroinformatics* 2. DOI: 10.3389/neuro.11.011.2008.

Dawkins, Richard (Sept. 2009). *The greatest show on earth. The Evidence for Evolution.* London, England: Bantam Press.

Dayan, Peter and Larry F. Abbott (2001). *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems.* Computational Neuroscience. Cambridge, Mass: Massachusetts Institute of Technology Press. ISBN: 978-0-262-04199-7.

De Solla Price, Derek (1974). 'Gears from the Greeks. The Antikythera Mechanism: A Calendar Computer from ca. 80 B. C.' In: *Transactions of the American Philosophical Society* 64.7. DOI: 10.2307/1006146.

Deaner, Robert O., Karin Isler et al. (2007). 'Overall Brain Size, and Not Encephalization Quotient, Best Predicts Cognitive Ability across Non-Human Primates'. In: *Brain, Behavior and Evolution* 70.2. DOI: 10.1159/000102973.

DeBole, Michael V., Brian Taba et al. (May 2019). 'TrueNorth: Accelerating From Zero to 64 Million Neurons in 10 Years'. In: *Computer* 52.5. DOI: 10.1109/mc.2019.2903009.

DeepSeek-AI, Aixin Liu et al. (2025). *DeepSeek-V3 Technical Report.* URL: https://arxiv.org/abs/2412.19437. arXiv: 2412.19437 [cs.CL].

Del Castillo, Jose and Bernard Katz (June 1954). 'Quantal components of the end-plate potential'. In: *The Journal of Physiology* 124.3. DOI: 10.1113/jphysiol.1954.sp005129.

Dennard, Robert H., Fritz H. Gaensslen et al. (Oct. 1974). 'Design of ion-implanted MOSFETs with very small physical dimensions'. In: *IEEE Journal of Solid-State Circuits* 9.5. DOI: 10.1109/jssc.1974.1050511.

Devlin, Jacob, Ming-Wei Chang et al. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* URL: https://arxiv.org/abs/1810.04805. arXiv: 1810.04805 [cs.CL].

Diehl, Peter U., Guido Zarrella et al. (Oct. 2016). 'Conversion of artificial recurrent neural networks to spiking neural networks for low-power neuromorphic hardware'. In: *2016 IEEE International Conference on Rebooting Computing (ICRC).* IEEE. DOI: 10.1109/icrc.2016.7738691.

Diesmann, Markus, Marc-Oliver Gewaltig and Ad Aertsen (Dec. 1999). 'Stable propagation of synchronous spiking in cortical neural networks'. In: *Nature* 402.6761. DOI: 10.1038/990101.

Dold, Dominik and Philipp Christian Petersen (2025). 'Causal pieces: analysing and improving spiking neural networks piece by piece'. In: arXiv: 2504.14015 [cs.NE]. DOI: 10.48550/arXiv.2504.14015.

Eccles, John C., Paul Fatt and Kyozo Koketsu (Dec. 1954). 'Cholinergic and inhibitory synapses in a pathway from motor-axon collaterals to motoneurones'. In: *The Journal of Physiology* 126.3. DOI: 10.1113/jphysiol.1954.sp005226.

Ellenberger, Benjamin, Paul Haider et al. (Dec. 2025). 'Backpropagation through space, time and the brain'. In: *Nature Communications* 17.1. DOI: 10.1038/s41467-025-66666-z.

Ermentrout, Bard (July 1996). 'Type I Membranes, Phase Resetting Curves, and Synchrony'. In: *Neural Computation* 8.5. DOI: 10.1162/neco.1996.8.5.979.

Esser, Steve K., Rathinakumar Appuswamy et al. (2015). 'Backpropagation for Energy-Efficient Neuromorphic Computing'. In: *Advances in Neural Information Processing Systems.* Ed. by C. Cortes, N. Lawrence et al. Vol. 28. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2015/file/10a5ab2db37feedfdeaab192ead4ac0e-Paper.pdf.

Feldman, Daniel E. (Aug. 2012). 'The Spike-Timing Dependence of Plasticity'. In: *Neuron* 75.4. DOI: 10.1016/j.neuron.2012.08.001.

Feynman, Richard P. (1988). *Richard Feynman's blackboard at time of his death.* 1.10-29. Caltech Images Collection, Images. California Institute of Technology Archives and Special Collections. Pasadena, CA. URL: https://collections.archives.caltech.edu/repositories/2/archival_objects/106350 (visited on 25/10/2025).

Fino, Elodie and Rafael Yuste (Mar. 2011). 'Dense Inhibitory Connectivity in Neocortex'. In: *Neuron* 69.6. DOI: 10.1016/j.neuron.2011.02.025.

Fontanini, Riccardo, Alessandro Pilotto et al. (July 2024). 'Reducing the spike rate of deep spiking neural networks based on time-encoding'. In: *Neuromorphic Computing and Engineering* 4.3. DOI: 10.1088/2634-4386/ad64fd.

Freeth, T., Y. Bitsakis et al. (Nov. 2006). 'Decoding the ancient Greek astronomical calculator known as the Antikythera Mechanism'. In: *Nature* 444.7119. DOI: 10.1038/nature05357.

Frémaux, Nicolas and Wulfram Gerstner (2016). 'Neuromodulated Spike-Timing-Dependent Plasticity, and Theory of Three-Factor Learning Rules'. In: *Frontiers in Neural Circuits* 9. DOI: 10.3389/fncir.2015.00085.

Frémaux, Nicolas, Henning Sprekeler and Wulfram Gerstner (Oct. 2010). 'Functional requirements for reward-modulated spike-timing-dependent plasticity'. In: *The Journal of Neuroscience* 30.40. DOI: 10.1523/jneurosci.6249-09.2010.

Frenkel, Charlotte (Sept. 2021). 'Sparsity provides a competitive advantage'. In: *Nature Machine Intelligence* 3.9. DOI: 10.1038/s42256-021-00387-y.

Frenkel, Charlotte, David Bol and Giacomo Indiveri (June 2023). 'Bottom-Up and Top-Down Approaches for the Design of Neuromorphic Processing Systems: Tradeoffs and Synergies Between Natural and Artificial Intelligence'. In: *Proceedings of the IEEE* 111.6. DOI: 10.1109/jproc.2023.3273520.

Frenkel, Charlotte, Jean-Didier Legat and David Bol (Oct. 2019). 'MorphIC: A 65-nm 738k-Synapse/mm^2 Quad-Core Binary-Weight Digital Neuromorphic Processor With Stochastic Spike-Driven Online Learning'. In: *IEEE Transactions on Biomedical Circuits and Systems* 13.5. DOI: 10.1109/tbcas.2019.2928793.

Fukushima, Kunihiko (Apr. 1980). 'Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position'. In: *Biological Cybernetics* 36.4. DOI: 10.1007/bf00344251.

Furber, Steve B., Francesco Galluppi et al. (May 2014). 'The SpiNNaker Project'. In: *Proceedings of the IEEE* 102.5. DOI: 10.1109/jproc.2014.2304638.

Gaiarsa, Jean-Luc, Olivier Caillard and Yehezkel Ben-Ari (Nov. 2002). 'Long-term plasticity at GABAergic and glycinergic synapses: mechanisms and functional significance'. In: *Trends in Neurosciences* 25.11. DOI: 10.1016/s0166-2236(02)02269-5.

Galán, Santos, William F. Feehery and Paul I. Barton (Sept. 1999). 'Parametric sensitivity functions for hybrid discrete/continuous systems'. In: *Applied Numerical Mathematics* 31.1. DOI: 10.1016/s0168-9274(98)00125-1.

Gasiunas, Giedrius, Rodolphe Barrangou et al. (Sept. 2012). 'Cas9–crRNA ribonucleoprotein complex mediates specific DNA cleavage for adaptive immunity in bacteria'. In: *Proceedings of the National Academy of Sciences* 109.39. DOI: 10.1073/pnas.1208507109.

Gemini Team, Petko Georgiev et al. (2024). *Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context.* URL: https://arxiv.org/abs/2403.05530. arXiv: 2403.05530 [cs.CL].

Gerstner, Wulfram (2001). 'What is different with spiking neurons?' In: *Plausible neural networks for biological modelling.*

Gerstner, Wulfram, Richard Kempter et al. (Sept. 1996). 'A neuronal learning rule for submillisecond temporal coding'. In: *Nature* 383.6595. DOI: 10.1038/383076a0.

Gerstner, Wulfram, Werner M. Kistler et al. (2014). *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition.* Cambridge, United Kingdom: Cambridge University Press. ISBN: 978-1-107-06083-8.

Gerstner, Wulfram and Richard Naud (Oct. 2009). 'How Good Are Neuron Models?' In: *Science* 326.5951. DOI: 10.1126/science.1181936.

Göltz, Julian, Sebastian Billaudelle et al. (2023). *Gradient-based methods for spiking physical systems.* URL: https://arxiv.org/abs/2309.10823. arXiv: 2309.10823 [q-bio.NC].

Göltz, Julian, Laura Kriener et al. (Sept. 2021). 'Fast and energy-efficient neuromorphic deep learning with first-spike times'. In: *Nature Machine Intelligence* 3.9. DOI: 10.1038/s42256-021-00388-x.

Göltz, Julian, Jimmy Weber et al. (Sept. 2025). 'DelGrad: Exact Event-Based Gradients for Training Delays and Weights on Spiking Neuromorphic Hardware'. In: *Nature Communications* 16.1. DOI: 10.1038/s41467-025-63120-y.

Gonzales, Roberto B., Cynthia J. DeLeon Galvan et al. (2001). 'Distribution of thorny excrescences on CA3 pyramidal neurons in the rat hippocampus'. In: *The Journal of Comparative Neurology* 430.3. DOI: 10.1002/1096-9861(20010212)430:3<357::aid-cne1036>3.0.co;2-k.

Granier, Arno, Katharina A. Wilmes et al. (2025). *Building functional and mechanistic models of cortical computation based on canonical cell type connectivity.* arXiv: 2504.03031 [q-bio.NC]. DOI: 10.48550/arXiv.2504.03031.

Gray, Charles M., Peter König et al. (Mar. 1989). 'Oscillatory responses in cat visual cortex exhibit inter-columnar synchronization which reflects global stimulus properties'. In: *Nature* 338.6213. DOI: 10.1038/338334a0.

Gray, Charles M. and Wolf Singer (Mar. 1989). 'Stimulus-specific neuronal oscillations in orientation columns of cat visual cortex.' In: *Proceedings of the National Academy of Sciences* 86.5. DOI: 10.1073/pnas.86.5.1698.

Grossberg, Stephen (Jan. 1987). 'Competitive learning: From interactive activation to adaptive resonance'. In: *Cognitive science* 11.1. DOI: 10.1016/s0364-0213(87)80025-3.

Gütig, Robert and Haim Sompolinsky (Feb. 2006). 'The tempotron: a neuron that learns spike timing-based decisions'. In: *Nature Neuroscience* 9.3. DOI: 10.1038/nn1643.

Gygax, Julia and Friedemann Zenke (Apr. 2025). 'Elucidating the Theoretical Underpinnings of Surrogate Gradient Learning in Spiking Neural Networks'. In: *Neural Computation* 37.5. DOI: 10.1162/neco_a_01752.

Hahnloser, Richard H. R., Rahul Sarpeshkar et al. (June 2000). 'Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit'. In: *Nature* 405.6789. DOI: 10.1038/35016072.

Haider, Paul, Benjamin Ellenberger et al. (2021). 'Latent Equilibrium: A Unified Learning Theory for Arbitrarily Fast Computation with Arbitrarily Slow Neurons'. In: *Advances in Neural Information Processing Systems.* Vol. 34. Curran Associates, Inc.

Hammouamri, Ilyass, Ismail Khalfaoui-Hassani and Timothée Masquelier (2024). 'Learning Delays in Spiking Neural Networks using Dilated Convolutions with Learnable Spacings'. In: *The Twelfth International Conference on Learning Representations.* URL: https://openreview.net/forum?id=4r2ybzJnmN.

Hansel, David, Germán Mato et al. (Feb. 1998). 'On Numerical Simulations of Integrate-and-Fire Neural Networks'. In: *Neural Computation* 10.2. DOI: 10.1162/089976698300017845.

Hanuschkin, Alexander, Susanne Kunkel et al. (2010). 'A General and Efficient Method for Incorporating Precise Spike Times in Globally Time-Driven Simulations'. In: *Frontiers in Neuroinformatics* 4. DOI: 10.3389/fninf.2010.00113.

Harkness, Jon M. (Dec. 2002). 'In Appreciation. A Lifetime of Connections: Otto Herbert Schmitt, 1913 - 1998'. In: *Physics in Perspective (PIP)* 4.4. DOI: 10.1007/s000160200005.

Hassabis, Demis, Dharshan Kumaran et al. (July 2017). 'Neuroscience-Inspired Artificial Intelligence'. In: *Neuron* 95.2. DOI: 10.1016/j.neuron.2017.06.011.

He, Kaiming, Xiangyu Zhang et al. (June 2016). 'Deep Residual Learning for Image Recognition'. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* Las Vegas, NV, USA: IEEE. DOI: 10.1109/CVPR.2016.90.

Hebb, Donald O. (1949). *The Organization of Behavior.* New York: Wiley & Sons.

Heil, Peter (Aug. 2004). 'First-Spike Latency of Auditory Neurons Revisited'. In: *Current Opinion in Neurobiology* 14.4. DOI: 10.1016/j.conb.2004.07.002.

Herculano-Houzel, Suzana (2009). 'The human brain in numbers: a linearly scaled-up primate brain'. In: *Frontiers in Human Neuroscience* 3. DOI: 10.3389/neuro.09.031.2009.

Herculano-Houzel, Suzana (June 2012). 'The remarkable, yet not extraordinary, human brain as a scaled-up primate brain and its associated cost'. In: *Proceedings of the National Academy of Sciences* 109.supplement_1. DOI: 10.1073/pnas.1201895109.

Hochreiter, Sepp (1991). 'Untersuchungen zu dynamischen neuronalen Netzen'. Diploma thesis. Institut für Informatik, Technische Universität München.

Hochreiter, Sepp, Yoshua Bengio et al. (2001). 'Gradient Flow in Recurrent Nets: The Difficulty of Learning Long-Term Dependencies'. In: *A Field Guide to Dynamical Recurrent Neural Networks*. Ed. by Stefan C. Kremer and John F. Kolen. IEEE Press. DOI: 10.1109/9780470544037.ch14.

Hodgkin, Alan L. and Andrew F. Huxley (Aug. 1952). 'A Quantitative Description of Membrane Current and Its Application to Conduction and Excitation in Nerve'. In: *The Journal of Physiology* 117.4. DOI: https://doi.org/10.1113/jphysiol.1952.sp004764.

Holtmaat, Anthony J. G. D., Joshua T. Trachtenberg et al. (Jan. 2005). 'Transient and Persistent Dendritic Spines in the Neocortex In Vivo'. In: *Neuron* 45.2. DOI: 10.1016/j.neuron.2005.01.003.

Hooker, Sara (Nov. 2021). 'The Hardware Lottery'. In: *Communications of the ACM* 64.12. DOI: 10.1145/3467017.

Hopfield, John J. (July 1995). 'Pattern recognition computation using action potential timing for stimulus representation'. In: *Nature* 376.6535. DOI: 10.1038/376033a0.

Horowitz, Mark (Feb. 2014). '1.1 Computing's energy problem (and what we can do about it)'. In: *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. IEEE. DOI: 10.1109/isscc.2014.6757323.

Householder, Alston S. (June 1941). 'A theory of steady-state activity in nerve-fiber networks: I. Definitions and preliminary lemmas'. In: *The Bulletin of Mathematical Biophysics* 3.2. DOI: 10.1007/bf02478220.

Hu, Hua, Jian Gan and Peter Jonas (Aug. 2014). 'Fast-spiking, parvalbumin + GABAergic interneurons: From cellular design to microcircuit function'. In: *Science* 345.6196. DOI: 10.1126/science.1255263.

Hubel, David H. and Torsten N. Wiesel (Oct. 1959). 'Receptive fields of single neurones in the cat's striate cortex'. In: *The Journal of Physiology* 148.3. DOI: 10.1113/jphysiol.1959.sp006308.

Hubel, David H. and Torsten N. Wiesel (Jan. 1962). 'Receptive Fields, Binocular Interaction and Functional Architecture in the Cat's Visual Cortex'. In: *The Journal of Physiology* 160.1. DOI: 10.1113/jphysiol.1962.sp006837.

Ifrah, Georges (Aug. 2001). *The universal history of computing: From the abacus to the quantum computer*. Nashville, TN: John Wiley & Sons. ISBN: 0-471-39671-0.

Indiveri, Giacomo, Bernabe Linares-Barranco et al. (May 2011). 'Neuromorphic Silicon Neuron Circuits'. In: *Frontiers in Neuroscience* 5. DOI: 10.3389/fnins.2011.00073.

International Energy Agency (2025). *Energy and AI*. Paris. URL: https://www.iea.org/reports/energy-and-ai.

Ioffe, Sergey and Christian Szegedy (2015). *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. URL: https://arxiv.org/abs/1502.03167. arXiv: 1502.03167 [cs.LG].

IPCC, Intergovernmental Panel on Climate Change (June 2023). *Climate Change 2021 – The Physical Science Basis: Working Group I Contribution to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*. Cambridge University Press. DOI: 10.1017/9781009157896.

Ishino, Yoshizumi, Hideo Shinagawa et al. (Dec. 1987). 'Nucleotide sequence of the iap gene, responsible for alkaline phosphatase isozyme conversion in Escherichia coli, and identification of the gene product'. In: *Journal of Bacteriology* 169.12. DOI: 10.1128/jb.169.12.5429-5433.1987.

Izhikevich, Eugene M. (Sept. 2004). 'Which Model to Use for Cortical Spiking Neurons?' In: *IEEE Transactions on Neural Networks* 15.5. DOI: 10.1109/TNN.2004.832719.

Izhikevich, Eugene M. (Feb. 2006). 'Polychronization: Computation with Spikes'. In: *Neural Computation* 18.2. DOI: 10.1162/089976606775093882.

Ji, Ziwei, Nayeon Lee et al. (Mar. 2023). 'Survey of Hallucination in Natural Language Generation'. In: *ACM Computing Surveys* 55.12. DOI: 10.1145/3571730.

Jia, Junteng and Austin R. Benson (2019). 'Neural Jump Stochastic Differential Equations'. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle et al. Vol. 32. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/59b1deff341edb0b76ace57820cef237-Paper.pdf.

Jinek, Martin, Krzysztof Chylinski et al. (Aug. 2012). 'A Programmable Dual-RNA–Guided DNA Endonuclease in Adaptive Bacterial Immunity'. In: *Science* 337.6096. DOI: 10.1126/science.1225829.

Johansson, Roland S. and Ingvars Birznieks (Jan. 2004). 'First spikes in ensembles of human tactile afferents code complex spatial fingertip events'. In: *Nature Neuroscience* 7.2. DOI: 10.1038/nn1177.

Jordan, Jakob, Maximilian Schmidt et al. (Oct. 2021). 'Evolving interpretable plasticity for spiking networks'. In: *eLife* 10. DOI: 10.7554/elife.66273.

Jouppi, Norman P., Doe Hyun Yoon et al. (June 2021). 'Ten Lessons From Three Generations Shaped Google's TPUv4i: Industrial Product'. In: *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. IEEE. DOI: 10.1109/isca52012.2021.00010.

Jumper, John, Richard Evans et al. (Aug. 2021). 'Highly Accurate Protein Structure Prediction with AlphaFold'. In: *Nature* 596.7873. DOI: 10.1038/s41586-021-03819-2.

Kagan, Brett J., Andy C. Kitchen et al. (Dec. 2022). 'In vitro neurons learn and exhibit sentience when embodied in a simulated game-world'. In: *Neuron* 110.23. DOI: 10.1016/j.neuron.2022.09.001.

Karnani, Mahesh M., Masakazu Agetsuma and Rafael Yuste (June 2014). 'A blanket of inhibition: functional inferences from dense inhibitory connectivity'. In: *Current Opinion in Neurobiology* 26. DOI: 10.1016/j.conb.2013.12.015.

Kempter, Richard, Wulfram Gerstner and J. Leo van Hemmen (Apr. 1999). 'Hebbian learning and spiking neurons'. In: *Physical Review E* 59.4. DOI: 10.1103/physreve.59.4498.

Keuninckx, Lars, Matthias Hartmann et al. (Aug. 2023). 'Not Biologically Inspired: On Training Networks of Monostable Multivibrator Timer Neurons'. In: DOI: 10.36227/techrxiv.24014988.v1.

Khan, Hassan N., David A. Hounshell and Erica R. H. Fuchs (Jan. 2018). 'Science and research policy at the end of Moore's law'. In: *Nature Electronics* 1.1. DOI: 10.1038/s41928-017-0005-9.

Kiefer, Jack and Jacob Wolfowitz (Sept. 1952). 'Stochastic Estimation of the Maximum of a Regression Function'. In: *The Annals of Mathematical Statistics* 23.3. DOI: 10.1214/aoms/1177729392.

Kingma, Diederik P. and Jimmy Ba (2014). 'Adam: A Method for Stochastic Optimization'. In: arXiv: 1412.6980.

Klos, Christian and Raoul-Martin Memmesheimer (Jan. 2025). 'Smooth Exact Gradient Descent Learning in Spiking Neural Networks'. In: *Physical Review Letters* 134.2. DOI: 10.1103/physrevlett.134.027301.

Knight, James C. and Thomas Nowotny (Apr. 2023). 'Easy and efficient spike-based Machine Learning with mlGeNN'. In: *Neuro-Inspired Computational Elements Conference*. NICE 2023. ACM. DOI: 10.1145/3584954.3585001.

Kobatake, Eucaly and Keiji Tanaka (Mar. 1994). 'Neuronal selectivities to complex object features in the ventral visual pathway of the macaque cerebral cortex'. In: *Journal of Neurophysiology* 71.3. DOI: 10.1152/jn.1994.71.3.856.

Kobayashi, Ryota (2009). 'Made-to-order spiking neuron model equipped with a multi-timescale adaptive threshold'. In: *Frontiers in Computational Neuroscience* 3. DOI: 10.3389/neuro.10.009.2009.

Koch, Christof and Idan Segev (Nov. 2000). 'The role of single neurons in information processing'. In: *Nature Neuroscience* 3.S11. DOI: 10.1038/81444.

Kriener, Laura, Julian Göltz and Mihai A. Petrovici (Mar. 2022). 'The Yin-Yang Dataset'. In: *Neuro-Inspired Computational Elements Conference*. NICE 2022. Virtual Event, USA: Association for Computing Machinery. DOI: 10.1145/3517343.3517380.

Krizhevsky, Alex, Ilya Sutskever and Geoffrey E. Hinton (2012). 'Imagenet classification with deep convolutional neural networks'. In: *Advances in neural information processing systems*.

Küchler, Joël, Katarina Vulić et al. (Apr. 2025). 'Engineered biological neuronal networks as basic logic operators'. In: *Frontiers in Computational Neuroscience* 19. DOI: 10.3389/fncom.2025.1559936.

Kumar, Arvind, Sven Schrader et al. (Jan. 2008). 'The High-Conductance State of Cortical Networks'. In: *Neural Computation* 20.1. DOI: 10.1162/neco.2008.20.1.1.

Lake, Peter Simon Paul (2025). 'Training Delays in Spiking Neural Networks on Neuromorphic Hardware'. Master's thesis. Universität Heidelberg.

Lamme, Victor A. F. and Pieter R. Roelfsema (Nov. 2000). 'The distinct modes of vision offered by feedforward and recurrent processing'. In: *Trends in Neurosciences* 23.11. DOI: 10.1016/s0166-2236(00)01657-x.

Lapicque, Louis (1907). 'Recherches Quatitatives Sur l'excitation Electrique Des Nerfs Traitee Comme Polarisation'. In: *Journal de Physiologie et de Pathologie Générale* 9.

Larkum, Matthew E., J. Julius Zhu and Bert Sakmann (Mar. 1999). 'A New Cellular Mechanism for Coupling Inputs Arriving at Different Cortical Layers'. In: *Nature* 398.6725. DOI: 10.1038/18686.

Latham, Peter E., Barry J. Richmond et al. (Feb. 2000). 'Intrinsic Dynamics in Neuronal Networks. I. Theory'. In: *Journal of Neurophysiology* 83.2. DOI: 10.1152/jn.2000.83.2.808.

LeCun, Yann, Bernhard Boser et al. (Dec. 1989). 'Backpropagation Applied to Handwritten Zip Code Recognition'. In: *Neural Computation* 1.4. DOI: 10.1162/neco.1989.1.4.541.

LeCun, Yann, Léon Bottou et al. (Nov. 1998). 'Gradient-Based Learning Applied to Document Recognition'. In: *Proceedings of the IEEE* 86.11. DOI: 10.1109/5.726791.

Lee, Jane H., Saeid Haghighatshoar and Amin Karbasi (Apr. 2023). 'Exact Gradient Computation for Spiking Neural Networks via Forward Propagation'. In: *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*. PMLR.

Lee, Wonyeol, Hangyeol Yu et al. (2020). 'On Correctness of Automatic Differentiation for Non-Differentiable Functions'. In: *Advances in Neural Information Processing Systems*. Ed. by H.

Larochelle, M. Ranzato et al. Vol. 33. Curran Associates, Inc. URL: `https://proceedings.neurips.cc/paper_files/paper/2020/file/4aaa76178f8567e05c8e8295c96171d8-Paper.pdf`.

Legenstein, Robert, Christian Naeger and Wolfgang Maass (Nov. 2005). 'What Can a Neuron Learn with Spike-Timing-Dependent Plasticity?' In: *Neural Computation* 17.11. DOI: `10.1162/0899766054796888`.

Leiserson, Charles E., Neil C. Thompson et al. (June 2020). 'There's plenty of room at the Top: What will drive computer performance after Moore's law?' In: *Science* 368.6495. DOI: `10.1126/science.aam9744`.

Lennie, Peter (Mar. 2003). 'The Cost of Cortical Computation'. In: *Current Biology* 13.6. DOI: `10.1016/S0960-9822(03)00135-0`.

Levy, William B. and Victoria G. Calvert (Apr. 2021). 'Communication consumes 35 times more energy than computation in the human cortex, but both costs are needed to predict synapse number'. In: *Proceedings of the National Academy of Sciences* 118.18. DOI: `10.1073/pnas.2008173118`.

Liewald, Daniel, Robert Miller et al. (Aug. 2014). 'Distribution of axon diameters in cortical white matter: an electron-microscopic study on three human brains and a macaque'. In: *Biological Cybernetics* 108.5. DOI: `10.1007/s00422-014-0626-2`.

Lillicrap, Timothy P. and Adam Santoro (Apr. 2019). 'Backpropagation through Time and the Brain'. In: *Current Opinion in Neurobiology*. Machine Learning, Big Data, and Neuroscience 55. DOI: `10.1016/j.conb.2019.01.011`.

Lillicrap, Timothy P., Adam Santoro et al. (Apr. 2020). 'Backpropagation and the Brain'. In: *Nature Reviews Neuroscience* 21.6. DOI: `10.1038/s41583-020-0277-3`.

Lind, Richard E. (Dec. 2006). *The seat of consciousness in ancient literature.* Jefferson, NC: McFarland.

Linnainmaa, Seppo (1970). 'The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors'. MA thesis. University of Helsinki.

Liu, Jia, Karen Dietz et al. (Nov. 2012). 'Impaired adult myelination in the prefrontal cortex of socially isolated mice'. In: *Nature Neuroscience* 15.12. DOI: `10.1038/nn.3263`.

Liu, Xiao-Bo and Cynthia M. Schumann (Apr. 2014). 'Optimization of electron microscopy for human brains with long-term fixation and fixed-frozen sections'. In: *Acta Neuropathologica Communications* 2.1. DOI: `10.1186/2051-5960-2-42`.

Luccioni, Alexandra Sasha, Yacine Jernite and Emma Strubell (June 2024). 'Power Hungry Processing: Watts Driving the Cost of AI Deployment?' In: *The 2024 ACM Conference on Fairness Accountability and Transparency.* FAccT '24. ACM. DOI: `10.1145/3630106.3658542`.

Luccioni, Alexandra Sasha, Sylvain Viguier and Anne-Laure Ligozat (Nov. 2022). *Estimating the Carbon Footprint of BLOOM, a 176B Parameter Language Model.* arXiv: `2211.02001 [cs]`. DOI: `10.48550/arXiv.2211.02001`.

Lüscher, Christian, Roger A. Nicoll et al. (June 2000). 'Synaptic plasticity and dynamic modulation of the postsynaptic membrane'. In: *Nature Neuroscience* 3.6. DOI: `10.1038/75714`.

Lyon, Richard F. and Carver A. Mead (July 1988). 'An analog electronic cochlea'. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 36.7. DOI: `10.1109/29.1639`.

Ma, Jun, Guolin Yu et al. (May 2023). 'Safe semi-supervised learning for pattern classification'. In: *Engineering Applications of Artificial Intelligence* 121. DOI: `10.1016/j.engappai.2023.106021`.

Maass, Wolfgang (Oct. 2016). ‘Searching for principles of brain computation’. In: *Current Opinion in Behavioral Sciences* 11. DOI: `10.1016/j.cobeha.2016.06.003`.

Maass, Wolfgang and Michael Schmitt (Aug. 1999). ‘On the Complexity of Learning for Spiking Neurons with Temporal Coding’. In: *Information and Computation* 153.1. DOI: `10.1006/inco.1999.2806`.

MacLean, Paul D. (Nov. 1949). ‘Psychosomatic Disease and the “Visceral Brain”: Recent Developments Bearing on the Papez Theory of Emotion’. In: *Psychosomatic Medicine* 11.6. DOI: `10.1097/00006842-194911000-00003`.

Magrassi, Lorenzo, Ketty Leto and Ferdinando Rossi (Feb. 2013). ‘Lifespan of neurons is uncoupled from organismal lifespan’. In: *Proceedings of the National Academy of Sciences* 110.11. DOI: `10.1073/pnas.1217505110`.

Mahdavi, Hamed, Alireza Hashemi et al. (2025). *Brains vs. Bytes: Evaluating LLM Proficiency in Olympiad Mathematics*. URL: `https://arxiv.org/abs/2504.01995`. arXiv: `2504.01995 [cs.AI]`.

Mahowald, Misha and Rodney Douglas (Dec. 1991). ‘A silicon neuron’. In: *Nature* 354.6354. DOI: `10.1038/354515a0`.

Mainen, Zachary F. and Terrence J. Sejnowski (June 1995). ‘Reliability of Spike Timing in Neocortical Neurons’. In: *Science* 268.5216. DOI: `10.1126/science.7770778`.

Makinodan, Manabu, Kenneth M. Rosen et al. (Sept. 2012). ‘A Critical Period for Social Experience–Dependent Oligodendrocyte Maturation and Myelination’. In: *Science* 337.6100. DOI: `10.1126/science.1220845`.

Markram, Henry, Joachim Lübke et al. (Jan. 1997). ‘Regulation of Synaptic Efficacy by Coincidence of Postsynaptic APs and EPSPs’. In: *Science* 275.5297. DOI: `10.1126/science.275.5297.213`.

Markram, Henry and Misha Tsodyks (Aug. 1996). ‘Redistribution of synaptic efficacy between neocortical pyramidal neurons’. In: *Nature* 382.6594. DOI: `10.1038/382807a0`.

Marschall, Owen, Kyunghyun Cho and Cristina Savin (2020). ‘A Unified Framework of Online Learning Algorithms for Training Recurrent Neural Networks’. In: *Journal of Machine Learning Research* 21.135. URL: `http://jmlr.org/papers/v21/19-562.html`.

Matsuzaki, Masanori, Graham C. R. Ellis-Davies et al. (Oct. 2001). ‘Dendritic spine geometry is critical for AMPA receptor expression in hippocampal CA1 pyramidal neurons’. In: *Nature Neuroscience* 4.11. DOI: `10.1038/nn736`.

Mattia, Maurizio and Paolo Del Giudice (Oct. 2000). ‘Efficient Event-Driven Simulation of Large Networks of Spiking Neurons and Dynamical Synapses’. In: *Neural Computation* 12.10. DOI: `10.1162/089976600300014953`.

Max, Kevin, Laura Kriener et al. (June 2024). ‘Learning Efficient Backprojections across Cortical Hierarchies in Real Time’. In: *Nature Machine Intelligence* 6.6. DOI: `10.1038/s42256-024-00845-3`.

Mayr, Christian, Sebastian Hoeppner and Steve Furber (2019). *SpiNNaker 2: A 10 Million Core Processor System for Brain Simulation and Machine Learning*. URL: `https://arxiv.org/abs/1911.02385`. arXiv: `1911.02385 [cs.ET]`.

McCulloch, Warren S. and Walter Pitts (Dec. 1943). ‘A Logical Calculus of the Ideas Immanent in Nervous Activity’. In: *The bulletin of mathematical biophysics* 5.4. DOI: `10.1007/BF02478259`.

McDonnell, Mark D., Kwabena Boahen et al. (May 2014). ‘Engineering intelligent electronic systems based on computational neuroscience [scanning the issue]’. In: *Proceedings of the IEEE* 102.5. DOI: `10.1109/jproc.2014.2314776`.

McKenzie, Ian A., David Ohayon et al. (Oct. 2014). 'Motor skill learning requires active central myelination'. In: *Science* 346.6207. DOI: 10.1126/science.1254960.

McMenemey, William H. (July 1963). '" NEURON " OR " NEURONE " ?' In: *The Lancet* 282.7299. DOI: 10.1016/s0140-6736(63)92625-4.

Mead, Carver A. (Jan. 1989). *Analog VLSI and Neural Systems*. Boston, MA: Addison Wesley.

Mead, Carver A. (Oct. 1990). 'Neuromorphic Electronic Systems'. In: *Proceedings of the IEEE* 78.10. DOI: 10.1109/5.58356.

Mead, Carver A. and Misha A. Mahowald (Jan. 1988). 'A Silicon Model of Early Visual Processing'. In: *Neural Networks* 1.1. DOI: 10.1016/0893-6080(88)90024-X.

Mehta, Arpan R., Puja R. Mehta et al. (Dec. 2019). 'Etymology and the neuron(e)'. In: *Brain* 143.1. DOI: 10.1093/brain/awz367.

Mercedes-Benz Group AG (Nov. 2024). *Wegweisende Innovationen für das Automobil der Zukunft. Mercedes-Benz gibt exklusive Einblicke in Forschungsaktivitäten und zukünftige Technologien.* Accessed: 2025-11-16. URL: https://media.mercedes-benz.com/article/e19821db-94b3-4f6f-9226-533f36aff630/(lightbox:document/0f23353e-4ab4-468a-9dc5-92532b1ea4cc).

Mészáros, Balázs, James C. Knight and Thomas Nowotny (Nov. 2025). 'Efficient event-based delay learning in spiking neural networks'. In: *Nature Communications* 16.1. DOI: 10.1038/s41467-025-65394-8.

Minsky, Marvin and Seymour Papert (1969). *Perceptrons. An Introduction to Computational Geometry.* Vol. 479. 480. The MIT Press.

Mirzadeh, Iman, Keivan Alizadeh et al. (2025). *GSM-Symbolic: Understanding the Limitations of Mathematical Reasoning in Large Language Models.* URL: https://arxiv.org/abs/2410.05229. arXiv: 2410.05229 [cs.LG].

Montúfar, Guido, Razvan Pascanu et al. (2014). 'On the Number of Linear Regions of Deep Neural Networks'. In: *Advances in Neural Information Processing Systems.* Vol. 27. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2014/file/fa6f2a469cc4d61a92d96e74617c3d2a-Paper.pdf.

Moore, Gordon E. (Apr. 1965). 'Cramming More Components onto Integrated Circuits'. In: *Electronics.* Reprinted in 'Cramming More Components Onto Integrated Circuits'. In: *Proceedings of the IEEE* 86.1 (Jan. 1998), pp. 82–85. DOI: 10.1109/jproc.1998.658762.

Moradi, Saber, Ning Qiao et al. (Feb. 2018). 'A Scalable Multicore Architecture With Heterogeneous Memory Structures for Dynamic Neuromorphic Asynchronous Processors (DYNAPs)'. In: *IEEE Transactions on Biomedical Circuits and Systems* 12.1. DOI: 10.1109/tbcas.2017.2759700.

Morrison, Abigail, Sirko Straube et al. (Jan. 2007). 'Exact Subthreshold Integration with Continuous Spike Times in Discrete-Time Neural Network Simulations'. In: *Neural Computation* 19.1. DOI: 10.1162/neco.2007.19.1.47.

Mostafa, Hesham (July 2018). 'Supervised Learning Based on Temporal Coding in Spiking Neural Networks'. In: *IEEE Transactions on Neural Networks and Learning Systems* 29.7. DOI: 10.1109/TNNLS.2017.2726060.

Mostafa, Hesham, Bruno U. Pedroni et al. (May 2017). 'Fast classification using sparsely active spiking networks'. In: *2017 IEEE International Symposium on Circuits and Systems (ISCAS).* IEEE. DOI: 10.1109/iscas.2017.8050527.

Müller, Eric, Moritz Althaus et al. (2024). 'jaxsnn: Event-driven Gradient Estimation for Analog Neuromorphic Hardware'. In: *Neuro-inspired Computational Elements Workshop (NICE 2024).* arXiv: 2401.16841 [cs.NE]. DOI: 10.1109/NICE61972.2024.10548709.

Müller, Eric, Elias Arnold et al. (May 2022). 'A Scalable Approach to Modeling on Accelerated Neuromorphic Hardware'. In: *Frontiers in Neuroscience* 16. DOI: 10.3389/fnins.2022.884128.

Naud, Richard, Nicolas Marcille et al. (2008). 'Firing patterns in the adaptive exponential integrate-and-fire model'. In: *Biological cybernetics* 99.4–5. DOI: 10.1007/s00422-008-0264-7.

Neftci, Emre O., Hesham Mostafa and Friedemann Zenke (2019). 'Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks'. In: *IEEE Signal Processing Magazine* 36.6. DOI: 10.1109/MSP.2019.2931595.

Neuman, A. Martina, Dominik Dold and Philipp Christian Petersen (2025). *Stable Learning Using Spiking Neural Networks Equipped With Affine Encoders and Decoders*. URL: https://arxiv.org/abs/2404.04549. arXiv: 2404.04549 [cs.NE].

Niell, Cristopher M. and Michael P. Stryker (July 2008). 'Highly Selective Receptive Fields in Mouse Visual Cortex'. In: *The Journal of Neuroscience* 28.30. DOI: 10.1523/jneurosci.0623-08.2008.

Nowotny, Thomas, James P. Turner and James C. Knight (Jan. 2025). 'Loss shaping enhances exact gradient learning with Eventprop in spiking neural networks'. In: *Neuromorphic Computing and Engineering* 5.1. DOI: 10.1088/2634-4386/ada852.

Oberste-Frielinghaus, Jonas, Anno C. Kurth et al. (2025). *Synchronization and semantization in deep spiking networks*. arXiv: 2508.12975 [q-bio.NC]. DOI: 10.48550/arXiv.2508.12975.

Oh, Kyoung-Su and Keechul Jung (June 2004). 'GPU implementation of neural networks'. In: *Pattern Recognition* 37.6. DOI: 10.1016/j.patcog.2004.01.013.

Oja, Erkki (Nov. 1982). 'Simplified neuron model as a principal component analyzer'. In: *Journal of Mathematical Biology* 15.3. DOI: 10.1007/bf00275687.

Olshausen, Bruno A. and David J. Field (June 1996). 'Emergence of simple-cell receptive field properties by learning a sparse code for natural images'. In: *Nature* 381.6583. DOI: 10.1038/381607a0.

OpenAI, Josh Achiam et al. (2024). *GPT-4 Technical Report*. URL: https://arxiv.org/abs/2303.08774. arXiv: 2303.08774 [cs.CL].

Owen, Richard (June 1839). 'Notes on the Anatomy of the Nubian Giraffe.' In: *The Transactions of the Zoological Society of London* 2.3. DOI: 10.1111/j.1469-7998.1839.tb00021.x.

Palmer, Lucy M., Adam S. Shai et al. (Feb. 2014). 'NMDA spikes enhance action potential generation during sensory input'. In: *Nature Neuroscience* 17.3. DOI: 10.1038/nn.3646.

Pan, Simon, Sonia R. Mayoral et al. (Feb. 2020). 'Preservation of a remote fear memory requires new myelin formation'. In: *Nature Neuroscience* 23.4. DOI: 10.1038/s41593-019-0582-1.

Panzeri, Stefano, Rasmus S. Petersen et al. (Mar. 2001). 'The Role of Spike Timing in the Coding of Stimulus Location in Rat Somatosensory Cortex'. In: *Neuron* 29.3. DOI: 10.1016/s0896-6273(01)00251-3.

Papez, James W. (Oct. 1937). 'A Proposed Mechanism Of Emotion'. In: *Archives of Neurology And Psychiatry* 38.4. DOI: 10.1001/archneurpsyc.1937.02260220069003.

Patiño-Saucedo, Alberto, Amirreza Yousefzadeh et al. (May 2023). 'Empirical study on the efficiency of Spiking Neural Networks with axonal delays, and algorithm-hardware benchmarking'. In: *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE. DOI: 10.1109/iscas46773.2023.10181778.

Pehle, Christian, Sebastian Billaudelle et al. (2022). 'The BrainScaleS-2 accelerated neuromorphic system with hybrid plasticity'. In: *Frontiers in Neuroscience* 16.

Petersen, Rasmus S., Stefano Panzeri and Mathew E. Diamond (Nov. 2001). 'Population Coding of Stimulus Location in Rat Somatosensory Cortex'. In: *Neuron* 32.3. DOI: 10.1016/s0896-6273(01)00481-0.

Petrov, Ivo, Jasper Dekoninck et al. (2025). *Proof or Bluff? Evaluating LLMs on 2025 USA Math Olympiad.* URL: https://arxiv.org/abs/2503.21934. arXiv: 2503.21934 [cs.CL].

Petrovici, Mihai A. (2016). 'Form vs. Function: Theory and Models for Neuronal Substrates'. Dissertation. Heidelberg, Germany: Heidelberg University. DOI: 10.11588/heidok.00019161.

Pfeiffer, Michael and Thomas Pfeil (Oct. 2018). 'Deep Learning With Spiking Neurons: Opportunities and Challenges'. In: *Frontiers in Neuroscience* 12. DOI: 10.3389/fnins.2018.00774.

Pfister, Jean-Pascal, Taro Toyoizumi et al. (June 2006). 'Optimal Spike-Timing-Dependent Plasticity for Precise Action Potential Firing in Supervised Learning'. In: *Neural Computation* 18.6. DOI: 10.1162/neco.2006.18.6.1318.

Philips, Thomas and Jeffrey D. Rothstein (Sept. 2017). 'Oligodendroglia: metabolic supporters of neurons'. In: *Journal of Clinical Investigation* 127.9. DOI: 10.1172/jci90610.

Pierro, Alessandro, Philipp Stratmann et al. (2024). *Solving QUBO on the Loihi 2 Neuromorphic Processor.* URL: https://arxiv.org/abs/2408.03076. arXiv: 2408.03076 [cs.NE].

Pisokas, Ioannis, Stanley Heinze and Barbara Webb (July 2020). 'The head direction circuit of two insect species'. In: *eLife* 9. DOI: 10.7554/elife.53985.

Project Apertus, Alejandro Hernández-Cano et al. (2025). *Apertus: Democratizing Open and Compliant LLMs for Global Language Environments.* URL: https://arxiv.org/abs/2509.14233. arXiv: 2509.14233 [cs.CL].

Queant, Alexandre, Ulysse Rançon et al. (2025). *DelRec: learning delays in recurrent spiking neural networks.* URL: https://arxiv.org/abs/2509.24852. arXiv: 2509.24852 [cs.NE].

Quiroga, R. Quian, Leila Reddy et al. (June 2005). 'Invariant visual representation by single neurons in the human brain'. In: *Nature* 435.7045. DOI: 10.1038/nature03687.

Ramón y Cajal, Santiago (1899). *Comparative Study of the Sensory Areas of the Human Cortex.* Clark University.

Ranvier, Louis-Antoine (1878). *Leçons sur l'histologie du Système Nerveux.* Ed. by E. F. Savy. Paris: Savy.

Rauch, Alexander, Giancarlo La Camera et al. (Sept. 2003). 'Neocortical Pyramidal Cells Respond as Integrate-and-Fire Neurons to In Vivo–Like Input Currents'. In: *Journal of Neurophysiology* 90.3. DOI: 10.1152/jn.00293.2003.

Reich, Daniel S., Ferenc Mechler and Jonathan D. Victor (Mar. 2001). 'Temporal Coding of Contrast in Primary Visual Cortex: When, What, and Why'. In: *Journal of Neurophysiology* 85.3. DOI: 10.1152/jn.2001.85.3.1039.

Renner, Alpha, Lazar Supic et al. (June 2024). 'Neuromorphic Visual Scene Understanding with Resonator Networks'. In: *Nature Machine Intelligence* 6.6. DOI: 10.1038/s42256-024-00848-0.

Reyes, Alex D. (May 2003). 'Synchrony-dependent propagation of firing rate in iteratively constructed networks in vitro'. In: *Nature Neuroscience* 6.6. DOI: 10.1038/nn1056.

Reynen, Joseph (1963). 'The Ephemeris of Double-Star Relative Radial Velocity Calculated on a Pace 231-R Analogue Computer'. In: *Symposium über Automation und Digitalisierung* *in der Astronomischen Meßtechnik am 27. und 28. April 1962 in Tübingen*. Springer Berlin Heidelberg. DOI: 10.1007/978-3-662-22448-9_3.

Richards, Blake Aaron and Konrad Paul Kording (2023). 'The Study of Plasticity Has Always Been about Gradients'. In: *The Journal of Physiology* 601.15. DOI: 10.1113/JP282747.

Richter, Ole, Chenxi Wu et al. (Jan. 2024). 'DYNAP-SE2: a scalable multi-core dynamic neuromorphic asynchronous spiking neural network processor'. In: *Neuromorphic Computing and Engineering* 4.1. DOI: 10.1088/2634-4386/ad1cd7.

Robbins, Herbert and Sutton Monro (Sept. 1951). 'A Stochastic Approximation Method'. In: *The Annals of Mathematical Statistics* 22.3. DOI: 10.1214/aoms/1177729586.

Ropireddy, Deepak, Ruggero Scorcioni et al. (Dec. 2010). 'Axonal morphometry of hippocampal pyramidal neurons semi-automatically reconstructed after in vivo labeling in different CA3 locations'. In: *Brain Structure and Function* 216.1. DOI: 10.1007/s00429-010-0291-8.

Rosenblatt, Frank (1958). 'The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain'. In: *Psychological Review* 65.6. DOI: 10.1037/h0042519.

Rosko, Lauren M., Tyler Gentile et al. (Feb. 2023). 'Cerebral Creatine Deficiency Affects the Timing of Oligodendrocyte Myelination'. In: *The Journal of Neuroscience* 43.7. DOI: 10.1523/jneurosci.2120-21.2022.

Rotter, Stefan and Markus Diesmann (Nov. 1999). 'Exact digital simulation of time-invariant linear systems with applications to neuronal modeling'. In: *Biological Cybernetics* 81.5–6. DOI: 10.1007/s004220050570.

Roy, Kaushik, Akhilesh Jaiswal and Priyadarshini Panda (Nov. 2019). 'Towards spike-based machine intelligence with neuromorphic computing'. In: *Nature* 575.7784. DOI: 10.1038/s41586-019-1677-2.

Rumelhart, David E., Geoffrey E. Hinton and Ronald J. Williams (Oct. 1986). 'Learning Representations by Back-Propagating Errors'. In: *Nature* 323.6088. DOI: 10.1038/323533a0.

Rust, Nicole C. and James J. DiCarlo (Sept. 2010). 'Selectivity and Tolerance ("Invariance") Both Increase as Visual Information Propagates from Cortical Area V4 to IT'. In: *The Journal of Neuroscience* 30.39. DOI: 10.1523/jneurosci.0179-10.2010.

Sacramento, João, Rui Ponte Costa et al. (2018). 'Dendritic Cortical Microcircuits Approximate the Backpropagation Algorithm'. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach et al. Vol. 31. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/1dc3a89d0d440ba31729b0ba74b93a33-Paper.pdf.

Sampaio-Baptista, Cassandra, Alexandre A. Khrapitchev et al. (Dec. 2013). 'Motor Skill Learning Induces Changes in White Matter Microstructure and Myelination'. In: *The Journal of Neuroscience* 33.50. DOI: 10.1523/jneurosci.3048-13.2013.

Saponati, Matteo, Chiara De Luca et al. (Mar. 2025). 'A feedback control optimizer for online and hardware-aware training of Spiking Neural Networks'. In: *2025 Neuro Inspired Computational Elements (NICE)*. IEEE. DOI: 10.1109/nice65350.2025.11065428.

Schemmel, Johannes, Sebastian Billaudelle et al. (Dec. 2021). 'Accelerated Analog Neuromorphic Computing'. In: *Analog Circuits for Machine Learning, Current/Voltage/Temperature Sensors, and High-speed Communication*. Springer International Publishing. DOI: 10.1007/978-3-030-91741-8_6.

Schmitt, Otto H. (Jan. 1938). 'A thermionic trigger'. In: *Journal of Scientific Instruments* 15.1. DOI: 10.1088/0950-7671/15/1/305.

Schmitt, Sebastian, Johann Klähn et al. (May 2017). 'Neuromorphic Hardware in the Loop: Training a Deep Spiking Network on the BrainScaleS Wafer-Scale System'. In: *2017 In-*

ternational Joint Conference on Neural Networks (IJCNN). DOI: 10.1109/IJCNN.2017.7966125.

Scholz, Jan, Miriam C. Klein et al. (Oct. 2009). 'Training induces changes in white-matter architecture'. In: *Nature Neuroscience* 12.11. DOI: 10.1038/nn.2412.

Schreiber, Korbinian (2021). 'Accelerated neuromorphic cybernetics'. Dissertation. Heidelberg, Germany: Heidelberg University.

Schug, Simon, Frederik Benzing and Angelika Steger (Oct. 2021). 'Presynaptic Stochasticity Improves Energy Efficiency and Helps Alleviate the Stability-Plasticity Dilemma'. In: *eLife* 10. Ed. by Timothy E. Behrens, Timothy O'Leary and Jean-Pascal Pfister. DOI: 10.7554/eLife.69884.

Schünke, Michael, Erik Schulte and Udo Schumacher (2012). *Prometheus: Kopf, Hals und Neuroanatomie. Lernatlas der Anatomie.* 3., überarbeitete und erweiterte Auflage. Georg Thieme Verlag.

Schwartz, Roy, Jesse Dodge et al. (Nov. 2020). 'Green AI'. In: *Communications of the ACM* 63.12. DOI: 10.1145/3381831.

Sejnowski, Terrence J. (1977). 'Storing covariance with nonlinearly interacting neurons'. In: *Journal of Mathematical Biology* 4.4. DOI: 10.1007/bf00275079.

Senn, Walter, Dominik Dold et al. (2024). 'A neuronal least-action principle for real-time learning in cortical circuits'. In: *ELife* 12. DOI: 10.7554/eLife.89674.

Serban, Radu and Antonio Recuero (July 2019). 'Sensitivity Analysis for Hybrid Systems and Systems With Memory'. In: *Journal of Computational and Nonlinear Dynamics* 14.9. DOI: 10.1115/1.4044028.

Shen, Siming, Juan Sandoval et al. (Aug. 2008). 'Age-dependent epigenetic control of differentiation inhibitors is critical for remyelination efficiency'. In: *Nature Neuroscience* 11.9. DOI: 10.1038/nn.2172.

Shimizu, Genki and Taro Toyoizumi (2025). *Diverse Neural Sequences in QIF Networks: An Analytically Tractable Framework for Synfire Chains and Hippocampal Replay.* URL: https://arxiv.org/abs/2508.06085. arXiv: 2508.06085 [q-bio.NC].

Shoesmith, Thomas, James C. Knight et al. (2025). *Eventprop training for efficient neuromorphic applications.* URL: https://arxiv.org/abs/2503.04341. arXiv: 2503.04341 [cs.NE].

Shrestha, Sumit Bam and Garrick Orchard (2018). *SLAYER: Spike Layer Error Reassignment in Time.* Ed. by S. Bengio, H. Wallach et al. Curran Associates, Inc. Chap. Advances in Neural Information Processing Systems 31. DOI: 10.48550/arXiv.1810.08646.

Silver, David, Thomas Hubert et al. (Dec. 2018). 'A General Reinforcement Learning Algorithm That Masters Chess, Shogi, and Go through Self-Play'. In: *Science* 362.6419. DOI: 10.1126/science.aar6404.

Silver, David, Julian Schrittwieser et al. (Oct. 2017). 'Mastering the Game of Go without Human Knowledge'. In: *Nature* 550.7676. DOI: 10.1038/nature24270.

Simpson, Ronald W. and Herbert E. Smith (Aug. 1967). 'Apollo rendezvous with command module active'. In: *Control and Flight Dynamics Conference.* American Institute of Aeronautics and Astronautics. DOI: 10.2514/6.1967-562.

Singer, Wolf and Charles M. Gray (Mar. 1995). 'Visual Feature Integration and the Temporal Correlation Hypothesis'. In: *Annual Review of Neuroscience* 18.1. DOI: 10.1146/annurev.ne.18.030195.003011.

Snaidero, Nicolas, Wiebke Möbius et al. (Jan. 2014). 'Myelin Membrane Wrapping of CNS Axons by PI(3,4,5)P3-Dependent Polarized Growth at the Inner Tongue'. In: *Cell* 156.1–2. DOI: 10.1016/j.cell.2013.11.044.

Sohya, Kazuhiro, Katsuro Kameyama et al. (Feb. 2007). 'GABAergic Neurons Are Less Selective to Stimulus Orientation than Excitatory Neurons in Layer II/III of Visual Cortex, as Revealed by In Vivo Functional Ca2+ Imaging in Transgenic Mice'. In: *The Journal of Neuroscience* 27.8. DOI: 10.1523/jneurosci.4641-06.2007.

Sokolov, Boris P. (Feb. 2007). 'Oligodendroglial abnormalities in schizophrenia, mood disorders and substance abuse. Comorbidity, shared traits, or molecular phenocopies?' In: *The International Journal of Neuropsychopharmacology* 10.04. DOI: 10.1017/s1461145706007322.

Song, Sen, Kenneth D. Miller and Larry F. Abbott (Sept. 2000). 'Competitive Hebbian learning through spike-timing-dependent synaptic plasticity'. In: *Nature Neuroscience* 3.9. DOI: 10.1038/78829.

Spalding, Kirsty L., Olaf Bergmann et al. (June 2013). 'Dynamics of Hippocampal Neurogenesis in Adult Humans'. In: *Cell* 153.6. DOI: 10.1016/j.cell.2013.05.002.

Spilger, Philipp, Elias Arnold et al. (Apr. 2023). 'hxtorch.snn: Machine-learning-inspired Spiking Neural Network Modeling on BrainScaleS-2'. In: *Neuro-Inspired Computational Elements Conference*. NICE 2023. ACM. DOI: 10.1145/3584954.3584993.

Stanojevic, Ana, Stanisław Woźniak et al. (Aug. 2024). 'High-performance deep spiking neural networks with 0.3 spikes per neuron'. In: *Nature Communications* 15.1. DOI: 10.1038/s41467-024-51110-5.

Steadman, Patrick E., Frances Xia et al. (Jan. 2020). 'Disruption of Oligodendrogenesis Impairs Memory Consolidation in Adult Mice'. In: *Neuron* 105.1. DOI: 10.1016/j.neuron.2019.10.013.

Stiefel, Klaus M. and Jay S. Coggan (Oct. 2023). 'The Energy Challenges of Artificial Superintelligence'. In: *Frontiers in Artificial Intelligence* 6. DOI: 10.3389/frai.2023.1240653.

Stradmann, Yannik, Julian Göltz et al. (2025). 'Lu.i – A low-cost electronic neuron for education and outreach'. In: *Trends in Neuroscience and Education* 38. DOI: 10.1016/j.tine.2025.100248.

Stradmann, Yannik, Joscha Ilmberger et al. (2025). *Sustainable operation of research infrastructure for novel computing*. URL: https://arxiv.org/abs/2506.23901. arXiv: 2506.23901 [cs.AR].

Stradmann, Yannik and Johannes Schemmel (Mar. 2024). 'Closing the loop: High-speed robotics with accelerated neuromorphic hardware'. In: *Frontiers in Neuroscience* 18. DOI: 10.3389/fnins.2024.1360122.

Stromatias, Evangelos, Daniel Neil et al. (July 2015). 'Scalable energy-efficient, low-latency implementations of trained spiking Deep Belief Networks on SpiNNaker'. In: *2015 International Joint Conference on Neural Networks (IJCNN)*. IEEE. DOI: 10.1109/ijcnn.2015.7280625.

Sun, Pengfei, Yansong Chua et al. (Nov. 2023). 'Learnable axonal delay in spiking neural networks improves spoken word recognition'. In: *Frontiers in Neuroscience* 17. DOI: 10.3389/fnins.2023.1275944.

Sutter, Herb et al. (Mar. 2005). 'The free lunch is over: A fundamental turn toward concurrency in software'. In: *Dr. Dobb's journal* 30.3.

Taylor, Luke, Andrew King and Nicol Harper (2022). *Robust and accelerated single-spike spiking neural network training with applicability to challenging temporal tasks.* URL: `https://arxiv.org/abs/2205.15286`. arXiv: `2205.15286 [cs.NE]`.

Teeter, Corinne, Ramakrishnan Iyer et al. (Feb. 2018). 'Generalized leaky integrate-and-fire models classify multiple neuron types'. In: *Nature Communications* 9.1. DOI: `10.1038/s41467-017-02717-4`.

Tetzlaff, Tom, Theo Geisel and Markus Diesmann (June 2002). 'The ground state of cortical feed-forward networks'. In: *Neurocomputing* 44–46. DOI: `10.1016/s0925-2312(02)00456-3`.

Thompson, Neil C., Kristjan Greenewald et al. (2022). *The Computational Limits of Deep Learning*. URL: `https://arxiv.org/abs/2007.05558`. arXiv: `2007.05558 [cs.LG]`.

Thomson, Alex M. and Jim Deuchars (Jan. 1994). 'Temporal and spatial properties of local circuits in neocortex'. In: *Trends in Neurosciences* 17.3. DOI: `10.1016/0166-2236(94)90121-x`.

Thorpe, Simon, Arnaud Delorme and Rufin Van Rullen (July 2001). 'Spike-based strategies for rapid processing'. In: *Neural Networks* 14.6–7. DOI: `10.1016/s0893-6080(01)00083-1`.

Thorpe, Simon, Denis Fize and Catherine Marlot (1996). 'Speed of processing in the human visual system'. In: *Nature* 381.6582.

Trachtenberg, Joshua T., Brian E. Chen et al. (Dec. 2002). 'Long-term in vivo imaging of experience-dependent synaptic plasticity in adult cortex'. In: *Nature* 420.6917. DOI: `10.1038/nature01273`.

Trengove, Chris, Cees van Leeuwen and Markus Diesmann (Aug. 2012). 'High-capacity embedding of synfire chains in a cortical network model'. In: *Journal of Computational Neuroscience* 34.2. DOI: `10.1007/s10827-012-0413-9`.

Tsodyks, Misha (Aug. 2008). 'Computational Neuroscience Grand Challenges - a Humble Attempt at Future Forecast'. In: *Frontiers in Neuroscience* 2. DOI: `10.3389/neuro.01.021.2008`.

Tsodyks, Misha and Henry Markram (Jan. 1997). 'The neural code between neocortical pyramidal neurons depends on neurotransmitter release probability'. In: *Proceedings of the National Academy of Sciences* 94.2. DOI: `10.1073/pnas.94.2.719`.

Valadez-Godínez, Sergio, Humberto Sossa and Raúl Santiago-Montero (Feb. 2020). 'On the accuracy and computational cost of spiking neuron implementation'. In: *Neural Networks* 122. DOI: `10.1016/j.neunet.2019.09.026`.

Van Rossum, Mark C. W., Gina G. Turrigiano and Sacha B. Nelson (Mar. 2002). 'Fast Propagation of Firing Rates through Layered Networks of Noisy Neurons'. In: *The Journal of Neuroscience* 22.5. DOI: `10.1523/jneurosci.22-05-01956.2002`.

Varadarajan, Supraja G., John L. Hunyara et al. (Jan. 2022). 'Central nervous system regeneration'. In: *Cell* 185.1. DOI: `10.1016/j.cell.2021.10.029`.

Vaswani, Ashish, Noam Shazeer et al. (Dec. 2017). 'Attention Is All You Need'. In: *arXiv:1706.03762 [cs]*. arXiv: `1706.03762 [cs]`.

Vickers, Neil J., Thomas A. Christensen et al. (Mar. 2001). 'Odour-plume dynamics influence the brain's olfactory code'. In: *Nature* 410.6827. DOI: `10.1038/35068559`.

Vogels, Tim P. and Larry F. Abbott (Nov. 2005). 'Signal Propagation and Logic Gating in Networks of Integrate-and-Fire Neurons'. In: *The Journal of Neuroscience* 25.46. DOI: `10.1523/jneurosci.3508-05.2005`.

Von Neumann, John (1945). 'First draft of a report on the EDVAC'. In: *IEEE Annals of the History of Computing* 15.4. DOI: `10.1109/85.238389`.

Warden, Pete (Apr. 2018). *Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition.* arXiv: 1804.03209 [cs]. DOI: 10.48550/arXiv.1804.03209.

Waxman, Stephen G. (Mar. 1980). 'Determinants of conduction velocity in myelinated nerve fibers'. In: *Muscle &amp; Nerve* 3.2. DOI: 10.1002/mus.880030207.

Werbos, Paul John (1982). 'Applications of advances in nonlinear sensitivity analysis'. In: *System Modeling and Optimization: Proceedings of the 10th IFIP Conference New York City, USA, August 31–September 4, 1981.* Reprinted in Paul J. Werbos. 'Applications of advances in nonlinear sensitivity analysis'. In: *System Modeling and Optimization.* Springer-Verlag, Sept. 2005, pp. 762–770. DOI: 10.1007/bfb0006203.

Werbos, Paul John (Oct. 1990). 'Backpropagation through Time: What It Does and How to Do It'. In: *Proceedings of the IEEE* 78.10. DOI: 10.1109/5.58337.

Williams, Robert W. and Karl Herrup (Mar. 1988). 'The Control of Neuron Number'. In: *Annual Review of Neuroscience* 11.1. DOI: 10.1146/annurev.ne.11.030188.002231.

Winding, Michael, Benjamin D. Pedigo et al. (Mar. 2023). 'The connectome of an insect brain'. In: *Science* 379.6636. DOI: 10.1126/science.add9330.

Winterhagen, Johannes (Nov. 2024). *Neugier auf Zukunft: Das Auto im Jahr 2040.* Accessed: 2025-11-16. URL: https://www.faz.net/aktuell/technik-motor/motor/das-auto-der-zukunft-besuch-im-forschungslabor-von-mercedes-benz-110123997.html.

Wu, Jibin, Yansong Chua et al. (July 2019). 'Deep Spiking Neural Network with Spike Count based Learning Rule'. In: *2019 International Joint Conference on Neural Networks (IJCNN).* IEEE. DOI: 10.1109/ijcnn.2019.8852380.

Wunderlich, Timo C., Akos F. Kungl et al. (2019). 'Demonstrating Advantages of Neuromorphic Computation: A Pilot Study'. In: *Frontiers in Neuroscience* 13. DOI: 10.3389/fnins.2019.00260.

Wunderlich, Timo C. and Christian Pehle (June 2021). 'Event-Based Backpropagation Can Compute Exact Gradients for Spiking Neural Networks'. In: *Scientific Reports* 11.1. DOI: 10.1038/s41598-021-91786-z.

Xin, Wendy and Jonah R. Chan (Oct. 2020). 'Myelin plasticity: sculpting circuits in learning and memory'. In: *Nature Reviews Neuroscience* 21.12. DOI: 10.1038/s41583-020-00379-8.

Yang, Wenyu, Dakun Yang and Yetian Fan (2014). 'A Proof of a Key Formula in the Error-Backpropagation Learning Algorithm for Multiple Spiking Neural Networks'. In: *Advances in Neural Networks – ISNN 2014.* Springer International Publishing. DOI: 10.1007/978-3-319-12436-0_3.

Yankner, Bruce A., Tao Lu and Patrick Loerch (Feb. 2008). 'The Aging Brain'. In: *Annual Review of Pathology: Mechanisms of Disease* 3.1. DOI: 10.1146/annurev.pathmechdis.2.010506.092044.

Yik, Jason, Korneel Van den Berghe et al. (Feb. 2025). 'The Neurobench Framework for Benchmarking Neuromorphic Computing Algorithms and Systems'. In: *Nature Communications* 16.1. DOI: 10.1038/s41467-025-56739-4.

Yousuf, Osama, Brian Hoskins et al. (2024). *Layer Ensemble Averaging for Improving Memristor-Based Artificial Neural Network Performance.* URL: https://arxiv.org/abs/2404.15621. arXiv: 2404.15621 [cs.ET].

Zalc, Bernard (June 2016). 'The acquisition of myelin: An evolutionary perspective'. In: *Brain Research* 1641. DOI: 10.1016/j.brainres.2015.09.005.

Zalc, Bernard, Daniel Goujet and David Colman (June 2008). 'The origin of the myelination program in vertebrates'. In: *Current Biology* 18.12. DOI: 10.1016/j.cub.2008.04.010.

Zeng, Hongkui (July 2022). ‘What is a cell type and how to define it?’ In: *Cell* 185.15. DOI: 10.1016/j.cell.2022.06.031.

Zenke, Friedemann and Surya Ganguli (June 2018). ‘SuperSpike: Supervised Learning in Multilayer Spiking Neural Networks’. In: *Neural Computation* 30.6. DOI: 10.1162/neco_a_01086.

Zhou, Ming, João H. Morais-Cabral et al. (June 2001). ‘Potassium channel receptor site for the inactivation gate and quaternary amine inhibitors’. In: *Nature* 411.6838. DOI: 10.1038/35079500.

Acronyms

ADC analogue-to-digital converter.

AdEx adaptive exponential leaky integrate-and-fire.

AI Artificial Intelligence.

aLIF adaptive leaky integrate-and-fire.

ANN artificial neural network.

ASIC application-specific integrated circuit.

ATP adenosine triphosphate.

BPTT backpropagation through time.

CC BY-NC-ND Creative Commons licence, described as ‘Attribution-NonCommercial-NoDerivatives’, see `https://creativecommons.org/licenses/by-nc-nd/4.0/`.

CMOS Complementary Metal-Oxide-Semiconductor.

CNN convolutional neural network.

CNS central nervous system.

CoBa conductance-based.

CuBa current-based.

DL deep learning.

DNN deep neural network.

GABA gamma-aminobutyric acid.

GD gradient descent.

GeNN GPU enhanced Neuronal Network simulation environment..

GPU graphical processing unit.

I/O input/output.

ITL in-the-loop.

LIF leaky integrate-and-fire.

LLM large language model.

LSM liquid state machine.

Lu.i Educational neuron PCB, implementing LIF dynamics. It is named after Louis Lapicque who proposed the model.

ML machine learning.

MLP multilayer perceptron.

MNIST Dataset comprising handwritten digits commonly used as a machine learning (ML) vision task. It is a modified version of an older dataset by NIST, thus the acronym 'Modified National Institute of Standards and Technology' dataset.

MSE mean squared error.

Myrf Myelin regulatory factor.

nLIF non-leaky integrate-and-fire.

ODE ordinary differential equation.

OL oligodendrocyte.

PCB printed circuit board.

PDMS polydimethylsiloxane.

PNS peripheral nervous system.

PSP postsynaptic potential.

qLIF quadratic integrate-and-fire.

ReLU rectifying linear unit.

RL reinforcement learning.

RLHF reinforcement learning from human feedback.

RNN recurrent neural network.

SGD stochastic gradient descent.

SNN spiking neural network.

STDP spike-timing-dependent plasticity.

SVM support vector machine.

TPU tensor processing unit.

TTFS time-to-first-spike.

YY Yin-Yang.

Acknowledgements

> You must enter the unknown not by accident, but fully aware you are taking a decisive step. Only then, you will know why you are where you are. Don't fret, you won't be alone.
>
> Aleksandra Kašubienė 'Kasuba'
> on her artwork *Spectral Passage*

Without much ado, I want to thank the lovely people around me that have enriched this wonderful time, specifically:

Manfred Salmhofer, Walter Senn, Johannes Schemmel, Wolfram Pernice, and Mihai Petrovici for reviewing and examining my thesis. In addition, I would like to extend my gratitude to Manfred Salmhofer for supervising me during this doctorate and the many insightful meetings. Furthermore, my thanks go to Johannes Schemmel for granting me access, in various forms, to the ElectronicVision(s) infrastructure.

Manfred Stärk for his longstanding support and funding: the stability and flexibility we, and I specifically, gained cannot be overstated. Here's to many more Manfred Stärk Symposia!

All the proofreaders: Katja, Laura, Jakob, Sebastian, Yannik, Jonas, Jimmy, Joscha, Paul, Elias, Timo, Andi, and especially for the last-minute hand-holding of Laura, Yannik and Sebastian!

Everyone in the ElectronicVision(s) group for creating and maintaining the research infrastructure as well as for (unavoidably…) tolerating me blocking BrainScaleS-2 setups when a deadline is coming up again.

All the delightful friends in the NeuroTMA and CompNeuro groups in Bern for your hospitality both for staying at your various places (Mihai, Paul, Ben, Jakob, Anja, Katha, Ismael, Henrik, Kevin, Paul, Timo, Simon, the institute…) and the general atmosphere in Bern that always makes me feel welcome and at home.

Federico Benitez and Virginie Sabado for always having an open ear to help me solve all the bureaucratic issues that come with double affiliations, and also the beautiful time at conferences: Marseille was a lovely experience with you, V, as was Lisbon, Fede!

Walter Senn for all your neuroscientific and academic knowledge, for precisely targeted remarks on manuscripts, for teaching me about life and the importance of time away from the desk, but also for always lightening up meetings.

Melika Payvand for always being positive and infectiously happy. I massively enjoy our collaboration, and look forward to seeing you more often again when I am physically closer to Zurich.

Jimmy Weber for exploring the power of delays together, for always having a bed for me, all the fun we had together, and introducing me to Sierre-Zinal.

Jonas Oberste-Frielinghaus for teaching me rhythm (i.e., learning insights into synchronisation), and for a last minute deep synchronisation figure!

Mohammed Sinan and Peter Lake for giving me a chance to dive into teaching and supervision, for winning you as friends and, specifically for Peter, for infecting him with both bouldering and climbing viruses.

Oliver Breitwieser for setting me up with a topic that has proved so very fruitful, and just as much for instilling in me the dedication to tooling. Where would I be without all the tooling.

Joscha Ilmberger for always being helpful when FPGAs yet again produce errors, and all the help on questions about electronics or 3D printing.

Andreas Baumbach for his often unconventional opinions and advice, which can be all the more helpful and insightful. For all the (late night) life advice, and for the shared journey of the BIC lecture that I'll never forget.

Katharina Wilmes for advice early on in my PhD and for making the Covid winter a lot more enjoyable.

Kevin Max for providing a safe space to share the fascination for Swiss rap – and comparable German art.

Paul Haider for nearly always answering affirmative when asked about going bouldering. I am looking forward to all the future sessions.

Ben von Hünerbein: for outright welcoming me in his home in Bern, as well as for perspective and insights, and together with Ismael for teaching me about openness.

Ismael Jaras for leading me to discover a new world, for supporting me with my broken foot, for being a good friend, and always teaching me about life. I am so sorry I could not meet you last time in Bern.

Yannik Stradman for always being available as a style guide, and for setting a paragon of unattainable perfection, and for the many skills you have taught me – but I never want to work with your latex setup again <3. I am certainly grateful with how the Lu.i project turned out, but not least because it provided the fertile ground to become your friend!

Laura Kriener for being there to withstand the pain and stress together (I do not want to know how the PhD would have turned out if it were not for you.), but also for your valuable insights, persistence, and commitment to excellence. There are countless memories of time spent together, both in person and virtual: thanks for keeping me sane during lockdowns.

Sebastian Billaudelle for your often contrarian but always constructive attitude, for your openness to discuss topics of any kind with verve, and for showing your outright enthusiasm for a million things. Naturally for your insights on surrogate gradients and hardware design, but most importantly for your friendship.

Mihai Petrovici for unprecedented freedom to do as I please, for the unspoken support that is heartening to have in the back of one's head. As a literally shining example, only this combined freedom and support made Lu.i possible. While often implicit, simply because of the distance, you were there when it mattered. Of the uncountable occasions, I will certainly never forget the Christmas Eve we shared virtually, finishing the paper. Thanks for the support, the guidance, and, not least, the life lessons I have learned from you – I will not divulge whether by sermon, example, or counterexample. And just once: aren't analogue neurones marvellous?
What deserves a special mention is the atmosphere you managed to create in Bern: scientific excellence in unison with kindness, inciting aspiration for greatness among a close group of friends. I truly hope we can spend much more time discussing science, consciousness, and everything beyond.

Taaaron für die vielen gemeinsamen Abenteuer trotz der kurzen Zeit, denen noch viele weitere folgen mögen: am Fels, auf Skiern und in Schriesheim, allen Boulderhallen und irgendwann auch in Bleau. Mit euch hätte ich selbst beim Tradclimben keine Zweifel, dass es Seefisch. Ich kann nicht ausdrücken, was es mir bedeutet hat, dass ihr mich in der BG besucht habt. Und danke für das Ablenken in der heißen Schreibphase.

Julian für die langwährende Freundschaft, von Rugby über Bouldern zu Fahrradtrips. Dafür, dass du mir zeigst was wichtig ist und dass du immer da bist, wenn es zählt.

Max for getting me interested in science early on, for showing me your life at university but also the importance of life outside of school. I fear that the conference in Erlangen was the closest we will get to attend one together, but we will make up for that in other time spend together.

My love Katja, for providing stability during all ups and downs, in the past and far into the future. For the constant medical advice. For being excited about the same things, for providing open spaces for development, and for teaching me your passions. Oh, and for feeding me during writing.

Meinen Eltern, Conny und Geri, für all eure Liebe und Unterstützung, dafür, dass ihr immer da seid, und ich mich stets auf euren Rückhalt verlassen kann. Danke für alles!

And now, let's step happily and decisively into the unknown.

Appendix

SI.A. Dissemination of Lu.i

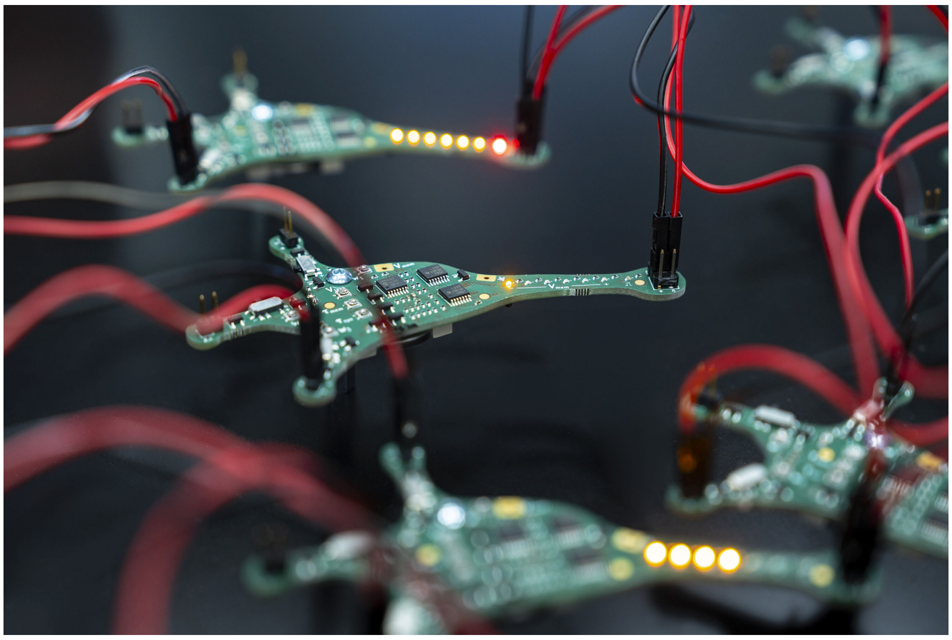

Figure SI.A.1.: Lu.i neuron as produced by Mercedes-Benz. Photo by *Mercedes-Benz Group AG* taken with permission from the press release (Mercedes-Benz Group AG 2024). This and similar photos were featured as part of their own public outreach[32] as well as in a number of press articles, e.g., by Top Gear (Abidin 2024) and the Frankfurter Allgemeine Zeitung (Winterhagen 2024).

[32]`https://group.mercedes-benz.com/innovations/product-innovation/autonomous-driving/neuromorphic-computing.html`

SI.B. Supplementary analysis of the activity in a deep network

In addition to the six and seven layer networks included in Chapter 6, we trained a deep network with ten layers (one input, eight hidden, and one output layer). The network adheres to Dale's principle, i.e., each hidden layer is separated into an excitatory and inhibitory population. In comparison to the more shallow networks, the layers in the network have a reduced size with 75 excitatory and 25 inhibitory neurons instead of 300 and 100, respectively. Even without hyperparameter tuning, the training algorithm performs well (Fig. SI.B.1) with a resulting test accuracy of 97.0 % (99.3 % train). Figure SI.B.2c highlights the constraint in the connectivity with a clear split in the weights between the excitatory and inhibitory neurons.

In Figure SI.B.2 we can see synchronisation emerging in this deep network: the activity spreads out in the early layers, and condenses towards the label layer. The effect is visible both directly in the raster plot (Fig. SI.B.2a), and in the histogram of the spiking activity as a sharpening of the peaks (Fig. SI.B.2b).

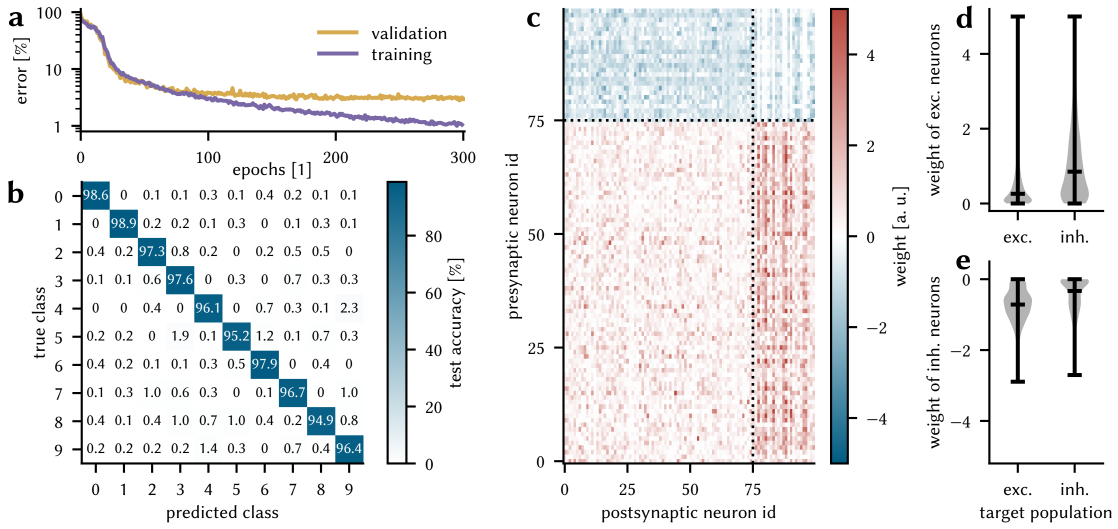

Figure SI.B.1.: Training a ten-layered network. (**a**) Training and validation error during the training indicates that the algorithm can make good use of the computational capabilities of the network, because the training error becomes very small. The gap between validation and training error implies that there is a lack of generalisation which is a consequence of the large network and could be combatted by more advanced data augmentation. (**b**) The confusion matrix has a prominent diagonal of correctly classified classes, reflecting the high test accuracy. (**c**) Example trained weight matrix between the first and second hidden layer. It is apparent that the first 75 neurons (rows below the horizontal dotted line) only have excitatory connections with positive weights (red), while the remainder is inhibitory, with negative weights (blue). The next layer has an identical setup, first excitatory then inhibitory neurons (separated by the vertical dotted line). There is a visible difference in the hue between these two populations. (**d**) Distribution of the weights from (c) of excitatory neurons, separated by whether they target excitatory (left) or inhibitory (right) neurons. The visible difference from before is confirmed: the connections to inhibitory neurons are stronger, highlighted by the higher median (central line) for the right distribution. (**e**) Weight distribution as in (d) but for inhibitory neurons. The large fraction of vanishing weights for inhibitory-inhibitory connections (white area on the top right in (c)) appears as a bump at 0 in the right distribution.

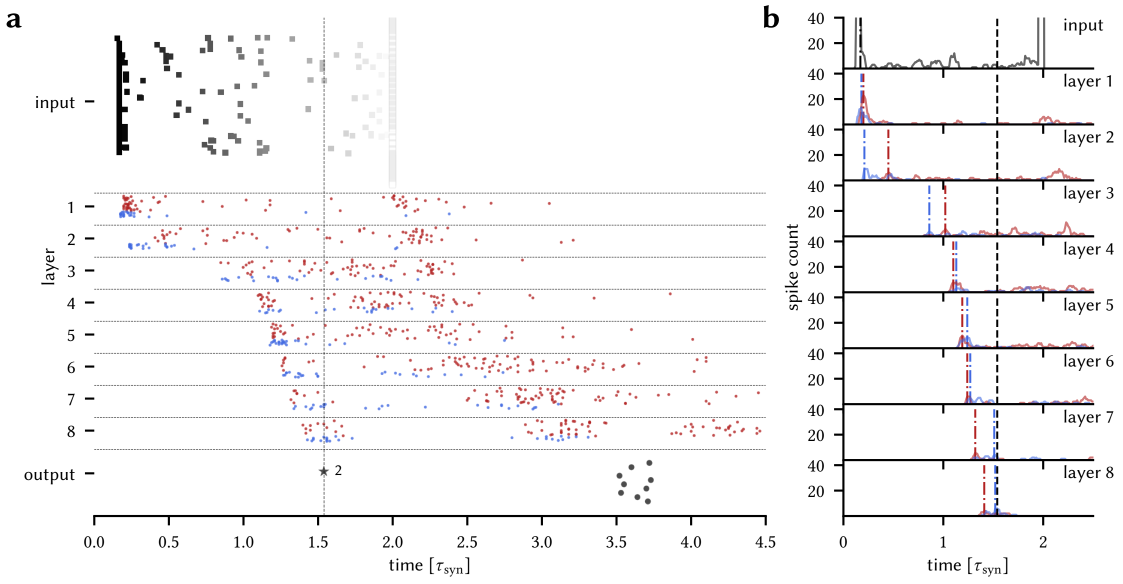

Figure SI.B.2.: Synchronisation in a deep network. (**a**) Raster plot of the activity in the network in response to the presentation of a sample image. The input spikes encode the pixel brightness in the timing of their spike (brighter is later). For the hidden neurons, spikes are colour-coded depending on whether they belong to an excitatory (red) or inhibitory (blue) neuron. The output neuron associated with the correct label of the sample (indicated by a star) spikes before the other output neuron, i.e., the network classified correctly. The activity in the input is initially very peaked, but causes a dispersed activity in layers 2 and 3. Afterwards, the pulse condenses in the later layers. (**b**) This impression is confirmed when looking at the histogram of the activity. The maximum of activity in the histogram (counting only spikes before the classifying spike in the output, black dashed line) is marked as a red or blue dash-dotted line for the excitatory or inhibitory population, respectively.

SI.C. Locality of the derivative

In the discussion we mentioned that the derivatives required in the learning rule (Chapter 5, Eqs. 2 and 3) can be brought into a form that depends only on local quantities. Apart from the difference between the spike of the neuron t_sp and the presynaptic spike times t_i, the equations are nonlocal in the first two factors:

$$\frac{\partial t_\mathrm{sp}}{\partial w_i} = -\frac{1}{a_1}\frac{1}{\mathcal{W}(z)+1}\exp\left(\frac{t_i}{\tau_\mathrm{syn}}\right)\left(t_\mathrm{sp}-t_i\right) , \tag{SI.C.1}$$

$$\frac{\partial t_\mathrm{sp}}{\partial t_i} = -\frac{1}{a_1}\frac{1}{\mathcal{W}(z)+1}\exp\left(\frac{t_i}{\tau_\mathrm{syn}}\right)\frac{w_i}{\tau_\mathrm{syn}}\left(t_\mathrm{sp}-t_i-\tau_\mathrm{syn}\right) . \tag{SI.C.2}$$

Looking specifically at this denominator, and using the definition for z (Chapter 5, Eqs. 15 and 18) as well as a_i and b (Chapter 5, Eqs. 11 and 12), we calculate:

$$a_1\big(\mathcal{W}(z)+1\big) = b - a_1\frac{t_\mathrm{sp}}{\tau_\mathrm{syn}} + a_1 \tag{SI.C.3}$$

$$= \sum_{j\in C}\exp\left(\frac{t_j}{\tau_\mathrm{syn}}\right) w_j\frac{t_j - t_\mathrm{sp}+\tau_\mathrm{syn}}{\tau_\mathrm{syn}} \tag{SI.C.4}$$

$$= \exp\left(\frac{t_\mathrm{sp}}{\tau_\mathrm{syn}}\right)\sum_{j\in C}\exp\left(-\frac{t_\mathrm{sp}-t_j}{\tau_\mathrm{syn}}\right) w_j\frac{t_j - t_\mathrm{sp}+\tau_\mathrm{syn}}{\tau_\mathrm{syn}} \tag{SI.C.5}$$

$$= \exp\left(\frac{t_\mathrm{sp}}{\tau_\mathrm{syn}}\right) C_\mathrm{mem}\dot{u}_\mathrm{mem}(t_\mathrm{sp}) . \tag{SI.C.6}$$

In Equation (SI.C.5), suggested by the last fraction that we always encounter in the context of time derivatives of the voltage, we can recognise $\dot{u}_\mathrm{mem}$. Inserting this into Equations (SI.C.1) and (SI.C.2) yields the final form:

$$\frac{\partial t_\mathrm{sp}}{\partial w_i} = -\frac{1}{C_\mathrm{mem}}\frac{1}{\dot{u}_\mathrm{mem}(t_\mathrm{sp})}\exp\left(-\frac{t_\mathrm{sp}-t_j}{\tau_\mathrm{syn}}\right)\left(t_\mathrm{sp}-t_i\right) , \tag{SI.C.7}$$

$$\frac{\partial t_\mathrm{sp}}{\partial t_i} = -\frac{1}{C_\mathrm{mem}}\frac{1}{\dot{u}_\mathrm{mem}(t_\mathrm{sp})}\exp\left(-\frac{t_\mathrm{sp}-t_j}{\tau_\mathrm{syn}}\right) w_i\frac{t_\mathrm{sp}-t_i-\tau_\mathrm{syn}}{\tau_\mathrm{syn}} , \tag{SI.C.8}$$

containing only quantities that are arguably locally available at the synapse. As argued in Section 8.6, when providing the error locally to the synapse this leads to a fully local learning rule.

Additionally, under careful inspection of these equations we note that another derivative appears, $\partial u_\mathrm{mem}/\partial w_i$ for the former and $\partial u_\mathrm{mem}/\partial t_i$ for the latter. In fact, this can be independently derived by using the implicit function theorem as shown by Yang et al. (2014), producing the derivative of the spike time t_sp with respect to a parameter θ:

$$\frac{\partial t_\mathrm{sp}}{\partial \theta} = -\frac{1}{\frac{\partial u_\mathrm{mem}}{\partial t}}\frac{\partial u_\mathrm{mem}}{\partial \theta} . \tag{SI.C.9}$$

Interestingly, this does not depend on the ratio of time constants, so together with Chapter 7, Section SI.D it provides an explicit training method for general leaky integrate-and-fire (LIF) neurons.

SI.D. Including delays into neuronal dynamics

In Chapter 7, we separated the delay mechanism from the computation in the neuron (see Chapter 7, Fig. 2) which enabled the efficient software execution. Another common formulation includes delays within the parameters of the neuronal computation, which we briefly develop in the following.

The effect of a transmission delay is the time difference between the moment of spike emission and the moment the spike influences another neuron. This is captured formally in Equation (2.10). Generalising to a neuron with multiple presynaptic partners, let us assume a sequence of input spikes $\{t_i\}$ with associated parameters, weights $\{w_i\}$ and delays $\{d_i\}$. For exponential current-based (CuBa) synapses, the voltage dynamics is then described by

$$u_{\mathrm{mem}}(t) = \sum_i \Theta(t-(t_i+d_i)) \frac{w_i}{g_\ell} \frac{\tau_{\mathrm{syn}}}{\tau_{\mathrm{mem}}-\tau_{\mathrm{syn}}} \cdot \left[\exp\left(-\frac{t-(t_i+d_i)}{\tau_{\mathrm{mem}}}\right) - \exp\left(-\frac{t-(t_i+d_i)}{\tau_{\mathrm{syn}}}\right)\right] . \tag{SI.D.1}$$

We amend the definition of the shorthands a and b in Chapter 5, Equations 11 and 12, to be

$$\tilde{a}_n := \sum_{i\in C} w_i \exp\left(\frac{t_i+d_i}{n\tau_{\mathrm{syn}}}\right) \quad \text{and} \quad \tilde{b} := \sum_{i\in C} w_i \frac{t_i+d_i}{\tau_{\mathrm{syn}}} \exp\left(\frac{t_i+d_i}{\tau_{\mathrm{syn}}}\right) . \tag{SI.D.2}$$

As before, the definition depends on the causal set $C = \{i \mid t_i + d_i < t_{\mathrm{sp}}\}$ of input spikes with a causal effect on the output spike t_{sp}. Corresponding to Chapter 5, Equations 2 and 3, for $\tau_{\mathrm{mem}} = \tau_{\mathrm{syn}}$ we get a spike time

$$t_{\mathrm{sp}} = \tau_{\mathrm{syn}} \left\{ \frac{\tilde{b}}{\tilde{a}_1} - \mathcal{W}\left[-\frac{g_\ell \vartheta}{\tilde{a}_1} \exp\left(\frac{\tilde{b}}{\tilde{a}_1}\right)\right]\right\} , \tag{SI.D.3}$$

and for $\tau_{\mathrm{mem}} = 2\tau_{\mathrm{syn}}$

$$t_{\mathrm{sp}} = 2\tau_{\mathrm{syn}} \ln\left[\frac{2\tilde{a}_1}{\tilde{a}_2 + \sqrt{\tilde{a}_2^2 - 4\tilde{a}_1 g_\ell \vartheta}}\right] . \tag{SI.D.4}$$

Because both $\tilde{a}_n$ and $\tilde{b}$, and therefore Equations (SI.D.3) and (SI.D.4), are symmetric under the exchange of the input spikes t_i and delays d_i, we can conclude that the derivatives with respect to t_i and d_i are identical

$$\frac{\partial t_{\mathrm{sp}}}{\partial t_i} = \frac{\partial t_{\mathrm{sp}}}{\partial d_i} . \tag{SI.D.5}$$

In other words, we recover the observation that shifting an input spike time versus changing a delay has the identical effect on the output spike.

Cover
Artistic representation of the response of two label neurons (front and back cover) after training a spiking network to classify the Yin-Yang data set on the neuromorphic hardware BrainScaleS-2. Each circle represents one pattern (see Chapter 4, Fig. 1), and its colour shows the spike time of the depicted label neuron, red meaning the neuron fired first and decided the classification, similar to Chapter 5, Fig. 4.