Brain-inspired methods for boosting temporal computing and learning capabilities of spiking neural networks

by

Darjan SALAJ

DISSERTATION

submitted for the degree of

Doctor Technicae

Institute for Theoretical Computer Science

Graz University of Technology

Thesis Advisor

Prof. Dr. Wolfgang MAASS

Graz, Feb 2021

This document is set in Palatino, compiled with pdfLaTeX2e and Biber.

The LaTeX template from Karl Voit is based on KOMA script and can be found online: https://github.com/novoid/LaTeX-KOMA-template

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used material other than from the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Graz, _____ _____

Date Signature

Eidesstattliche Erklärung[1]

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am _____ _____

Datum Unterschrift

[1] Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008; Genehmigung des Senates am 1.12.2008

Abstract

Despite many decades of active research in different fields of neuroscience, psychology, and artificial intelligence, understanding the relationship between the biophysical mechanisms of the brain and their role in the processes of cognition and task solving is still a standing challenge. Even for many simple behaviours that are exhibited by non-human brains, reliable theories about the way the brain works from the cognitive perspective is missing. A major hurdle to understanding the brain is the fact that the brain has developed under strict energy constraints. Thus it is difficult to decipher which of the biophysical mechanisms support the cognitive processes, which support the energy efficiency aspect, and which are critical for the cognition and the energy efficiency of the brain. One prominent aspect of a brain is the ability to remember which is vital for the emergence of most cognitive abilities. Most temporal computation tasks encountered by brains — like every-day human activities such as speech, reading, building and playing games — rely on some form of memory. Spike-frequency adaptation is a prominent biophysical mechanism present in a substantial fraction of neurons. It describes a property of biological neurons where their firing activity is transiently dampened by the neurons preceding firing activity. Thus this mechanism is well known for its role in the energy efficiency aspect of brain networks. We demonstrate that, in addition to the energy efficiency role, this mechanism can be exploited and is sufficient to develop a robust form of memory in networks of spiking neurons which supports a variety of temporal computations. We find many similarities between this emergent form of memory and the activity-silent form of working memory previously observed in biological neural networks. Further, we develop the negative imprint hypothesis that offers an explanation about how this mechanism is exploited for the function of short-term and working memory.

Zusammenfassung

Trotz jahrzehntelanger Forschung am Gehirn aus der Perspektive der Neurowissenschaften, Psychologie und künstlichen Intelligenz ist das Verständnis der Beziehung zwischen den biophysikalischen Mechanismen des Gehirns und ihrer funktionalen Rolle in der Kognition und Aufgabenlösung immer noch eine ständige Herausforderung. Auch für die einfachsten, selbst bei Tieren beobachtbaren Verhaltensweisen, fehlen verlässliche Theorien über die Rolle und Funktionsweise des Gehirns. Eine große Hürde für das Verständnis des Gehirns ist die Tatsache, dass sich das Gehirn evolutinär auf hohe Energieeffizienz hin entwickeln musste. Daher ist auch für einen einzelnen biophysikalischen Prozess schwer zu entschlüsseln, ob er für die kognitive Funktion, die Energieeffizenz oder gar für beides kritisch ist. Ein wichtiger Aspekt des Gehirns ist die Fähigkeit, sich zu erinnern — unabdingbar für die Entstehung der meisten kognitiven Fähigkeiten. Die meisten Aufgaben, mit denen das Gehirn konfrontiert ist — beispielsweise alltägliche Aktivitäten wie Sprechen, Lesen, Bauen und Spielen — brauchen in irgendeiner Form ein Arbeitsgedächtnis. Diese Arbeit untersucht den Zusammenhang zwischen Spike Frequency Adaptation und der Fähigkeit zur Verarbeitung von Informationen mit zeitlichem Kontext. Spike Frequency Adaptation ist ein wohlbekannter biophysikalischer Mechanismus, der in dem meisten Neuronen vorhanden ist. Durch diesen wird deren Feueraktivität durch eigene vorhergehende Aktivität vorübergehend gedämpft, was bekanntermaßen zu verbesserter Energieeffizienz führt. Wir zeigen in dieser Arbeit, dass dieser Mechanismus zusätzlich auch eine funktionale Rolle hat: Er reicht, eine robuste Art von Gedächtnis in Netzwerken von Spike-Neuronen zu entwickeln. Wir finden viele Ähnlichkeiten zwischen dieser emergenten Form von Gedächtnis und bereits in den Neurowisschenschaften beobachteten aktivitätsfreien Formen von Arbeitsgedächtnis. Weiter entwickeln wir die Hypothese des negativen Abdrucks (negative imprint hypothesis), die

eine Erklärung dafür bietet, wie dieser Mechanismus für die Funktion des Kurzzeit- und Arbeitsgedächtnisses genutzt wird.

Acknowledgements

First, I would like to thank my supervisor Wolfgang Maass for giving me the opportunity to work on very exciting scientific problems, and for his support and guidance during my time as a PhD student. I would especially like to thank him for understanding and support for my unique circumstances and wishes towards the end of my PhD studies.

I would also like to thank Christian Mayr for his time in being the second referee for this thesis, and for many exciting and motivational discussions about our research and neuromorphic applications.

A special thanks is due to Robert Legenstein for his huge help in writing and discussions on all the research I was involved in, to Guillaume Bellec and Anand Subramoney for extensive supervision, collaboration, advising and many fun social interactions during my studies. I would like to extend my gratitude to Michael Müller for many discussions during our walks and for all the bread! A very special thanks to Anand Subramoney for all the help in writing this thesis and our regular discussions, to Florian Unger for the best pizzas, social events and help with translation of the abstract of this thesis, and to Thomas Limbacher for being my first diving buddy. I am very grateful to the administrative staff at our institute: Daniela Windisch-Scharler, Charlotte Rumpf, Oliver Friedl and Nicoletta Kähling for their very needed positive presence and helpful support in all administrative matters. I would like to gratefully thank all of my colleagues: David Kappel, Franz Scherr, Elias Hajek, Arjun Rao, Ceca Kraisnikovic, Thomas Bohnstingl, Horst Petschenig, Philip Plank, and Martin Pernull for many fun conversations, journal clubs and social events.

Most importantly I would like to thank my wife Anida, parents, sister and grandmother, for their love and support in everything I do.

Contents

List of Figures

List of Tables

1. Introduction

Contents

The human brain is the only known instance of general intelligence, and thus it continues to serve as an inspiration for building and improving artificial intelligence systems. Humans evolved to survive and adapt to many challenging environments, and human brain is the substrate that allowed the emergence of highly adaptive and complex behaviours that ensured survival. To increase the chances of survival, brains not only had to provide flexible problem-solving abilities but also do it under extreme energy constraints which made the brains very specialized for efficient problem-solving. This poses an interesting challenge for brain research: Which mechanisms of the biological brain are serving a functional role, and which are present only to support the energy efficiency of the system?

Despite the rich history and recent acceleration in all the research domains relating to brain research — including many branches of neuroscience, machine learning and artificial intelligence — we are still in the nascent stage of comprehending intelligence and how it arises from the underlying biological mechanisms of the brain. This can be attributed to the opaque relation between the diverse parts of the brain — including many different cell types, brain regions and underlying dynamical processes — and their functional role. This problem is amplified by the sheer scale of the brain, where not only the number of neurons and synapses make even the simplest of simulations of the brain prohibitive, but also the extent of dynamical processes which evolve temporally on many orders of magnitude from

milliseconds to years. Another difficulty is due to the perspective we take on the brain functions, which are influenced by the computational theories and philosophical ideas from the past (György Buzsáki, 2019). Overcoming these difficulties leads us one step closer to understanding the brains, which enables better brain disease treatments and building better and more efficient artificial intelligence systems.

Modeling some aspects of biological brains in artificial models is a viable approach to elucidate inner workings of the brain (Levenstein et al., 2020). There exists a wide variety of models that capture different details of the networks of neurons in the brain. One of the simplest models that have been studied in the context of brain research is the recurrent neural network (RNN). Another class of models that go more in the direction of biological realism are the recurrent spiking neural networks (RSNNs). RSNNs have been used as models for investigating learning and memory in biological brains, and have been referred to as the third generation of neural network models (Maass, 1997).

This thesis focuses on understanding a specific biophysical mechanisms — spike-frequency adaptation — in the context of its role in memory and problem-solving in the temporal domain.

1.1. Working memory

One of the pre-requisites for the emergence of complex cognitive abilities is some form of memory. For example, most human activities, such as speech, text comprehension, sports and playing games, rely on the brain's ability to memorize information. In such tasks, a person needs to perform many actions while temporarily keeping the intermediate results in mind (Buchweitz, 2008).

The term "working memory" was originally published in cognitive psychology publication (G. Miller, 1960) where it described the theoretical concept that refers to the mechanism active during the performance of a cognitive task and which is responsible for the maintenance of task-relevant information. It has been stated that the working memory is "perhaps the most significant achievement of the human mental evolution" (S. Goldman-Rakic,

1992). In (Diamond, 2013; Cowan, 2008) the term is defined as "the ability to hold information and manipulate it as opposed to short-term memory which refers to just holding the memory". Its significance lead to it being studied in other fields such as cognitive neuroscience and artificial intelligence. The term "working memory", however, is ambiguous and has been defined differently within different contexts and research communities.

In this thesis, we investigate the role of spike-frequency adaptation and other biophysical mechanisms in both the working memory and short-term memory, as defined in (Diamond, 2013; Cowan, 2008), but focus on the broader computing capabilities in the temporal domain.

1.2. Temporal computing

Brains operate in dynamically changing environments and thus are able to integrate and manipulate temporally dispersed information from continuous input streams on the behavioural timescale of seconds. We label "temporal computing tasks", tasks that involve such temporal computation. Throughout this thesis, a variety of temporal computing tasks will be used to benchmark the abilities of different RSNN models. Different instances of temporal computing tasks can have very different requirements in order to be solved. Broadly the tasks that appear in this thesis can be categorized in one of the following three categories:

Memorization and recall is the simplest form of temporal task. The information does not need to be processed but just stored and recalled reliably. Difficulties in learning this task come in the form of noisy input streams where the network has to learn to extract only the relevant information on demand, and more importantly ignore the noise input before the recall stage. Another difficulty in this type of tasks is the ability of the network to continuously store and recall different values without explicitly resetting the internal state.

Sequence classification or temporal pattern recognition is a very common machine learning setup, most prominently appearing in the speech processing and recognition domain. Here a longer input stream has to be classified to one of the known classes. The major difficulty in this type of task is

the diversity between the samples of the same class which makes strong generalization a requirement for successfully solving such tasks.

Cognitive tasks are the most complex class of the tasks used in this thesis. They are often formalized as operations on sequences of symbols and demand generalization to previously unseen strings of symbols. Realization of such generalization requires multiple levels of working memory. This type of generalization is considered an essential feature of cognitive architectures (Marcus, 2003).

1.3. Structure of the thesis

The results in this thesis are based on publications to which I have contributed as first- or co-first author or unpublished material I authored during my PhD studies. A detailed statement about author contributions is given at the beginning of every chapter.

Chapter 2 introduces the modeling of the biological neurons and presents the models and the training methods that are used in the following chapters.

In Chapter 3, we show how using a pseudo-derivative to make the spiking neuron model differentiable allows us to train the RSNNs using backpropagation through time. In addition, we demonstrate how the inclusion of spike-frequency adaptation leads to a significant increase in temporal computing capabilities of RSNNs.

In Chapter 4, we investigate the functional role of spike-frequency adaptation in more depth and propose the negative imprint hypothesis about how the spike-frequency adaptation is exploited as the substrate that supports the emergence of robust memory capabilities. We demonstrate the ability of such RSNNs to solve cognitive tasks that involve operations on strings of symbols or require multiple levels of working memory. Further, we find that representations such as assembly sequences and mixed selectivity emerge in those RSNNs.

In Chapter 5, we perform a comparative benchmarking on the temporal computing capabilities of RSNNs endowed with synaptic and neural biophysical mechanisms different from spike-frequency adaptation.

Chapter 6 builds upon *e-prop*, the biologically plausible approximation of backpropagation through time published in (Bellec, Scherr, Subramoney, et al., 2020). There we demonstrate how RSNNs endowed with spike-frequency adaptation can also be trained with biologically plausible learning rules to solve challenging temporal tasks. Also we introduce a variant of the learning rule labelled *adaptive e-prop*.

2. Models

Contents

Brains and nervous systems in general consist of many cell types arranged in complex structures. The most common building block of neural networks is the spiking neuron. It produces discrete output by integrating the inputs received from neighbouring cells. There is also an established consensus that the main functional role of neural networks is performed by spiking neurons. Thus, this work focuses on modelling and analysis of recurrent networks of spiking neurons.

2.1. The biological neuron

Biological neurons themselves have many different types, however, the basic architecture and electrical properties are common among all of them. Biological neurons are composed of three functionally distinct parts: dendrites, soma, and axon. See Figure 2.1.

Dendrites are fibers with synapses through which a neuron connects to the neighbouring neurons. Dendrites receive the input signals and relay them to the soma. *Soma* is the central body of a neuron. Soma integrates the incoming signals from many dendrites and generates the action potentials

which are called spikes. The generated spikes are mostly uniform across neurons in terms of duration and shape and are therefore often abstracted as binary events (Barnett and Larkman, 2007). *Axon* relays the spikes generated by soma to the dendrites of other neurons.

A synapse is a connection between the dendrite of a post-synaptic neuron and the axon of a pre-synaptic neuron. Synapses do not propagate the spikes between neurons, instead, the spike is processed and transmitted through chemical processes in the synapse. This involves modulation of calcium ion channels by pre-synaptic spikes, merging of vesicles to the axon membrane by higher calcium concentration, vesicle release of neurotransmitter in the synaptic cleft, binding of neurotransmitters to the receptors of the post-synaptic dendrite, opening of the ion channels through which cations and anions flow, and finally, the modulation of potential in post-synaptic dendrite (Gerstner and W. Kistler, 2002).

Neurons have a dynamic electric potential between their cellular walls which is termed membrane potential. The dynamic of membrane potential involves the transport of different types of ions which are gated through their corresponding ion channels. Without input, the membrane potential of a neuron decays to the cell-specific resting potential.

The dendrites are often connected to many axons and integrate the incoming spike trains. Due to the heterogeneity of synapses, the integrated post-synaptic potentials have high variance in shape and amplitude. The soma integrates the post-synaptic potentials from all dendrites into a single membrane potential. Synapses can contribute positively or negatively to the membrane potential and so are functionally segregated into inhibitory (those that decrease the membrane potential) and excitatory (those that depolarize, increase, the membrane potential).

Every neuron has a so-called firing threshold. When the membrane potential increases to the value of the neuron-specific firing threshold, a spike fires (an action potential is generated). This spike is relayed by the axon to the synapses of other neurons. After a spike fires, the membrane potential undergoes fast repolarization, which lowers it to a value below the resting potential, and enters a refractory period, during which it does not change.

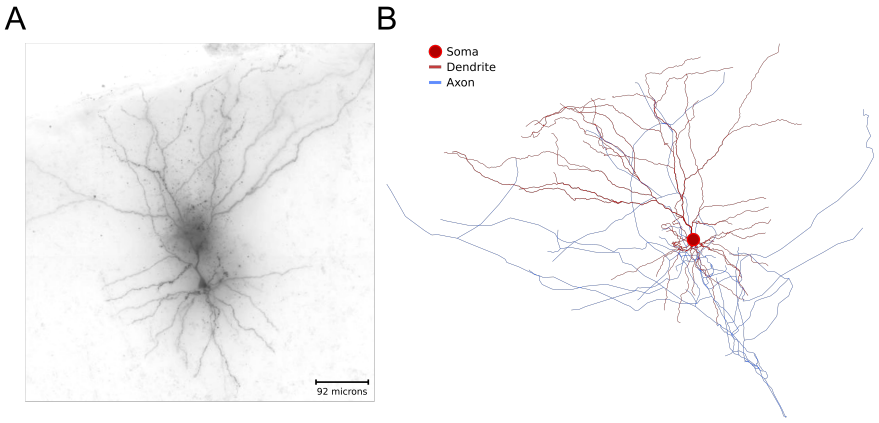

Fig. 2.1.: Morphology of a biological neuron: (**A**) Projected top view of a neuron from human temporal lobe (Donor H16.03.005, ID 526714733, Allen Institute, 2018). (**B**) Projection of a partial 3D reconstruction of the same neuron with marked neuron morphology (Allen Institute, 2018). Dendrites make synaptic connections with neurons from which they receive the input. Soma integrates the inputs from dendrites and generates spikes which are propagated by axon to the synaptic connections of the downstream neurons.

The membrane potential is only one of many neuron-specific chemical states that evolves over time. Another prominent neuron-specific state which can evolve over time is the firing threshold. The process which modulates firing threshold is commonly labelled spike-frequency adaptation and it denotes a feature of spiking neurons where their preceding firing activity transiently increases their firing threshold. A substantial number of neurons in the mouse visual cortex and in the human frontal lobe exhibit spike-frequency adaptation (Allen Institute, 2018).

Communication between neurons through spike trains is modulated by the complex dynamics of the synapses which are conditioned by previous activity and other factors. It is commonly accepted that most learning in the brain is primarily implemented through the adaptation of the strengths of the synaptic connections and their rewiring.

2.2. Modelling biological neurons

Different research communities have developed a variety of neuron models with different levels of detail and biological realism. The machine learning and deep learning community developed artificial neural networks with a focus on solving complex tasks with efficient use of contemporary hardware. Modern artificial neural networks are categorized as the second generation of neural networks (Maass, 1997). Neuroscience and related fields focused more on the biologically realistic models, such as SNNs, which are categorized as the third generation of neural networks (Maass, 1997).

One of the earliest spiking neuron models was developed by Hodgkin, Huxley, and Katz, 1952. It described the voltage-gated ion channels observed during the electrophysiological experiments conducted on the giant squid neurons. This model was developed with the goal of describing biophysical mechanisms of the neurons in detail.

Later, simpler models were developed by describing the essential properties of the earlier, more complex, models. One of the most popular simpler models is the leaky integrate and fire (LIF) model (Gerstner and W. M. Kistler, 2002). This model describes the membrane potential as a linear sum of the all incoming signals which are modelled as currents weighted by the synaptic weights. The membrane potential also includes a leak current which models the decay of membrane potential in neurons to the resting potential.

In this thesis, we exclusively explore the RSNN models and mostly use LIF neuron models which are often extended with the spike-frequency adaptation mechanism.

2.3. Leaky integrate and fire (LIF) neurons

A LIF neuron j spikes as soon at its membrane potential $V_j(t)$ is above its threshold v_{th}. At each spike time t, the membrane potential $V_j(t)$ is reset by subtracting the threshold value v_{th} and the neuron enters a strict refractory period for 1 to 5 ms (depending on the experiment) where it

cannot spike again. Between spikes, the membrane voltage $V_j(t)$ is following the dynamic:

$$\tau_m \dot{V}_j(t) \quad = \quad -V_j(t) + R_m I_j(t),$$

where τ_m is the membrane constant of neuron j, R_m is the resistance of the cell membrane, and I_j the input current.

Our simulations were performed in discrete time with a time step $\delta t = 1\,\text{ms}$. In discrete time, the input and output spike trains are modeled as binary sequences $x_i(t), z_j(t) \in \{0, \frac{1}{\delta t}\}$ respectively. Neuron j emits a spike at time t if it is currently not in a refractory period, and its membrane potential $V_j(t)$ is above its threshold. During the refractory period following a spike, $z_j(t)$ is fixed to zero. The neural dynamics in discrete time reads as follows:

$$V_j(t+\delta t) \quad = \quad \alpha V_j(t) + (1-\alpha) R_m I_j(t) - v_{\text{th}} z_j(t) \delta t, \tag{2.1}$$

where $\alpha = \exp(-\frac{\delta t}{\tau_m})$, with τ_m being the membrane constant of the neuron j. The spike of neuron j is defined by $z_j(t) = H\left(\frac{V_j(t) - v_{\text{th}}}{v_{\text{th}}}\right) \frac{1}{\delta t}$, with $H(x) = 0$ if $x < 0$ and 1 otherwise. The term $-v_{\text{th}} z_j(t) \delta t$ implements the reset of the membrane voltage after each spike.

In all simulations, the R_m was set to $1\,\text{G}\Omega$. The input current $I_j(t)$ is defined as the weighted sum of spikes from external inputs and other neurons in the network:

$$I_j(t) \quad = \quad \sum_i W^{in}_{ji} x_i(t - d^{in}_{ji}) + \sum_i W^{rec}_{ji} z_i(t - d^{rec}_{ji}), \tag{2.2}$$

where W^{in}_{ji} and W^{rec}_{ji} denote respectively the input and the recurrent synaptic weights and d^{in}_{ji} and d^{rec}_{ji} the corresponding synaptic delays.

2.4. Spike-frequency adaptation

SFA denotes a feature of spiking neurons where their preceding firing activity transiently increases their firing threshold. Experimental data from the Allen Institute (Allen Institute, 2018) show that a substantial fraction

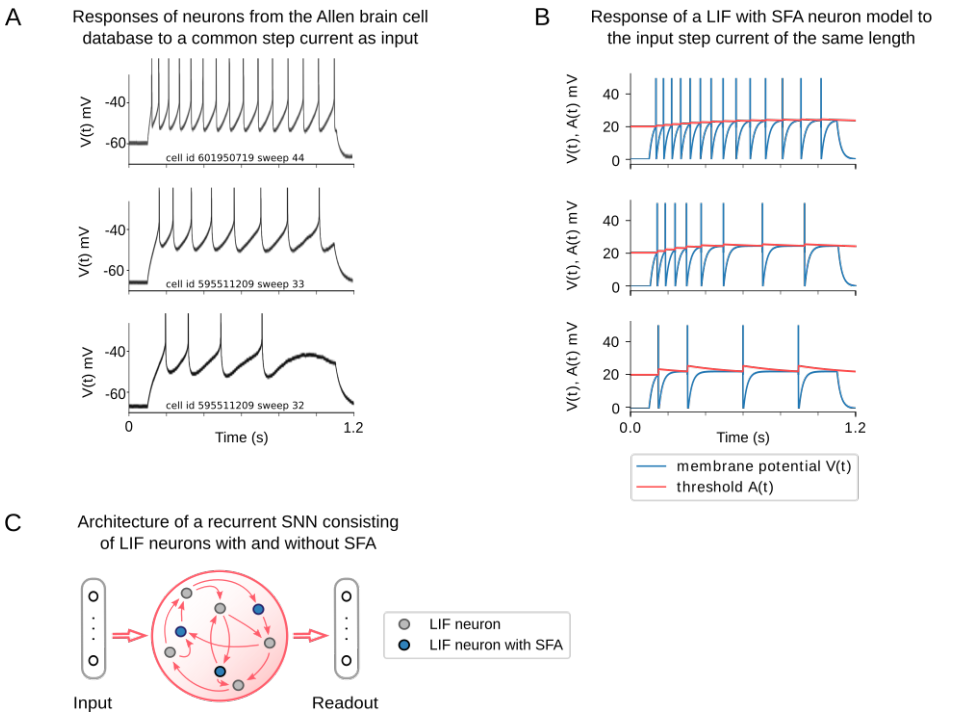

Fig. 2.2.: Experimental data on neurons with SFA, and a simple model for SFA. (**A**) The response to a 1-second long step current is displayed for three sample neurons from the Allen brain cell database Allen Institute, 2018. The cell id and sweep number identify the exact cell recording in the Allen brain cell database. (**B**) The response of a simple LIF neuron model with SFA to the 1-second long step current. Neuron parameters used: top row $\beta = 0.5\,\mathrm{mV}$, $\tau_a = 1\,\mathrm{s}$, $I_{input} = 0.024\,\mathrm{A}$; middle row $\beta = 1\,\mathrm{mV}$, $\tau_a = 1\,\mathrm{s}$, $I_{input} = 0.024\,\mathrm{A}$; bottom row $\beta = 1\,\mathrm{mV}$, $\tau_a = 300\,\mathrm{ms}$, $I_{input} = 0.022\,\mathrm{A}$. (**C**) Symbolic architecture of recurrent SNN consisting of LIF neurons with SFA.

of excitatory neurons of the neocortex, ranging from 20% in mouse visual cortex to 40% in the human frontal lobe, exhibit SFA, see Fig. 2.2A for sample responses of neurons with different levels of SFA. Although a rigorous survey of time constants of SFA is still missing, the available experimental data show that SFA does produce history dependence of neural firing on the time scale of seconds, in fact, up to 20 seconds according to Pozzorini, Naud, et al., 2013; Pozzorini, Mensi, et al., 2015. The biophysical mechanisms behind SFA include inactivation of depolarizing currents and the activity-dependent activation of slow hyperpolarizing or shunting currents (Gutkin and Zeldenrust, 2014; Benda and Herz, 2003). These have already been implicated in cellular short-term memory (Eve Marder et al., 1996; Turrigiano, Marder, and L. Abbott, 1996). SFA has also been argued to contribute to a number of other important features of brain networks Gutkin and Zeldenrust, 2014. On the single neuron level, these features include the enhancement of sensitivity to synchronous input and effects on the frequency response curve (Benda, Maler, and Longtin, 2010; Ermentrout, 1998; X.-J. Wang, 1998). On the network level, SFA may influence population coding or benefit Bayesian inference (Kilpatrick and Ermentrout, 2011; Deneve, 2008). The contribution of SFA to temporal computing capabilities of recurrent SNNs had first been examined in Bellec, Salaj, et al., 2018. The role of SFA for language processing in feedforward networks was subsequently examined in Fitz et al., 2020.

Different ways for fitting models for neurons with SFA to data are described in Gerstner, W. M. Kistler, et al., 2014; N. W. Gouwens et al., 2018. We employ a very simple model for SFA, the generalized leaky integrate-and-fire model $GLIF_2$ from Teeter et al., 2018; Allen Institute, 2017. A practical advantage of this simple model is that it can be very efficiently simulated and is amenable to gradient descent training methods. It assumes that the firing threshold $A(t)$ of a LIF neuron contains a variable component $a(t)$ that increases by a fixed amount after each of its spikes $z(t)$ (Fig. 2.2B), and then decays exponentially back to 0. This variable threshold models the inactivation of voltage-dependent sodium channels in a qualitative manner. We write $z_j(t)$ for the spike output of neuron j, that switches from 0 to 1 at time t when the neuron fires at time t, and otherwise has value 0. Between spikes of neuron j, the membrane voltage $V_j(t)$ and the threshold $A_j(t)$ are following

the dynamics

$$\tau_m \dot{V}_j(t) = -V_j(t) + R_m I_j(t) \tag{2.3}$$
$$\tau_{a,j} \dot{A}_j(t) = v_{\text{th}} - A_j(t), \tag{2.4}$$

where $\tau_{a,j}$ is the adaptation time constant. In discreete time one can define the SFA model by the equations:

$$V_j(t+\delta t) = \alpha V_j(t) + (1-\alpha) R_m I_j(t) - A_j(t) z_j(t) \delta t, \tag{2.5}$$
$$A_j(t) = v_{\text{th}} + \beta a_j(t), \tag{2.6}$$
$$a_j(t+\delta t) = \rho_j a_j(t) + (1-\rho_j) z_j(t) \delta t, \tag{2.7}$$

where v_{th} is the constant baseline of the firing threshold $A_j(t)$, and $\beta > 0$ scales the amplitude of the activity-dependent component. The parameter $\rho_j = \exp\left(\frac{-\delta t}{\tau_{a,j}}\right)$ controls the speed by which $a_j(t)$ decays back to 0, where $\tau_{a,j}$ is the adaptation time constant of neuron j. The term $A_j(t) z_j(t) \delta t$ implements the reset of the membrane voltage after each spike. The current $I_j(t)$ is the weighted sum of the incoming spikes. The definition of the input current in equation (2.2) holds also for discrete time, with the difference that spike trains now assume values in $\{0, \frac{1}{\delta t}\}$. In all our simulations, δt was set to 1 ms unless specified differently.

The spiking output of LIF neuron with SFA j is then defined by $z_j(t) = H\left(\frac{V_j(t) - A_j(t)}{A_j(t)}\right) \frac{1}{\delta t}$.

Adaptation time constants of neurons with SFA were chosen to match the task requirements in individual chapters while still conforming to the experimental data from rodents (Allen Institute, 2018; Pozzorini, Naud, et al., 2013; Pozzorini, Mensi, et al., 2015; Mensi et al., 2012). For an analysis of the impact of the adaptation time constants on the performance see Table 4.1 in Chapter 4.

2.5. Training method

In order to demonstrate the contribution of SFA and other mechanisms on temporal computing capabilities of SNNs, we optimized the weights of the

SNN for each temporal computing tasks. We used Backpropagation through time (BPTT) (Werbos, 1990) for this, which is arguably the best performing optimization method for SNNs that is currently known.

In artificial recurrent neural networks, such as LSTMs, gradients can be computed with BPTT using the automatic differentiation libraries. For BPTT in spiking neural networks, complications arise from the non-differentiability of the output of spiking neurons, and from the fact that gradients need to be propagated either through continuous time or through many time steps if time is discretized. Therefore, in Courbariaux et al., 2016; Esser et al., 2016 it was proposed to use a pseudo-derivative that smoothly increases from 0 to 1, and then decays back to 0:

$$\frac{dz_j(t)}{dv_j(t)} := \max\{0, 1 - |v_j(t)|\}, \tag{2.8}$$

where $v_j(t)$ denotes the normalized membrane potential $v_j(t) = \frac{V_j(t) - A_j(t)}{A_j(t)}$. This made it possible to train deep feed-forward networks of deterministic binary neurons (Courbariaux et al., 2016; Esser et al., 2016). We observed that this convention tends to be unstable for very deep (unrolled) recurrent networks of spiking neurons. To achieve stable performance we dampened the increase of back propagated errors through spikes by using a pseudo-derivative of amplitude $\gamma < 1$ (typically $\gamma = 0.3$):

$$\frac{dz_j(t)}{dv_j(t)} := \gamma \max\{0, 1 - |v_j(t)|\}. \tag{2.9}$$

Note that in adaptive neurons, gradients can propagate through many time steps in the dynamic threshold. This propagation is not affected by the dampening.

3. Long short-term memory and learning-to-learn in networks of spiking neurons

Contents

Abstract. Recurrent networks of spiking neurons (RSNNs) underlie the astounding computing and learning capabilities of the brain. But computing and learning capabilities of RSNN models have remained poor, at least in comparison with artificial neural networks (ANNs). We address two possible reasons for that. One is that RSNNs in the brain are not randomly connected or designed according to simple rules, and they do not start learning as a tabula rasa network. Rather, RSNNs in the brain were optimized for their tasks through evolution, development, and prior experience. Details of these optimization processes are largely unknown. But their functional

contribution can be approximated through powerful optimization methods, such as backpropagation through time (BPTT).

A second major mismatch between RSNNs in the brain and models is that the latter only show a small fraction of the dynamics of neurons and synapses in the brain. We include neurons in our RSNN model that reproduce one prominent dynamical process of biological neurons that takes place at the behaviourally relevant time scale of seconds: spike-frequency adaptation (SFA). We denote these networks as LSNNs because of their Long short-term memory. The inclusion of adapting neurons — neurons that exhibit SFA — drastically increases the computing and learning capability of RSNNs if they are trained and configured by deep learning (BPTT combined with a rewiring algorithm that optimizes the network architecture). In fact, the computational performance of these RSNNs approaches for the first time that of LSTM networks. In addition RSNNs with SFA can acquire abstract knowledge from prior learning in a Learning-to-Learn (L2L) scheme, and transfer that knowledge in order to learn new but related tasks from very few examples. We demonstrate this for supervised learning and reinforcement learning.

Acknowledgments and author contributions. This chapter is based on the manuscript

> Guillaume Bellec*, Darjan Salaj*, Anand Subramoney*, Robert Legenstein, Wolfgang Maass (2018). "Long short-term memory and learning-to-learn in networks of spiking neurons." *Advances in Neural Information Processing Systems 31, 787-797.*

To this study, I contributed as first author together with GB and AS. The study was conceived by GB, DS, AS, WM. The experiments were designed by DS, AS, GB, WM and were conducted by DS, AS, GB. The manuscript was written by GB, DS, AS, RL, WM.

3.1. Introduction

Recurrent networks of spiking neurons (RSNNs) are frequently studied as models for networks of neurons in the brain. In principle, they should be especially well-suited for computations in the temporal domain, such as speech processing, as their computations are carried out via spikes, i.e., events in time and space. But the performance of RSNN models has remained suboptimal also for temporal processing tasks. One difference between RSNNs in the brain and RSNN models is that RSNNs in the brain have been optimized for their function through long evolutionary processes, complemented by a sophisticated learning curriculum during development. Since most details of these biological processes are currently still unknown, we asked whether deep learning is able to mimic these complex optimization processes on a functional level for RSNN models. We used BPTT as the deep learning method for network optimization. Backpropagation has been adapted previously for feed forward networks with binary activations in (Courbariaux et al., 2016; Esser et al., 2016), and we adapted BPTT to work in a similar manner for RSNNs. In order to also optimize the connectivity of RSNNs, we augmented BPTT with DEEP R, a biologically inspired heuristic for synaptic rewiring (Kappel, Legenstein, et al., 2018; Bellec, Kappel, et al., 2018). Compared to LSTM networks, RSNNs tend to have inferior short-term memory capabilities. Since neurons in the brain are equipped with a host of dynamics processes on time scales larger than a few dozen ms (Hasson, Chen, and Honey, 2015), we enriched the inherent dynamics of neurons in our model by a standard neural SFA process.

We first show (section 3.4) that this approach produces new computational performance levels of RSNNs for two common benchmark tasks: Sequential MNIST and TIMIT (a speech processing task). We then show that it makes L2L applicable to RSNNs (section 3.5), similarly as for LSTM networks. In particular, we show that meta-RL (J. X. Wang, Kurth-Nelson, Tirumala, et al., 2016; Duan et al., 2016) produces new motor control capabilities of RSNNs (section 3.6). This result links a recent abstract model for reward-based learning in the brain J. X. Wang, Kurth-Nelson, Kumaran, et al., 2018 to spiking activity. In addition, we show that RSNNs with sparse connectivity and sparse firing activity of 10-20 Hz can solve these and other tasks. Hence these RSNNs compute with spikes, rather than firing rates.

The superior computing and learning capabilities of LSNNs suggest that they are also of interest for implementation in spike-based neuromorphic chips such as Brainscales (Schemmel et al., 2010), SpiNNaker (Furber, Lester, et al., 2013), True North (Esser et al., 2016), chips from ETH Zürich (Qiao et al., 2015), and Loihi (Davies et al., 2018). In particular, nonlocal learning rules such as backprop are challenges for some of these neuromorphic devices (and for many brain models). Hence alternative methods for RSNN learning of nonlinear functions are needed. We show in sections 3.5 and 3.6 that L2L can be used to generate RSNNs that learn very efficiently even in the absence of synaptic plasticity.

Relation to prior work: We refer to (Eliasmith, 2013; DePasquale, Churchland, and L. Abbott, 2016; Huh and Sejnowski, 2018; Nicola and Clopath, 2017) for summaries of preceding results on computational capabilities of RSNNs. The focus there was typically on the generation of dynamic patterns. Such tasks are not addressed in this article, but it is shown in (Bellec, Scherr, Subramoney, et al., 2020) that LSNNs provide an alternative model to Nicola and Clopath, 2017 for the generation of complex temporal patterns. Huh et al. (Huh and Sejnowski, 2018) applied gradient descent to recurrent networks of spiking neurons. There, neurons without a leak were used. Hence, the voltage of a neuron could used in that approach to store information over an unlimited length of time.

We are not aware of previous attempts to bring the performance of RSNNs for time series classification into the performance range of LSTM networks. We are also not aware of any previous literature on applications of L2L to SNNs.

3.2. LSNN model

Neurons and synapses in common RSNN models are missing many of the dynamic processes found in their biological counterparts, especially those on larger time scales. We integrate SFA into our RSNN model. We refer to the resulting type of RSNNs as Long short-term memory Spiking Neural Networks (LSNNs). LSNNs consist of a population R of integrate-and-fire (LIF) neurons (excitatory and inhibitory), and a second population A of

LIF excitatory neurons whose excitability is temporarily reduced through preceding firing activity, i.e., these neurons exhibit SFA (see Fig. 3.1C and Suppl.). Both populations R and A receive spike trains from a population X of external input neurons. Results of computations are read out by a population Y of external linear readout neurons, see Fig. 3.1C.

3.3. Applying BPTT with DEEP R to RSNNs and LSNNs

We optimize the synaptic weights, and in some cases also the connectivity matrix of an LSNN for specific ranges of tasks. The optimization algorithm that we use, backpropagation through time (BPTT), is not claimed to be biologically realistic. But like evolutionary and developmental processes, BPTT can optimize LSNNs for specific task ranges. How the BPTT is applied to non-differentiable activation function of spiking neurons is described in Chapter 2. Additionaly we reduced ("dampened") the amplitude of the pseudo-derivative by a factor < 1 (see Suppl. for details). This enhances the performance of BPTT for RSNNs that compute during larger time spans, that require backpropagation through several 1000 layers of an unrolled feedforward network of spiking neurons. A similar implementation of BPTT for RSNNs was proposed in (Huh and Sejnowski, 2018). It is not yet clear which of these two versions of BPTT work best for a given task and a given network.

In order to optimize not only the synaptic weights of a RSNN but also its connectivity matrix, we integrated BPTT with the biologically inspired (Kappel, Legenstein, et al., 2018) rewiring method DEEP R (Bellec, Kappel, et al., 2018) (see Suppl. for details). DEEP R converges theoretically to an optimal network configuration by continuously updating the set of active connections (Kappel, Habenschuss, et al., 2015; Kappel, Legenstein, et al., 2018; Bellec, Kappel, et al., 2018).

3.4. Computational performance of LSNNs

Sequential MNIST: We tested the performance of LSNNs on a standard benchmark task that requires continuous updates of short term memory over a long time span: sequential MNIST (Le, Jaitly, and G. E. Hinton, 2015; Costa et al., 2017). We compare the performance of LSNNs with that of LSTM networks. The size of the LSNN, in the case of full connectivity, was chosen to match the number of parameters of the LSTM network. This led to 120 regular spiking and 100 adaptive neurons (with adaptation time constant τ_a of 700 ms) in comparison to 128 LSTM units. Actually it turned out that the sparsely connected LSNN shown in Fig. 3.1C, which was generated by including DEEP R in BPTT, had only 12% of the synaptic connections but performed better than the fully connected LSNN (see "DEEP R LSNN" versus "LSNN" in Fig. 3.1B).

The task is to classify the handwritten digits of the MNIST dataset when the pixels of each handwritten digit are presented sequentially, one after the other in 784 steps, see Fig. 3.1A. After each presentation of a handwritten digit, the network is required to output the corresponding class. The grey values of pixels were given directly to artificial neural networks (ANNs), and encoded by spikes for RSNNs. We considered both the case of step size 1 ms (requiring 784 ms for presenting the input image) and 2 ms (requiring 1568 ms for each image, the adaptation time constant τ_a was set to 1400 ms in this case, see Fig. 3.1B.). The top row of Fig. 3.1D shows a version where the grey value of the currently presented pixel is encoded by population coding through the firing probability of the 80 input neurons. Somewhat better performance was achieved when each of the 80 input neurons is associated with a particular threshold for the grey value, and this input neuron fires whenever the grey value crosses its threshold in the transition from the previous to the current pixel (this input convention is chosen for the SNN results of Fig. 3.1B). In either case, an additional input neuron becomes active when the presentation of the 784 pixel values is finished, in order to prompt an output from the network. The firing of this additional input neuron is shown at the top right of the top panel of Fig. 3.1D. The softmax of 10 linear output neurons Y is trained through BPTT to produce, during this time segment, the label of the sequentially presented handwritten digit. We refer to the yellow shading around 800 ms of the output neuron for label

3 in the plot of the dynamics of the output neurons Y in Fig. 3.1D. This output was correct.

A performance comparison is given in Fig. 3.1B. LSNNs achieve 94.7% and 96.4% classification accuracy on the test set when every pixel is presented for 1 and 2ms respectively. An LSTM network achieves 98.5% and 98.0% accuracy on the same task setups. The LIF and RNN bars in Fig. 3.1B show that this accuracy is out of reach for BPTT applied to spiking or nonspiking neural networks without enhanced short term memory capabilities. We observe that in the sparse architecture discovered by DEEP R, the connectivity onto the readout neurons Y is denser than in the rest of the network (see Fig. 3.1C). Detailed results are given in the supplement.

Speech recognition (TIMIT): We also tested the performance of LSNNs for a real-world speech recognition task, the TIMIT dataset. A thorough study of the performance of many variations of LSTM networks on TIMIT has recently been carried out in (Greff et al., 2017). We used exactly the same setup which was used there (framewise classification) in order to facilitate comparison. We found that a standard LSNN consisting of 300 regularly firing (200 excitatory and 100 inhibitory) and 100 excitatory adapting neurons with an adaptation time constant of 200 ms, and with 20% connection probability in the network, achieved a classification error of 33.2%. This error is below the mean error around 40% from 200 trials with different hyperparameters for the best performing (and most complex) version of LSTMs according to Fig. 3 of (Greff et al., 2017), but above the mean of 29.7% of the 20 best performing choices of hyperparameters for these LSTMs. The performance of the LSNN was however somewhat better than the error rates achieved in (Greff et al., 2017) for a less complex version of LSTMs without forget gates (mean of the best 20 trials: 34.2%).

We could not perform a similarly rigorous search over LSNN architectures and meta-parameters as was carried out in (Greff et al., 2017) for LSTMs. But if all adapting neurons are replaced by regularly firing excitatory neurons one gets a substantially higher error rate than the LSNN with adapting neurons: 37%. Details are given in the supplement.

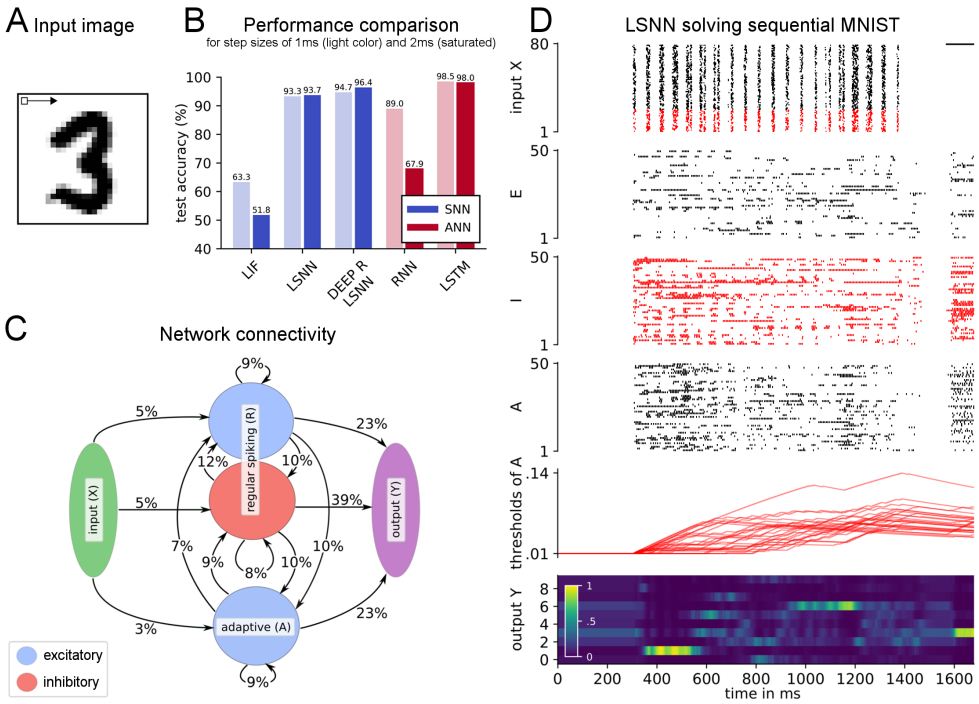

Fig. 3.1.: Sequential MNIST. (**A**) The task is to classify images of handwritten digits when the pixels are shown sequentially pixel by pixel, in a fixed order row by row. (**B**) The performance of RSNNs is tested for three different setups: without adapting neurons (LIF), a fully connected LSNN, and an LSNN with randomly initialized connectivity that was rewired during training (DEEP R LSNN). For comparison, the performance of two ANNs, a fully connected RNN and an LSTM network are also shown. (**C**) Connectivity (in terms of connection probabilities between and within the 3 subpopulations) of the LSNN after applying DEEP R in conjunction with BPTT. The input population X consisted of 60 excitatory and 20 inhibitory neurons. Percentages on the arrows from X indicate the average connection probabilities from excitatory and inhibitory neurons. (**D**) Dynamics of the LSNN after training when the input image from A was sequentially presented. From top to bottom: spike rasters from input neurons (X), and random subsets of excitatory (E) and inhibitory (I) regularly spiking neurons, and adaptive neurons (A), dynamics of the firing thresholds of a random sample of adaptive neurons; activation of softmax readout neurons.

3.5. LSNNs learn-to-learn from a teacher

One likely reason why learning capabilities of RSNN models have remained rather poor is that one usually requires a tabula rasa RSNN model to learn. In contrast, RSNNs in the brain have been optimized through a host of preceding processes, from evolution to prior learning of related tasks, for their learning performance. We emulate a similar training paradigm for RSNNs using the L2L setup. We explore here only the application of L2L to LSNNs, but L2L can also be applied to RSNNs without adapting neurons (Subramoney et al., 2018). An application of L2L to LSNNs is tempting, since L2L is most commonly applied in machine learning to their ANN counterparts: LSTM networks see e.g. (J. X. Wang, Kurth-Nelson, Tirumala, et al., 2016; Duan et al., 2016). LSTM networks are especially suited for L2L since they can accommodate two levels of learning and representation of learned insight: Synaptic connections and weights can encode, on a higher level, a learning algorithm and prior knowledge on a large time-scale. The short-term memory of an LSTM network can accumulate, on a lower level of learning, knowledge during the current learning task. It has recently been argued J. X. Wang, Kurth-Nelson, Kumaran, et al., 2018 that the pre-frontal cortex (PFC) similarly accumulates knowledge during fast reward-based learning in its short-term memory, without using dopamine-gated synaptic plasticity, see the text to Suppl. Fig. 3 in (J. X. Wang, Kurth-Nelson, Kumaran, et al., 2018). The experimental results of Perich, Gallego, and L. E. Miller, 2018 suggest also a prominent role of short-term memory for fast learning in the motor cortex.

The standard setup of L2L involves a large, in fact in general infinitely large, family $\mathcal{F}$ of learning tasks C. Learning is carried out simultaneously in two loops (see Fig. 3.2A). The *inner loop* learning involves the learning of a single task C by a neural network $\mathcal{N}$, in our case by an LSNN. Some parameters of $\mathcal{N}$ (termed hyper-parameters) are optimized in an *outer loop* optimization to support fast learning of a randomly drawn task C from $\mathcal{F}$. The outer loop training – implemented here through BPTT – proceeds on a much larger time scale than the inner loop, integrating performance evaluations from many different tasks C of the family $\mathcal{F}$. One can interpret this outer loop as a process that mimics the impact of evolutionary and developmental optimization processes, as well as prior learning, on the learning capability

of brain networks. We use the terms training and optimization interchangeably, but the term training is less descriptive of the longer-term evolutionary processes we mimic. Like in (Hochreiter, Younger, and Conwell, 2001; J. X. Wang, Kurth-Nelson, Tirumala, et al., 2016; Duan et al., 2016) we let all synaptic weights of $\mathcal{N}$ belong to the set of hyper-parameters that are optimized through the outer loop. Hence the network is forced to encode all results from learning the current task C in its internal state, in particular in its firing activity and the thresholds of adapting neurons. Thus the synaptic weights of the neural network $\mathcal{N}$ are free to encode an efficient *algorithm* for learning arbitrary tasks C from $\mathcal{F}$.

When the brain learns to predict sensory inputs, or state changes that result from an action, this can be formalized as learning from a teacher (i.e., supervised learning). The teacher is in this case the environment, which provides – often with some delay – the target output of a network. The L2L results of (Hochreiter, Younger, and Conwell, 2001) show that LSTM networks can learn nonlinear functions from a teacher without modifying their synaptic weights, using their short-term memory instead. We asked whether this form of learning can also be attained by LSNNs.

Task: We considered the task of learning complex non-linear functions from a teacher. Specifically, we chose as family $\mathcal{F}$ of tasks a class of continuous functions of two real-valued variables (x_1, x_2). This class was defined as the family of all functions that can be computed by a 2-layer artificial neural network of sigmoidal neurons with 10 neurons in the hidden layer, and weights and biases from [-1, 1], see Fig. 3.2B. Thus overall, each such target network (TN) from $\mathcal{F}$ was defined through 40 parameters in the range [-1, 1]: 30 weights and 10 biases. We gave the teacher input to the LSNN for learning a particular TN C from $\mathcal{F}$ in a delayed manner as in (Hochreiter, Younger, and Conwell, 2001): The target output value was given after $\mathcal{N}$ had provided its guessed output value for the preceding input.

This delay of the feedback is consistent with biologically plausible scenarios. Simultaneously, having a delay for the feedback prevents $\mathcal{N}$ from passing on the teacher value as output without first producing a prediction on its own.

Implementation: We considered a LSNN $\mathcal{N}$ consisting of 180 regularly firing neurons (population R) and 120 adapting neurons (population A) with

a spread of adaptation time constants sampled uniformly between 1 and 1000 ms and with full connectivity. Sparse connectivity in conjunction with rewiring did not improve performance in this case. All neurons in the LSNN received input from a population X of 300 external input neurons. A linear readout received inputs from all neurons in R and A. The LSNN received a stream of 3 types of external inputs (see top row of Fig. 3.2D): the values of x_1, x_2, and of the output $C(x_1', x_2')$ of the TN for the preceding input pair x_1', x_2' (set to 0 at the first trial), all represented through population coding in an external population of 100 spiking neurons. It produced outputs in the form of weighted spike counts during 20 ms windows from all neurons in the network (see bottom row of Fig. 3.2D), where the weights for this linear readout were trained, like all weights inside the LSNN, in the outer loop, and remained fixed during learning of a particular TN.

The training procedure in the outer loop of L2L was as follows: Network training was divided into training episodes. At the start of each training episode, a new target network TN was randomly chosen and used to generate target values $C(x_1, x_2) \in [0, 1]$ for randomly chosen input pairs (x_1, x_2). 500 of these input pairs and targets were used as training data, and presented one per step to the LSNN during the episode, where each step lasted 20 ms. LSNN parameters were updated using BPTT to minimize the mean squared error between the LSNN output and the target in the training set, using gradients computed over batches of 10 such episodes, which formed one iteration of the outer loop. In other words, each weight update included gradients calculated on the input/target pairs from 10 different TNs. This training procedure forced the LSNN to adapt its parameters in a way that supported learning of many different TNs, rather than specializing on predicting the output of single TN. After training, the weights of the LSNN remained fixed, and it was required to learn the input/output behaviour of TNs from $\mathcal{F}$ that it had never seen before in an online manner by just using its short-term memory and dynamics. See the suppl. for further details.

Results: Most of the functions that are computed by TNs from the class $\mathcal{F}$ are nonlinear, as illustrated in Fig. 3.2G for the case of inputs (x_1, x_2) with $x_1 = x_2$. Hence learning the input/output behaviour of any such TN with biologically realistic local plasticity mechanisms presents a daunting challenge for a SNN. Fig. 3.2C shows that after a few thousand training iterations in the outer loop, the LSNN achieves low MSE for learning new

TNs from the family $\mathcal{F}$, significantly surpassing the performance of an optimal linear approximator (linear regression) that was trained on all 500

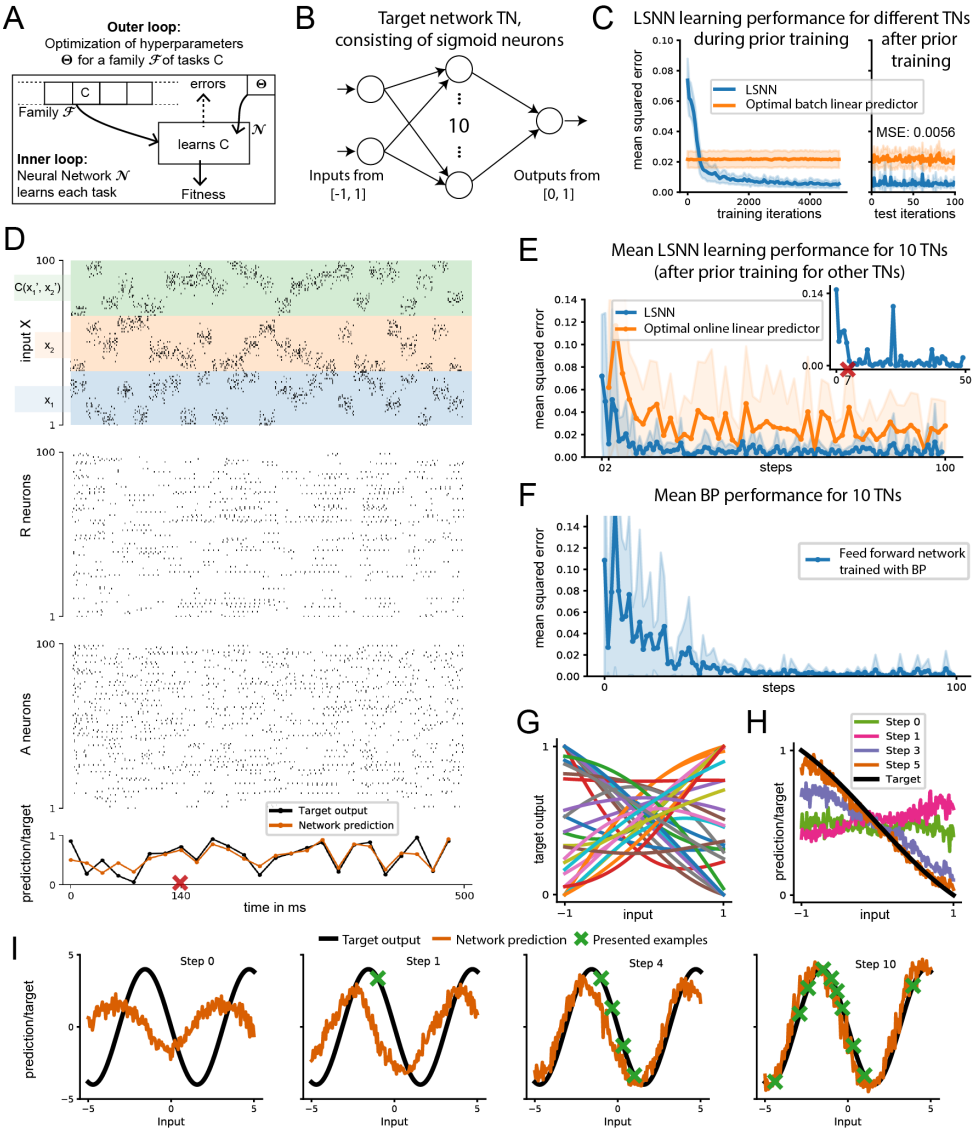

Fig. 3.2.: LSNNs learn to learn from a teacher. (Caption on the next page.)

Fig. 3.2.: LSNNs learn to learn from a teacher. (**A**) L2L scheme for an SNN $\mathcal{N}$. (**B**) Architecture of the two-layer feed-forward target networks (TNs) used to generate nonlinear functions for the LSNN to learn; weights and biases were randomly drawn from [-1,1]. (**C**) Performance of the LSNN in learning a new TN during (left) and after (right) training in the outer loop of L2L. Performance is compared to that of an optimal linear predictor fitted to the batch of all 500 experiments for a TN. (**D**) Network input (top row, only 100 of 300 neurons shown), internal spike-based processing with low firing rates in the populations R and A (middle rows), and network output (bottom row) for 25 trials of 20 ms each. (**E**) Learning performance of the LSNN for 10 new TNs. Performance for a single TN is shown as insert, a red cross marks step 7 after which output predictions became very good for this TN. The spike raster for this learning process is the one depicted in C. Performance is compared to that of an optimal linear predictor, which, for each example, is fitted to the batch of all preceding examples. (**F**) Learning performance of BP for the same 10 TNs as in D, working directly on the ANN from A, with a prior for small weights. (**G**) Sample input/output curves of TNs on a 1D subset of the 2D input space, for different weight and bias values. (**H**) These curves are all fairly smooth, like the internal models produced by the LSNN while learning a particular TN. (**I**) Illustration of the prior knowledge acquired by the LSNN through L2L for another family $\mathcal{F}$ (sinus functions). Even adversarially chosen examples (Step 4) do not induce the LSNN to forget its prior.

pairs of inputs and target outputs, see orange curve in Fig. 3.2C,E. In view of the fact that each TN is defined by 40 parameters, it comes at some surprise that the resulting network learning algorithm of the LSNN for learning the input/output behaviour of a new TN produces in general a good approximation of the TN after just 5 to 20 trials, where in each trial one randomly drawn labelled example is presented. One sample of a generic learning process is shown in Fig. 3.2D. Each sequence of examples evokes an internal model that is stored in the short-term memory of the LSNN. Fig. 3.2H shows the fast evolution of internal models of the LSNN for the TN during the first trials (visualized for a 1D subset of the 2D input space). We make the current internal model of the LSNN visible by probing its prediction $C(x_1, x_2)$ for hypothetical new inputs for evenly spaced points (x_1, x_2) in the domain (without allowing it to modify its short-term memory; all other inputs advance the network state according to the dynamics of the LSNN). One sees that the internal model of the LSNN is from the beginning a smooth function, of the same type as the ones defined by the TNs in $\mathcal{F}$. Within a few trials this smooth function approximated the TN quite well.

Hence the LSNN had acquired during the training in the outer loop of L2L a prior for the types of functions that are to be learnt, that was encoded in its synaptic weights. This prior was in fact quite efficient, since Fig. 3.2E and F show that the LSNN was able to learn a TN with substantially fewer trials than a generic learning algorithm for learning the TN directly in an artificial neural network as in Fig. 3.2A: BP with a prior that favored small weights and biases (see end of Sec. 3 in suppl.). These results suggest that L2L is able to install some form of prior knowledge about the task in the LSNN. We conjectured that the LSNN fits internal models for smooth functions to the examples it received.

We tested this conjecture in a second, much simpler, L2L scenario. Here the family $\mathcal{F}$ consisted of all sinus functions with arbitrary phase and amplitudes between 0.1 and 5. Fig. 3.2I shows that the LSNN also acquired an internal model for sinus functions (made visible analogously as in Fig. 3.2H) in this setup from training in the outer loop. Even when we selected examples in an adversarial manner, which happened to be in a straight line, this did not disturb the prior knowledge of the LSNN.

Altogether the network learning that was induced through L2L in the LSNNs is of particular interest from the perspective of the design of learning algorithms, since we are not aware of previously documented methods for installing structural priors for online learning of a recurrent network of spiking neurons.

3.6. LSNNs learn-to-learn from reward

We now turn to an application of meta reinforcement learning (meta-RL) to LSNNs. In meta-RL, the LSNN receives rewards instead of teacher inputs. Meta-RL has led to a number of remarkable results for LSTM networks, see e.g. (J. X. Wang, Kurth-Nelson, Tirumala, et al., 2016; Duan et al., 2016). In addition, J. X. Wang, Kurth-Nelson, Kumaran, et al., 2018 demonstrates that meta-RL provides a very interesting perspective of reward-based learning in the brain. We focused on one of the more challenging demos of J. X. Wang, Kurth-Nelson, Tirumala, et al., 2016 and Duan et al., 2016, where an agent had to learn to find a target in a 2D arena, and to navigate subsequently

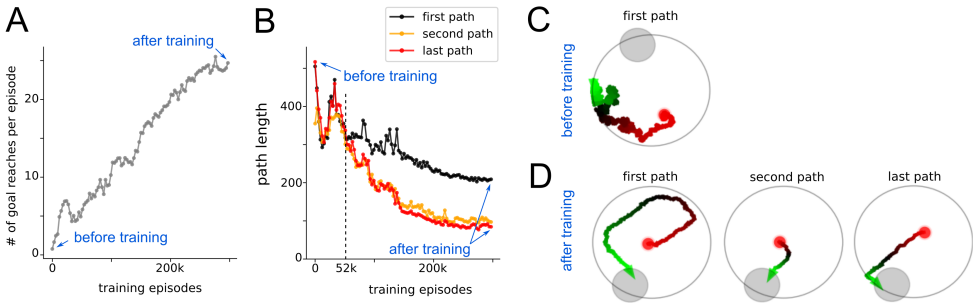

Fig. 3.3.: Meta-RL results for an LSNN. (**A, B**) Performance improvement during training in the outer loop. (**C, D**) Samples of navigation paths produced by the LSNN before and after this training. Before training, the agent performs a random walk (**C**). In this example it does not find the goal within the limited episode duration. After training (**D**), the LSNN had acquired an efficient exploration strategy that uses two pieces of abstract knowledge: that the goal always lies on the border, and that the goal position is the same throughout an episode. Note that all synaptic weights of the LSNNs remained fixed after training.

to this target from random positions in the arena. This task is related to the well-known biological learning paradigm of the Morris water maze task (Morris, 1984; Vasilaki et al., 2009). We study here the capability of an agent to discover two pieces of abstract knowledge from the concrete setup of the task: the distribution of goal positions, and the fact that the goal position is constant within each episode. We asked whether the agent would be able to exploit the pieces of abstract knowledge from learning for many concrete episodes, and use it to navigate more efficiently.

Task: An LSNN-based agent was trained on a family of navigation tasks with continuous state and action spaces in a circular arena. The task is structured as a sequence of episodes, each lasting 2 seconds. The goal was placed randomly for each episode on the border of the arena. When the agent reached the goal, it received a reward of 1, and was placed back randomly in the arena. When the agent hit a wall, it received a negative reward of -0.02 and the velocity vector was truncated to remain inside the arena. The objective was to maximize the number of goals reached within the episode. This family $\mathcal{F}$ of tasks is defined by the infinite set of possible goal positions. For each episode, an optimal agent is expected to explore until it finds the goal position, memorize it and exploits this knowledge

until the end of the episode by taking the shortest path to the goal. We trained an LSNN so that the network could control the agent's behaviour in all tasks, without changing its network weights.

Implementation: Since LSNNs with just a few hundred neurons are not able to process visual input, we provided the current position of the agent within the arena through a place-cell like Gaussian population rate encoding of the current position. The lack of visual input made it already challenging to move along a smooth path, or to stay within a safe distance from the wall. The agent received information about positive and negative rewards in the form of spikes from external neurons. For training in the outer loop, we used BPTT together with DEEP R applied to the surrogate objective of the Proximal Policy Optimization (PPO) algorithm Schulman et al., 2017. In this task the LSNN had 400 recurrent units (200 excitatory, 80 inhibitory and 120 adaptive neurons with adaptation time constant τ_a of 1200 ms), the network was rewired with a fixed connectivity of 20%. The resulting network diagram and spike raster is shown in Suppl. Fig. 3.4.

Results: The network behaviour before, during, and after L2L optimization is shown in Fig. 3.3. Fig. 3.3A shows that a large number of training episodes finally provides significant improvements. With a close look at Fig. 3.3B, one sees that before 52k training episodes, the intermediate path planning strategies did not seem to use the discovered goal position to make subsequent paths shorter. Hence the agents had not yet discovered that the goal position does not change during an episode. After training for 300k episodes, one sees from the sample paths in Fig. 3.3D that both pieces of abstract knowledge had been discovered by the agent. The first path in Fig. 3.3D shows that the agent exploits that the goal is located on the border of the maze. The second and last paths show that the agent knows that the position is fixed throughout an episode. Altogether this demo shows that meta-RL can be applied to RSNNs, and produces previously not seen capabilities of sparsely firing RSNNs to extract abstract knowledge from experimentation, and to use it in clever ways for controlling behaviour.

3.7. Discussion

We have demonstrated that deep learning provides a useful new tool for the investigation of networks of spiking neurons: It allows us to create architectures and learning algorithms for RSNNs with enhanced computing and learning capabilities. In order to demonstrate this, we adapted BPTT so that it works efficiently for RSNNs, and can be combined with a biologically inspired synaptic rewiring method (DEEP R). We have shown in section 3.4 that this method allows us to create sparsely connected RSNNs that approach the performance of LSTM networks on common benchmark tasks for the classification of spatio-temporal patterns (sequential MNIST and TIMIT). This qualitative jump in the computational power of RSNNs was supported by the introduction of adapting neurons into the model. Adapting neurons introduce a spread of longer time constants into RSNNs, as they do in the neocortex according to Allen Institute, 2018. We refer to the resulting variation of the RSNN model as LSNNs, because of the resulting longer short-term memory capability. This form of short-term memory is of particular interest from the perspective of energy efficiency of SNNs, because it stores and transmits stored information through non-firing of neurons: A neuron that holds information in its increased firing threshold tends to fire less often.

We have shown in Fig. 3.2 that an application of deep learning (BPTT and DEEP R) in the outer loop of L2L provides a new paradigm for learning of nonlinear input/output mappings by a RSNN. This learning task was thought to require an implementation of BP in the RSNN. We have shown that it requires no BP, not even changes of synaptic weights. Furthermore we have shown that this new form of network learning enables RSNNs, after suitable training with similar learning tasks in the outer loop of L2L, to learn a new task from the same class substantially faster. The reason is that the prior deep learning has installed abstract knowledge (priors) about common properties of these learning tasks in the RSNN. To the best of our knowledge, transfer learning capabilities and the use of prior knowledge (see Fig. 3.2I) have previously not been demonstrated for SNNs. Fig 3.3 shows that L2L also embraces the capability of RSNNs to learn from rewards (meta-RL). For example, it enables a RSNN – without any additional outer control or clock – to embody an agent that first searches an arena for a goal, and subsequently

exploits the learnt knowledge in order to navigate fast from random initial positions to this goal. Here, for the sake of simplicity, we considered only the more common case when all synaptic weights are determined by the outer loop of L2L. But similar results arise when only some of the synaptic weights are learnt in the outer loop, while other synapses employ local synaptic plasticity rules to learn the current task Subramoney et al., 2018.

Altogether we expect that the new methods and ideas that we have introduced will advance our understanding and reverse engineering of RSNNs in the brain. For example, the RSNNs that emerged in Fig. 3.1-3.3 all compute and learn with a brain-like sparse firing activity, quite different from a SNN that operates with rate-codes. In addition, these RSNNs present new functional uses of short-term memory that go far beyond remembering a preceding input as in (Mongillo, Barak, and Tsodyks, 2008), and suggest new forms of activity-silent memory (Stokes, 2015).

Apart from these implications for computational neuroscience, our finding that RSNNs can acquire powerful computing and learning capabilities with very energy-efficient sparse firing activity provides new application paradigms for spike-based computing hardware through non-firing.

Acknowledgments

This research/project was supported by the HBP Joint Platform, funded from the European Union's Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreement No. 720270 (Human Brain Project SGA1) and under the Specific Grant Agreement No. 785907 (Human Brain Project SGA2). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Quadro P6000 GPU used for this research. Research leading to these results has in parts been carried out on the Human Brain Project PCP Pilot Systems at the Jülich Supercomputing Centre, which received co-funding from the European Union (Grant Agreement No. 604102). We gratefully acknowledge Sandra Diaz, Alexander Peyser and Wouter Klijn from the Simulation Laboratory Neuroscience of the Jülich Supercomputing Centre for their support. The computational results presented have been achieved in part using the Vienna Scientific Cluster (VSC).

3.8. Methods

LIF neuron model, SFA, and training method are described in Chapter 2.

3.8.1. Rewiring and weight initialization of excitatory and inhibitory neurons

In all experiments except those reported in Fig. 3.2, the neurons were either excitatory or inhibitory. When the neuron sign were not constrained, the initial network weights were drawn from a Gaussian distribution $W_{ji} \sim \frac{w_0}{\sqrt{n_{in}}}\mathcal{N}(0,1)$, where n_{in} is the number of afferent neurons in the considered weight matrix (i.e., the number of columns of the matrix), $\mathcal{N}(0,1)$ is the zero-mean unit-variance Gaussian distribution and w_0 is a weightscaling factor chosen to be $w_0 = \frac{1\text{Volt}}{R_m}\delta t$. With this choice of w_0 the resistance R_m becomes obsolete but the vanishing-exploding gradient theory (Bengio, Simard, and Frasconi, 1994; Sussillo and L. Abbott, 2014) can be used to avoid tuning by hand the scaling of W_{ji}. In particular the scaling $\frac{1}{\sqrt{n_{in}}}$ used above was sufficient to initialize networks with realistic firing rates and that can be trained efficiently.

When the neuron sign were constrained, all outgoing weights W_{ji}^{rec} or W_{ji}^{out} of a neuron i had the same sign. In those cases, DEEP R Bellec, Kappel, et al., 2018 was used as it maintains the sign of each synapse during training. The sign is thus inherited from the initialization of the network weights. This raises the need of an efficient initialization of weight matrices for given fractions of inhibitory and excitatory neurons. To do so, a sign $\kappa_i \in \{-1, 1\}$ is generated randomly for each neuron i by sampling from a Bernoulli distribution. The weight matrix entries are then sampled from $W_{ji} \sim \kappa_i|\mathcal{N}(0,1)|$ and post-processed to avoid exploding gradients. Firstly, a constant is added to each weight so that the sum of excitatory and inhibitory weights onto each neuron j ($\sum_i W_{ji}$) is zero Rajan and L. F. Abbott, 2006 (if j has no inhibitory or no excitatory incoming connections this step is omitted). To avoid exploding gradients it is important to scale the weight so that the largest eigenvalue is lower of equal to 1 (Bengio, Simard, and Frasconi, 1994). Thus, we divided W_{ji} by the absolute value of its largest eigenvalue. When

the matrix is not square, eigenvalues are ill-defined. Therefore, we first generated a large enough square matrix and selected the required number of rows or columns with uniform probabilities. The final weight matrix is scaled by w_0 for the same reasons as before.

To initialize matrices with a sparse connectivity, dense matrices were generated as described above and multiplied with a binary mask. The binary mask was generated by sampling uniformly the neuron coordinates that were non-zero at initialization. DEEP R maintains the initial connectivity level throughout training by dynamically disconnecting synapses and reconnecting others elsewhere. The L_1-norm regularization parameter of DEEP R was set to 0.01 and the temperature parameter of DEEP R was left at 0.

3.8.2. Tasks

Computational performance of LSNNs

MNIST setup: The pixels of an MNIST image were presented sequentially to the LSNN in 784 time steps. Two input encodings were considered. First, we used a population coding where the grey scale value (which is in the range $[0, 1]$) of the currently presented pixel was directly used as the firing probability of each of the 80 input neurons in that time step.

In a second type of input encoding – that is closer to the way how spiking vision sensors encode their input – each of the 80 input neurons was associated with a particular threshold for the grey value, and this input neuron fired whenever the grey value of the currently presented pixel crossed its threshold. Here, we used two input neurons per threshold, one spiked at threshold crossings from below, and one at the crossings from above. This input convention was chosen for the LSNN results of Fig. 3.1B.

The output of the network was determined by averaging the readout output over the 56 time steps following the presentation of the digit. The network was trained by minimizing the cross entropy error between the softmax of the averaged readout and the label distributions. The best performing models use rewiring with a global connectivity level of 12% was used during training to optimize a sparse network connectivity structure (i.e.,

when randomly picking two neurons in the network, the probability that they would be connected is 0.12). This implies that only a fraction of the parameters were finally used as compared to a similarly performing LSTM network.

Tables 3.1 and 3.2 contain the results and details of training runs where each time step lasted for 1 ms and 2 ms respectively.

Model	# neurons	conn.	# params	# runs	mean	std.	max.
LSTM	128	100%	67850	12	79.8%	26.6%	98.5%
RNN	128	100%	17930	10	71.3%	24.5%	89%
LSNN	100(A), 120(R)	12%	8185 (full 68210)	12	94.2%	0.3%	94.7%
LSNN	100(A), 200(R)	12%	14041 (full 117010)	1	-	-	95.7%
LSNN	350(A), 350(R)	12%	66360 (full 553000)	1	-	-	96.1%
LSNN	100(A), 120(R)	100%	68210	10	92.0%	0.7%	93.3%
LIF	220	100%	68210	10	60.9%	2.7%	63.3%

Table 3.1.: Results on the sequential MNIST task when each pixel is displayed for 1ms. For an LSNN, DEEP R is used to optimize the network under a sparse connectivity constraint, we report the number of parameters including and not including the disconnected synapses.

Model	# neurons	conn.	# params	# runs	mean	std.	max.
LSTM	128	100%	67850	12	48.2%	39.9%	98.0%
RNN	128	100%	17930	12	30%	23.6%	67.9%
LSNN	100(A), 120(R)	12%	8185 (full 68210)	12	93.8%	5.8%	96.4%
LSNN	350(A), 350(R)	12%	66360 (full 553000)	1	-	-	97.1%
LSNN	100(A), 120(R)	100%	68210	10	90.5%	1.4%	93.7%
LIF	220	100%	68210	11	34.6%	8.8%	51.8%

Table 3.2.: Results on the sequential MNIST task when each pixel is displayed for 2ms.

TIMIT setup: To investigate if the performance of LSNNs can scale to real world problems, we considered the TIMIT speech recognition task. We focused on the frame-wise classification where the LSNN has to classify each audio-frame to one of the 61 phoneme classes.

We followed the convention of Halberstadt (Glass, Smith, and K. Halberstadt,

1999) for grouping of training, validation, and testing sets (3696, 400, and 192 sequences respectively). The performance was evaluated on the *core test set* for consistency with the literature. Raw audio is preprocessed into 13 Mel Frequency Cepstral Coefficients (MFCCs) with frame size 10 ms and on input window of 25 ms. We computed the first and the second order derivatives of MFCCs and combined them, resulting in 39 input channels. These 39 input channels were mapped to 39 input neurons which unlike in MNIST emit continuous values $x_i(t)$ instead of spikes, and these values were directly used in equation 2.2 for the currents of the postsynaptic neurons.

Since we simulated the LSNN network in 1 ms time steps, every input frame which represents 10 ms of the input audio signal was fed to the LSNN network for 10 consecutive 1 ms steps. The softmax output of the LSNN was averaged over every 10 steps to produce the prediction of the phone in the current input frame. The LSNN was rewired with global connectivity level of 20%.

Parameter values: For adaptive neurons, we used $\beta_j = 1.8$, and for regular spiking neurons we used $\beta_j = 0$ (i.e. A_j is constant). The baseline threshold voltage was $v_{\text{th}} = 0.01$ and the membrane time constant $\tau_m = 20$ ms. Networks were trained using the default Adam optimizer, and a learning rate initialized at 0.01. The dampening factor for training was $\gamma = 0.3$.

For sequential MNIST, all networks were trained for 36000 iterations with a batch size of 256. Learning rate was decayed by a factor 0.8 every 2500 iterations. The adaptive neurons in the LSNN had an adaptation time constant $\tau_a = 700$ ms (1400 ms) for 1 ms (2 ms) per pixel setup. The baseline artificial RNN contained 128 hidden units with the hyperbolic tangent activation function. The LIF network was formed by a fully connected population of 220 regular spiking neurons.

For TIMIT, the LSNN network consisted of 300 regular neurons and 100 adaptive neurons which resulted in approximately 400000 parameters. Network was trained for 80 epochs with batches of 32 sequences. Adaptation time constant of adaptive neurons was set to $\tau_a = 200$ ms. Refractory period of the neurons was set to 2 ms, the membrane time constant of the output Y

neurons to 3 ms, and the synaptic delay was randomly picked from $\{1, 2\}$ ms.

We note that due to the rewiring of the LSNN using DEEP R Bellec, Kappel, et al., 2018 method, only a small fraction of the weights had non-zero values (8185 in MNIST, $\sim$ 80000 in TIMIT).

LSNNs learn-to-learn from a teacher

Experimental setup:

Function families: The LSNN was trained to implement a regression algorithm on a family of functions $\mathcal{F}$. Two specific families were considered: In the first function family, the functions were defined by feed-forward neural networks with 2 inputs, 1 hidden layer consisting of 10 hidden neurons, and 1 output, where all the parameters (weights and biases) were chosen uniformly randomly between $[-1, 1]$. The inputs were between $[-1, 1]$ and the outputs were scaled to be between $[0, 1]$. We call these networks Target Networks (TNs). In the second function family, the targets were defined by sinusoidal functions $y = A \sin(\phi + x)$ over the domain $x \in [-5, 5]$. The specific function to be learned was defined then by the phase ϕ and the amplitude A, which were chosen uniformly random between $[0, \pi]$ and $[0.1, 5]$ respectively.

Input encoding: Analog values were transformed into spiking trains to serve as inputs to the LSNN as follows: For each input component, 100 input neurons are assigned values $m_1, \dots m_{100}$ evenly distributed between the minimum and maximum possible value of the input. Each input neuron has a Gaussian response field with a particular mean and standard deviation, where the means are uniformly distributed between the minimum and maximum values to be encoded, and with a constant standard deviation. More precisely, the firing rate r_i (in Hz) of each input neuron i is given by $r_i = r_{max} \exp\left(-\frac{(m_i - z_i)^2}{2\sigma^2}\right)$, where $r_{max} = 200$ Hz, m_i is the value assigned to that neuron, z_i is the analog value to be encoded, and $\sigma = \frac{(m_{max} - m_{min})}{1000}$, m_{min} with m_{max} being the minimum and maximum values to be encoded.

LSNN setup and training schedule: The standard LSNN model was used, with 300 hidden neurons for the TN family of learning tasks, and 100 for the sinusoidal family. Of these, 40% were adaptive in all simulations. We used all-to-all connectivity between all neurons (regular and adaptive). The output of the LSNN was a linear readout that received as input the mean firing rate of each of the neurons per step i.e the number of spikes divided by 20 for the 20 ms time window that the step consists of.

The network training proceeded as follows: A new target function was randomly chosen for each **episode** of training, i.e., the parameters of the target function are chosen uniformly randomly from within the ranges above (depending on whether its a TN or sinusoidal). Each **episode** consisted of a sequence of 500 **steps**, each lasting for 20 ms. In each step, one training example from the current function to be learned was presented to the LSNN. In such a step, the inputs to the LSNN consisted of a randomly chosen vector $\mathbf{x}$ with its dimensionality d and range determined by the target function being used ($d = 2$ for TNs, $d = 1$ for sinusoidal target function). In addition, at each step, the LSNN also got the target value $C(\mathbf{x}')$ from the previous step, i.e., the value of the target calculated using the target function for the inputs given at the previous step (in the first step, $C(\mathbf{x}')$ is set to $\mathbf{0}$).

All the weights of the LSNN were updated using our variant of BPTT, once per **iteration**, where an **iteration** consists of a batch of 10 **episodes**, and the weight updates are accumulated across episodes in an iteration. The Adam Kingma and Ba, 2014 variant of BP was used with standard parameters and a learning rate of 0.001. The loss function for training was the mean squared error (MSE) of the LSNN predictions over an iteration (i.e. over all the steps in an episode, and over the entire batch of episodes in an iteration). In addition, a regularization term was used to maintain a firing rate of 20 Hz. Specifically, the regularization term R is defined as the mean squared difference between the average neuron firing rate in the LSNN and a target of 20 Hz. The total loss L was then given by $L = MSE + 30\,R$. In this way, we induce the LSNN to use sparse firing. We trained the LSNN for 5000 iterations in all cases.

Parameter values: The LSNN parameters were as follows: 5 ms neuronal refractory period, delays spread uniformly between $0 - 5$ ms, adaptation

time constants of the adaptive neurons spread uniformly between $1 - 1000$ ms, $\beta = 1.6$ for adaptive neurons (0 for regular neurons), membrane time constant $\tau = 20$ ms, 0.03 mV baseline threshold voltage. The dampening factor for training was $\gamma = 0.4$.

Analysis and comparison: The linear baseline was calculated using linear regression with L2 regularization with a regularization factor of 100 (determined using grid search), using the mean spiking trace of all the neurons. The mean spiking trace was calculated as follows: First the neuron traces were calculated using an exponential kernel with 20 ms width and a time constant of 20 ms. Then, for every step, the mean value of this trace was calculated to obtain the mean spiking trace. In Fig. 3.2C, for each episode consisting of 500 steps, the mean spiking trace from a random subset of 450 steps was used to train the linear regressor, and the mean spiking trace from remaining 50 steps was used to calculate the test error. The reported baseline is the mean of the test error over one batch of 10 episodes with error bars of one standard deviation. In Fig. 3.2E, for each episode, after every step k, the mean spiking traces from the first $k - 1$ steps were used to train the linear regressor, and the test error was calculated using the mean spiking trace for the kth step. The reported baseline is a mean of the test error over one batch of 10 episodes with error bars of one standard deviation.

For the case where neural networks defined the function family, the total test MSE was 0.0056 ± 0.0039 (linear baseline MSE was 0.0217 ± 0.0046). For the sinusoidal function family, the total test MSE was 0.3134 ± 0.2293 (linear baseline MSE was 1.4592 ± 1.2958).

Comparison with backprop: The comparison was done for the case where the LSNN is trained on the function family defined by target networks. A feed-forward (FF) network with 10 hidden neurons and 1 output was constructed. The input to this FF network were the analog values that were used to generate the spiking input and targets for the LSNN. Therefore the FF had 2 inputs, one for each of x_1 and x_2. The error reported in Fig 2F is the mean training error over 10 batches with error bars of one standard deviation.

The FF network was initialized with Xavier normal initialization Glorot

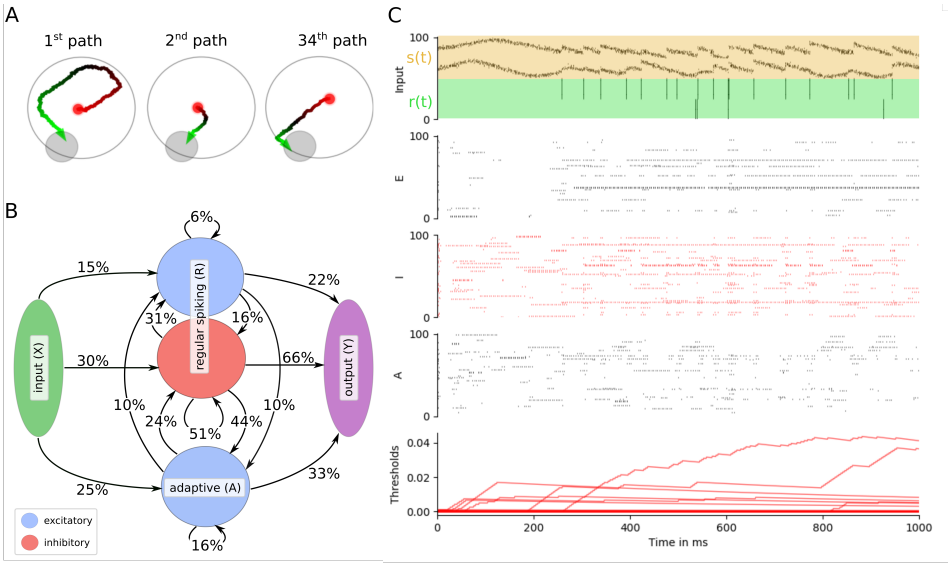

Fig. 3.4.: Meta-RL results for an LSNN. A Samples of paths after training. **B** Connectivity between sub-populations of the network after training. The global connectivity in the network was constrained to 20%. **C** The network dynamics that produced the behavior shown in A. Raster plots and thresholds are displayed as in Fig. 3.1D, only 1 second and 100 neurons are shown in each raster plots.

and Bengio, 2010 (which had the best performance, compared to Xavier uniform and plain uniform between $[-1, 1]$). Adam Kingma and Ba, 2014 with AMSGrad Reddi, Kale, and Kumar, 2018 was used with parameters $\eta = 10^{-1}, \beta_1 = 0.7, \beta_2 = 0.9, C = 10^{-5}$. These were the optimal parameters as determined by a grid search. Together with the Xavier normal initialization and the weight regularization parameter C, the training of the FF favoured small weights and biases.

LSNNs learn-to-learn from reward

Experimental setup:

Task family: An LSNN-based agent was trained on a family of navigation

tasks in a two dimensional circular arena. For all tasks, the arena is a circle with radius 1 and goals are smaller circles of radius 0.3 with centres uniformly distributed on the circle of radius 0.85. At the beginning of an episode and after the agent reaches a goal, the agent's position is set randomly with uniform probability within the arena. At every timestep, the agent chooses an action by generating a small velocity vector of Euclidean norm smaller or equal to $a_{scale} = 0.02$. When the agent reaches the goal, it receives a reward of 1. If the agent attempts to move outside the arena, the new position is given by the intersection of the velocity vector with the border and the agent receives a negative reward of -0.02.

Input encoding: Information of the current environmental state $s(t)$ and the reward $r(t)$ were provided to the LSNN at each time step t as follows: The state $s(t)$ is given by the x and y coordinate of the agent's position (see top of Fig. 3.4C). Each position coordinate $\xi(t) \in [-1, 1]$ is encoded by 40 neurons which spike according to a Gaussian population rate code defined as follows: a preferred coordinate value ξ_i, is assigned to each of the 40 neurons, where ξ_i's are evenly spaced between -1 and 1. The firing rate of neuron i is then given by $r_{max} \exp(-100(\xi_i - \xi)^2)$ where r_{max} is 500 Hz. The instantaneous reward $r(t)$ is encoded by two groups of 40 neurons (see green row at the top of Fig. 3.4C). All neuron in the first group spike in synchrony each time a reward of 1 is received (i.e., the goal was reached), and the second group spikes when a reward of -0.02 is received (i.e., the agent moved into a wall).

Output decoding: The output of the LSNN is provided by five readout neurons. Their membrane potentials $y_i(t)$ define the outputs of the LSNN. The action vector $\mathbf{a}(t) = (a_x(t), a_y(t))^T$ is sampled from the distribution π_θ which depends on the network parameters θ through the readouts $y_i(t)$ as follows: The coordinate $a_x(t)$ ($a_y(t)$) is sampled from a Gaussian distribution with mean $\mu_x = \tanh(y_1(t))$ ($\mu_y = \tanh(y_2(t))$) and variance $\phi_x = \sigma(y_3(t))$ ($\phi_y = \sigma(y_4(t))$). The velocity vector that updates the agent's position is then defined as $a_{scale}\, \mathbf{a}(t)$. If this velocity has a norm larger than a_{scale}, it is clipped to a norm of a_{scale}.

The last readout output $y_5(t)$ is used to predict the value function $V_\theta(t)$. It estimates the expected discounted sum of future rewards $R(t) = \sum_{t'>t} \eta^{t'-t} r(t')$, where $\eta = 0.99$ is the discount factor and $r(t')$ denotes the reward at time t'.

To enable the network to learn complex forms of exploration we introduced current noise in the neuron model in this task. At each time step, we added a small Gaussian noise with mean 0 and standard deviation $\frac{1}{R_m}\nu_j$ to the current I_j into neuron j. Here, ν_j is a network parameter initialized at 0.03 and optimized by BPTT alongside the network weights.

Network training: To train the network we used the Proximal Policy Optimization algorithm (PPO) Schulman et al., 2017. For each training iteration, K full episodes of T timesteps were generated with fixed parameters θ_{old} (here $K = 10$ and $T = 2000$). We write the clipped surrogate objective of PPO as $O^{PPO}(\theta_{old}, \theta, t, k)$ (this is defined under the notation L^{CLIP} in Schulman et al., 2017). The loss with respect to θ is then defined as follows:

$$\mathcal{L}(\theta) = -\frac{1}{KT}\sum_{k<K}\sum_{t<T} O^{PPO}(\theta_{old}, \theta, t, k) + \mu_v \left(R(t,k) - V_\theta(t,k)\right)^2 \quad (3.1)$$

$$-\mu_e H(\pi_\theta(k,t)) + \mu_{firing}\frac{1}{n}\sum_j ||\frac{1}{KT}\sum_{k,t} z_j(t,k) - f^0||^2, \quad (3.2)$$

where $H(\pi_\theta)$ is the entropy of the distribution π_θ, f^0 is a target firing rate of 10 Hz, and μ_v, μ_e, μ_{firing} are regularization hyper-parameters. Importantly probability distributions used in the definition of the loss $\mathcal{L}$ (i.e. the trajectories) are conditioned on the current noises, so that for the same noise and infinitely small parameter change from θ_{old} to θ the trajectories and the spike trains are the same. At each iteration this loss function $\mathcal{L}$ is then minimized with one step of the ADAM optimizer.

Parameter values: In this task the LSNN had 400 hidden units (200 excitatory neurons, 80 inhibitory neurons and 120 adaptive neurons with adaptation time constants $\tau_a = 1200$ ms) and the network was rewired with a fixed global connectivity of 20% Bellec, Kappel, et al., 2018. The membrane time constants were similarly sampled between 15 and 30 ms. The adaptation amplitude β was set to 1.7. The refractory period was set to 3 ms and delays were sampled uniformly between 1 and 10 ms. The regularization parameters μ_v, μ_e and μ_{firing} were respectively 1, 0.001, and 100. The parameter ϵ of the PPO algorithm was set to 0.2. The learning rate

was initialized to 0.01 and decayed by a factor 0.5 every 5000 iterations. We used the default parameters for ADAM, except for the parameter ϵ which we set to 10^{-5}.

4. Spike-frequency adaptation provides a long short-term memory to networks of spiking neurons

Contents

Abstract. For solving tasks such as recognizing a song, answering a question, or inverting a sequence of symbols, cortical microcircuits need to integrate and manipulate information that was dispersed over time during the preceding seconds. Creating biologically realistic models for the underlying computations, especially with spiking neurons and for behaviorally relevant integration time spans, is notoriously difficult. We examine the role of spike-frequency adaptation in such computations and find that it has a surprisingly large impact. The inclusion of this well known property of a substantial fraction of neurons in the neocortex — especially in higher areas of the human neocortex — moves the performance of spiking neural network models for computations on network inputs that are temporally dispersed from a fairly low level up to the performance level of the human brain.

Acknowledgments and author contributions. This chapter is based on the manuscripts

> Darjan Salaj*, Anand Subramoney*, Ceca Kraisnikovic*, Guillaume Bellec, Robert Legenstein, Wolfgang Maass (2020). "Spike frequency adaptation supports network computations on temporally dispersed information." *Submitted for publication. bioRXiv:10.1101/2020.05.11.081513*.

To this study, I contributed as first author together with AS and CK. The study was conceived by DS, AS, GB, WM. The experiments were designed by DS, AS, GB, RL, WM and were conducted by DS, AS, CK. The manuscript was written by DS, AS, CK, RL, GB, WM.

4.1. Introduction

Since brains have to operate in dynamically changing environments, neural networks of the brain need to be able to solve "temporal computing tasks" that require integration and manipulation of temporally dispersed information from continuous input streams on the behavioral time scale of seconds.

Models for neural networks of the brain have inherent difficulties in modeling such temporal computations on the time scale of seconds since spikes and postsynaptic potentials take place on the much shorter time scales of milliseconds and tens of milliseconds. Most work on models for temporal computing in neural networks of the brain has focused on networks of non-spiking neurons and temporal computing tasks where one or a few bits have to be stored in working memory without requiring computational operations other than storage and recall. We address the question of how biologically more realistic spike-based neural network models can solve a range of generic temporal computing tasks that require extraction of a fairly large amount of temporally dispersed information from continuous input streams. These tasks also require information stored in working memory to be continuously updated, as required, for example, for speech understanding or manipulations of sequences of symbols. So far, these tasks could not be solved by spiking neural network models.

In this chapter we examine the impact of SFA of neurons, see chapter 2, on temporal computing capabilities of spiking neural networks (SNNs).

We investigate here the role of SFA for a range of demanding temporal computing tasks, including the well-known 12AX task and manipulations of sequences of symbols that are also used for testing cognitive capabilities of brains. We find that SFA contributes strongly to temporal computing, in spite of the counter-intuitive fact that SFA reduces — rather than enhances — persistent firing. We introduce the negative imprinting principle as an explanation for this. Finally, we discuss the neural codes that emerge in neural networks with SFA for manipulations on sequences of symbols, and compare them with neural codes for corresponding tasks in the brain Barone and Joseph, 1989; Liu et al., 2019; Carpenter et al., 2018.

Spike-based neural networks are also of interest from the perspective of novel computing technology, since spike-based neuromorphic hardware promises to provide AI implementations with drastically reduced energy consumption in comparison with hardware implementations of ANNs Furber, Galluppi, et al., 2014; Davies et al., 2018. But because of deficiencies in the performance of recurrent networks of standard spiking neurons models for temporal computing tasks, spike-based hardware has so far not been able to reach the performance levels of ANNs for temporal computing tasks. Since it is

easy to add SFA to neurons in spike-based hardware, our results provide a new strategy for enabling spike-based hardware to reach competitive performance levels for temporal computing tasks.

In this chapter we consider recurrent networks of LIF neurons where some fraction of those is equipped with SFA, labeled Long short-term memory Spiking Neural Networks (LSNNs). Model details are described in Chapter 2.

4.2. SFA provides working memory simultaneously for many pieces of information, and yields powerful generalization capability

In order to elucidate the mechanisms by which SFA supports temporal computing capabilities of SNNs we first consider simpler tasks where the stored information does not have to be updated. Simultaneously we examine whether the working memory contribution of SFA scales up to working memory demands larger than in the commonly considered task where just a single bit has to be stored in working memory. Obviously, brains have a much larger working memory capacity insofar as they are able to store many salient bits of information from a previously seen image or movie or text. Furthermore, they are able to ignore irrelevant parts of complex sensory input streams. Some of these demanding aspects are captured in the task that is considered in Fig. 4.1. This task aims at capturing the need to remember a fair number of higher-level features that are needed for later recall of the content of an image in a higher visual area such as area IT (inferior temporal cortex). There, a 20-dimensional input stream of bits provides a continuous stream of input patterns, where each pattern can be visualized as 4×5 image, as indicated in the top row of Fig. 4.1. Occasionally, a pattern in the input stream is marked as being salient through simultaneous activation of a STORE command in a separate input channel, corresponding for example to an attentional signal from a higher brain area. The task is to reproduce during a RECALL command the pattern that had been presented during the most recent STORE command. Delays between STORE and RECALL ranged from 200 to 1600 ms. 20 binary values were simultaneously extracted as

network outputs during RECALL by rounding the output values of 20 linear readout neurons. We found that an LSNN consisting of 500 neurons with SFA, whose adaptive firing threshold had a time constant of $\tau_a = 800\,\text{ms}$, was able to solve this task with an accuracy above 99%. SFA was essential for this behavior, because the recall performance of a recurrent network of LIF neurons without SFA, trained in exactly the same way, stayed at chance level (see Methods).

4.2.1. Generalization of SFA-enhanced temporal computations to unseen inputs.

Humans can retain previously unseen stimuli in short-term memory. In order to probe whether LSNNs are capable of such generalization, we made sure that none of the patterns shown during testing had occurred during training, and in fact, had a Hamming distance of at least 5 bits to all training patterns. The resulting recall performance of the LSNN was 99.09%, i.e., 99.09% of the stored feature vectors were perfectly reproduced during recall. Note that in contrast to most models for working memory, we require that this SNN is able to store content other than what was used when the values of its synaptic weights were determined. A sample segment of a test trial is shown in Fig. 4.1, with the activity of input neurons at the top and the activation of readout neurons at the bottom.

4.2.2. Negative imprinting principle.

Fig. 4.1 elucidates how neurons with SFA support temporal computing: The 3rd to last row shows the temporal dynamics of 25 selected neurons with SFA. One fraction of these neurons, with firing thresholds drawn in blue (second to last row), fires strongly during a STORE command. A complementary set of neurons with SFA, with trajectories of firing thresholds drawn in red, fires during the subsequent RECALL command. We propose that the non-firing of the neurons with thresholds drawn in blue during the RECALL command signals to the readout neurons which of the 20 bits were active during STORE, and thereby enables them to reproduce this 20-bit pattern. We refer to this coding method as negative imprinting principle.

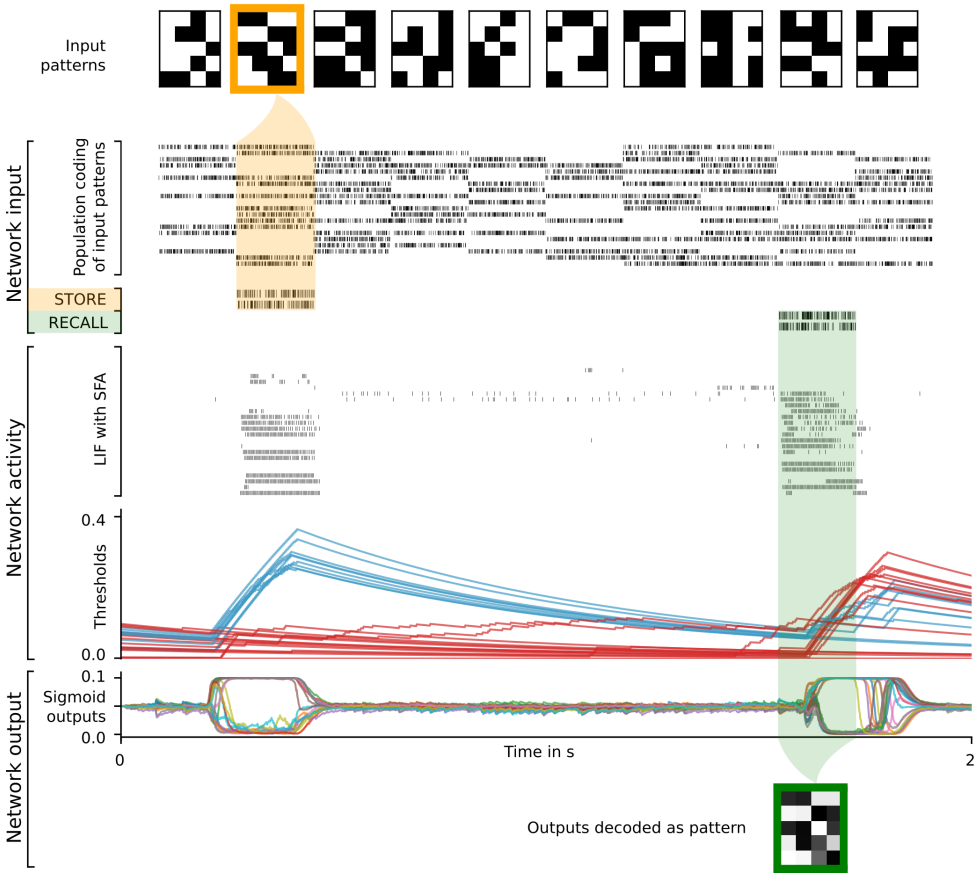

Fig. 4.1.: Sample trial of 20-dimensional STORE-RECALL task. Rows top to bottom: Stream of randomly drawn 20-dimensional input patterns, represented by the firing activity of 20 populations of input neurons (a subsample is shown), firing activity of two additional populations of input neurons for the STORE and RECALL commands, firing activity of 25 sample LIF neurons with SFA in the LSNN (we first ordered all neurons with regard to the variance of their dynamic firing thresholds, and then picked every 20th), the temporal evolution of the firing thresholds of these 25 neurons, traces of the activation of 20 sigmoidal readout neurons, and their average value during the 200 ms time window of the RECALL command represented by grey values. During the RECALL command (green shading) the network successfully reproduced the pattern that had been given as input during the preceding STORE command (yellow shading). Coloring of the threshold traces in blue or red was done after visual inspection to highlight the emergent two disjoint populations of neurons.

It is of general interest insofar as one often focuses on the information that is transmitted by spikes, and forget that non-spiking can also transmit information — in fact in a much more energy-efficient manner.

Interestingly, the firing activity of the network was rather low during the delay between STORE and RECALL. Furthermore, a powerful classifier — a Support Vector Machine (SVM) trained on the network activity during the delay — was not able to decode the stored feature vector from the firing activity (the decoding accuracy during the delay was 4.38%, as opposed to 100% decoding accuracy during RECALL; see Methods). Hence the type of working memory that the LSNN exhibits during the STORE-RECALL task is related to the activity-silent form of working memory in the human brain that had been examined in the experiments of Wolff et al., 2017.

If the content of working memory is encoded by an attractor of the network dynamics, one would expect that the neural code for the content of the working memory changes little between encoding and a subsequent network reactivation. In contrast to this principle, it had been shown for the human brain in Wolff et al., 2017 that the representation of working memory content changes significantly between memory encoding and subsequent network reactivation during the delay by an "impulse stimulus": A classifier trained on the network activity during encoding was not able to classify the memory content during a network reactivation in the delay, and vice versa. This experimental result from the human brain is consistent with the negative imprinting principle. We also tested directly whether such change of neural codes occurs in our model for the STORE-RECALL task. We found that a classifier trained for decoding the content of working memory during STORE was not able to decode this content during RECALL, and vice versa (see Methods). Hence our model is in this regard consistent with the experimental data of Wolff et al., 2017.

4.2.3. No precise alignment between time constants of SFA and working memory duration is needed.

Experimental data from the Allen Institute database suggest that different neurons exhibit a diversity of SFA properties. We show that correspondingly a diversity of time constants of SFA in different neurons provides high

performance for temporal computing. We consider for simplicity the one-dimensional version of the task of Fig. 4.1, where just a single bit needs to be stored in working memory between STORE and RECALL commands. The expected delay between STORE and RECALL (see the top row of Table 4.1) scales the working memory time span that is required to solve this task. In the upper four rows of Table 4.1, a different fixed time constant for SFA neurons in the network is chosen. In the last two rows a power-law distribution of these time constants for SFA, as well as a uniform distribution is considered. One sees that the resulting diversity of time constants for SFA yields about the same performance as a fixed choice of the time constant that is aligned with the required memory span of the task. However, a much larger time constant (see the row with $\tau_a = 8$ s in the column with an expected memory span of 200 ms or 2 s for the task) or a substantially smaller time constant (see the row with $\tau_a = 2$ s in the column with an expected memory span of 8 s) tends to work well.

Expected delay between STORE and RECALL	200 ms	2 s	4 s	8 s	16 s
without SFA ($\tau_a = 0$ ms)	96.7	51	50	49	51
$\tau_a = 200$ ms	99.92	73.6	58	51	51
$\tau_a = 2$ s	99.0	99.6	98.8	92.2	75.2
$\tau_a = 4$ s	99.1	99.7	99.7	97.8	90.5
$\tau_a = 8$ s	99.6	99.8	99.7	97.7	97.1
τ_a power-law dist. in [0, 8] s	99.6	99.7	98.4	96.3	83.6
τ_a uniform dist. in [0, 8] s	96.2	99.9	98.6	92.1	92.6

Table 4.1.: Recall accuracy (in %) of network models with different time constants of SFA (rows) for variants of the STORE-RECALL task with different required memory time spans (columns). Good task performance does not require good alignment of SFA time constants with the required time span for working memory. An SNN consisting of 60 LIF neurons with SFA was trained for many different choices of SFA time constants for variations of the one-dimensional STORE-RECALL task with different required time spans for working memory. A network of 60 LIF neurons without SFA trained under the same parameters did not improve beyond chance level ($\sim$ 50% accuracy), except for the task instance with an expected delay of 200 ms where the LIF network reached 96.7% accuracy (see top row).

4.3. SFA improves the performance of SNNs for common benchmark tasks that require computational operations on temporally dispersed information

An efficient model of temporal computing must not only be able to robustly store and recall information, but also actively manipulate the memory content. To investigate the impact of SFA in this context, we trained SNNs with and without SFA on a standard benchmark task for time series classification: the pixel-wise sequential MNIST (sMNIST) pattern classification task. In this variant of the well-known handwritten digit recognition data set MNIST, the pixels of each sample of a handwritten digit are temporally dispersed: they are given to the network one at a time, as they arise from a row-wise scanning pattern. This temporal computing task can apparently not be solved by just storing some bits in a working memory and recalling them later. Instead, evidence that speaks for or against the hypothesis that a particular digit is represented by the currently received time series must be updated continuously. This task also requires very good generalization capability, since the pixel sequences for different handwriting styles of the same digit may vary widely, and the network is tested on samples that were not used during optimization of the weights. For details see Methods and Supplement.

An LSNN was able to solve this task with a test accuracy of 93.7%, whereas an SNN without SFA was only able to reach an accuracy of 51.8%. This demonstrates the significant impact of SFA on the ability of SNNs to actively manipulate information retained from previous inputs. See section 2 of the Supplement for more results with sparse connectivity, enforcement of Dale's Law and comparison to ANNs.

We also compared the performance of LSNNs and SNNs without SFA on the keyword spotting task Google Speech Commands Dataset Warden, 2018 (v0.02). To solve this task the network needs to learn to correctly disambiguate between audio recordings of silence, unknown words, or one of ten keywords. On this task, the performance of SNNs increases with the inclusion of SFA (from 89.04% to 91.21%) and approaches the state-of-

the-art artificial recurrent model (93.18%), see section 3 of the Suppl. and Table S1.

Finally, we tested the performance of LSNNs on the delayed-memory XOR task, because this task has previously already been used as benchmark tasks for SNNs in Huh and Sejnowski, 2018. In this task, the network is required to compute the exclusive-or operation on the history of input pulses when prompted by a go-cue signal. Across 10 different runs, an LSNN solved the task with $95.19 \pm 0.014\%$ accuracy, whereas the SNN without SFA converged at a much lower accuracy of $61.30 \pm 0.029\%$, see section 4 of the Suppl. and Fig. S3.

The good performance of LSNNs on all three tasks demonstrates the powerful temporal computation capability of these networks.

4.4. SFA supports demanding cognitive computations on sequences with dynamically changing rules

The 12AX task — which can be viewed as a simplified version of the Wisconsin Card Sorting task E. A. Berg, 1948 — tests the capability of subjects to apply dynamically changing rules for detecting specific subsequences in a long sequence of symbols as target sequences, and to ignore currently irrelevant inputs O'Reilly and Frank, 2006; MacDonald III, 2008. It also probes the capability to maintain and update a hierarchical working memory, since the currently active rule — the context — stays valid for a longer period of time, and governs what other symbols should be stored in working memory.

More precisely, after processing any symbol in the sequence, the network should output "R" if this symbol terminates a context-dependent target sequence and "L" otherwise. The current target sequence depends on the current context, which is defined through the symbols "1" and "2". If the most recently received digit was a "1", the subject should output "R" only when it encounters a symbol "X" that terminates a subsequence A...X. This occurs, for example, for the 7th symbol in the trial shown in Fig. 4.2. In case that the most recent input digit was a "2", the subject should instead respond

"R" only after the symbol "Y" in a subsequent subsequence B...Y (see the 20th symbol in Fig. 4.2). In addition, the processed sequence contains letters "C" and "Z" that are irrelevant and serve as distractors. This task requires a hierarchical working memory because the most recently occurring digit determines whether subsequent occurrences of "A" or "B" should be placed into working memory. Note also that neither the content of the higher-level working memory — the digit — nor the content of the lower-level working memory — the letter A or B — are simply recalled. Instead, they affect the target outputs of the network in a more indirect way. Furthermore, the higher-level processing rule affects what is to be remembered at the lower level.

A simpler version of this task, where X and Y were relevant only if they directly followed A or B respectively, and where fewer irrelevant letters occurred in the input, was solved in O'Reilly and Frank, 2006; Martinolli, Gerstner, and Gilra, 2018; Kruijne et al., 2020 through biologically inspired artificial neural network models that were endowed with special working memory modules. Note that for this simpler version no lower-order working memory is needed, because one just has to wait for an immediate transition from A to X in the input sequence, or for an immediate transition from B to Y. But neither the simpler nor the more complex version, that is considered here, of the 12AX task has previously been solved by a network of spiking neurons.

In the version of the task that we consider, the distractor symbols between relevant symbols occur rather frequently. Hence robust maintenance of relevant symbols in the hierarchical working memory becomes crucial, because time spans between relevant symbols become longer, and hence the task is more demanding — especially for a neural network implementation.

Overall, the network received during each trial (episode) sequences of 90 symbols from the set {1, 2, A, B, C, X, Y, Z}, with repetitions as described in Methods. See the top of Fig. 4.2 for an example (the context-relevant symbols are marked in bold for visual ease).

We show in Fig. 4.2 that a generic LSNN can solve this quite demanding version of the 12AX task. The network consisted of 200 recurrently connected spiking neurons (100 with and 100 without SFA), with all-to-all connections between them. After training, for new symbol sequences that had never

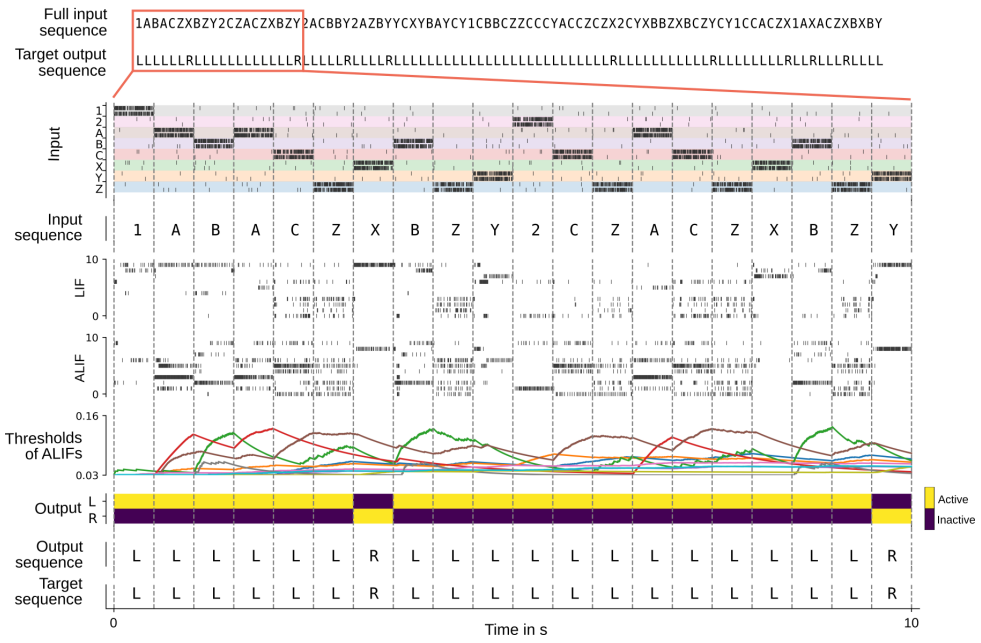

Fig. 4.2.: Solving the 12AX task by a network of spiking neurons with SFA. A sample trial of the trained network is shown. From top to bottom: Full input and target output sequence for a trial, consisting of 90 symbols each, blow-up for a subsequence of the input symbols, firing activity of 10 sample LIF neurons without and 10 sample neurons with SFA from the network, time course of the firing thresholds of these neurons with SFA, activation of the two readout neurons, the resulting sequence of output symbols which the network produced, and the target output sequence.

occurred during training, the network produced an output string with all correct symbols in 97.79% of episodes. In contrast, a recurrent SNN with the same architecture but no neurons with SFA could achieve only 0.39% fully correct output strings (not shown).

Surprisingly, it was not necessary to create a special network architecture for the two levels of working memory that our more complex version of the 12AX task requires: A near perfectly performing network emerged from training a generic LSNN. This shows that neurons with SFA enable generic recurrent networks of spiking neurons to solve demanding cognitive tasks involving dynamically changing rules and two levels of working memory.

4.5. SFA enables SNNs to carry out complex operations on sequences of symbols

Learning to carry out operations on sequences of symbols in such a way that they generalize to new sequences is a fundamental capability of the human brain, but a generic difficulty for neural networks Marcus, 2003. Not only humans, but also non-human primates are able to carry out operations on sequences of items, and numerous neural recordings — starting with Barone and Joseph, 1989 up to recent results such as Carpenter et al., 2018; Liu et al., 2019 — provide information about the neural codes for sequences that accompany such operations in the brain. One fundamental question is how serial order of items is encoded in working memory. Behind this is the even more basic question of how transient structural information — the serial position of an item — is combined in the brain with content information about the identity of the item Lashley, 1951. Obviously, this question also lies at the heart of open questions about the interplay between neural codes for syntax and semantics that enable language understanding in the human brain. The experimental data both of Barone and Joseph, 1989 and Liu et al., 2019 suggest that the brain uses a factorial code where position and identity of an item in a sequence are encoded separately by some neurons, thereby facilitating flexible generalization of learned experience to new sequences.

We show here that LSNNs can be trained to carry out complex operations on sequences, are able to generalize such capabilities to new sequences, and

produce spiking activity and neural codes that can be compared with neural recordings from the brain. In particular, they also produce factorial codes, where separate neurons encode the position and identity of a symbol in a sequence. One basic operation on sequences of symbols is remembering and reproducing a given sequence Liu et al., 2019. This task had been proposed by Marcus, 2003 to be a symbolic computation task that is fundamental for symbol processing capabilities of the human brain. But non-human primates can also learn simpler versions of this task, and hence one had been able to analyze how neurons in the brain encode the position and identity of symbols in a sequence Barone and Joseph, 1989; Carpenter et al., 2018. A more complex operation that can also be carried out by the human brain is the reversal of a sequence Marcus, 2003; Liu et al., 2019. We show that an LSNN can carry out both of these operations, and is able to apply them to new sequences of symbols that did not occur during the training of the network.

We trained an LSNN consisting of 320 recurrently connected LIF neurons (192 with and 128 without SFA) to carry out these two operations on sequences of 5 symbols from a repertoire of 31 symbols. After training, the LSNN was able to apply duplication and reversal to new sequences also, achieving a success rate of 95.88% (average over 5 different network initializations) for unseen sequences. The "success rate" was defined as the fraction of test episodes (trials) where the full output sequence was generated correctly. Sample episodes of the trained LSNN are shown in Fig. 4.3A. For comparison, we also trained a LIF network without SFA in exactly the same way with the same number of neurons. It achieved a performance of 0.0%.

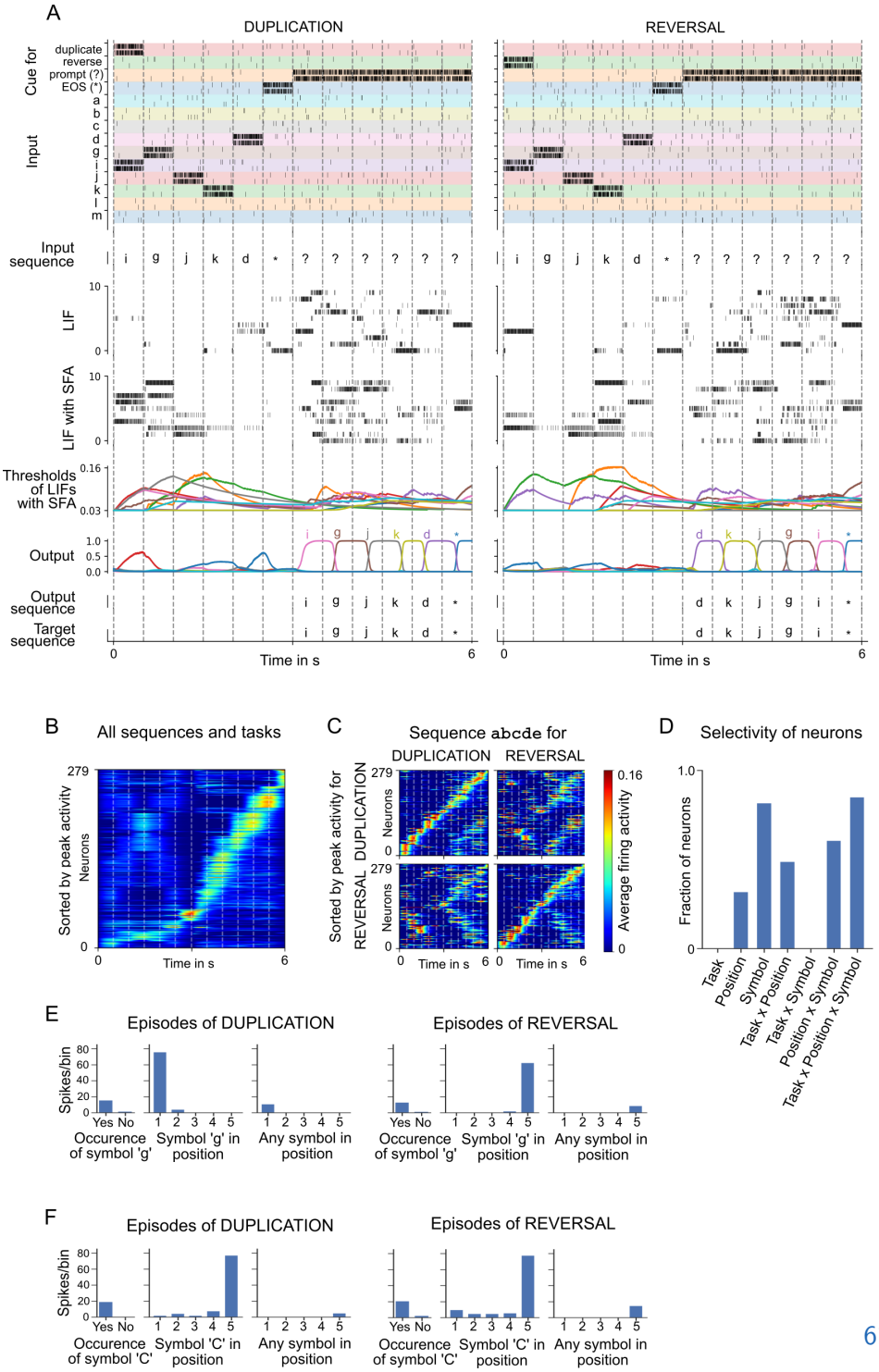

Fig. 4.3.: (Caption on the next page.)

Fig. 4.3.: Analysis of an LSNN trained to carry out operations on sequences. (**A**) Two sample episodes where the network carried out sequence duplication (left) and reversal (right). Top to bottom: Spike inputs to the network (subset), sequence of symbols they encode, spike activity of 10 sample LIF neurons (without and with SFA) in the LSNN, firing threshold dynamics for these 10 LIF neurons with SFA, activation of linear readout neurons, output sequence produced by applying argmax to them, target output sequence. (**B-F**) Emergent neural coding of 279 neurons in the LSNN, and Peri-Condition Time Histogram (PCTH) plots of two sample neurons. Neurons are sorted by time of peak activity. (**B**) A substantial number of neurons were sensitive to the overall timing of the tasks, especially for the second half of trials when the output sequence is produced. (**C**) Neurons separately sorted for duplication episodes (left column) and reversal episodes (right column). Many neurons responded to input symbols according to their serial position, but differently for different tasks. (**D**) Histogram of neurons categorized according to conditions with statistically significant effect (3-way ANOVA). Firing activity of a sample neuron that fired primarily when: (**E**) the symbol "g" was to be written at the beginning of the output sequence. The activity of this neuron depended on the task context during the input period; (**F**) the symbol "C" occurred in position 5 in the input, irrespective of the task context.

4.5.1. A diversity of neural codes in LSNNs.

Emergent coding properties of neurons in the LSNN are analyzed in Fig. 4.3B-F. Neurons are sorted in Fig. 4.3B,C according to the time of their peak activity (averaged over 1000 episodes), like in Christopher D Harvey, Coen, and Tank, 2012. A number of network neurons (about one-third) participate in sequential firing activity independent of the type of task and the symbols involved (Fig. 4.3B). Instead, these neurons have learned to abstract the overall timing of the tasks. This kind of activity is reminiscent of the neural activity relative to the start of a trial that was recorded in rodents after they had learned to solve tasks that had a similar duration Tsao et al., 2018.

The time of peak activity of other neurons depended on the task and the concrete content, see Fig. 4.3C. Interestingly enough, these neurons change their activation order already during the loading of the input sequence in dependence of the task (duplication or reversal). Using 3-way ANOVA, we were able to categorize each neuron as selective to a specific condition or a non-linear combination of conditions based on the effect size ω^2. Each

neuron could belong to more than one category if the effect size was above the threshold of 0.14 (as suggested by Field, 2013). Similar to recordings from the brain Carpenter et al., 2018, a diversity of neural codes emerged that encode one or several of the variables: symbol identity, serial position in the sequence, and type of task. In other words, a large fraction of neurons were mixed-selective, i.e. selective to non-linear combinations of all three variables. Peri-Condition Time Histogram (PCTH) plots of two sample neurons are shown in Fig. 4.3E,F: One neuron is selective to symbol "g" but at different positions depending on task context; The other neuron is selective to symbol "C" occurring at position 5 in the input, independent of task context. Thus one sees that a realization of this task by an LSNN, which was previously not available, provides rich opportunities for a comparison of emergent spike codes in the model and neuronal recordings from the brain.

4.6. Discussion

Brains are able to carry out complex computations on temporally dispersed information, for example, on visual inputs streams, or on sequences of symbols. Previous neural network solutions for similarly demanding temporal computing tasks were based on artificial Long Short-Term Memory (LSTM) units. These LSTM units are commonly used in machine learning, but they cannot readily be mapped to units of neural networks in the brain. Hence it had remained an open problem how spiking neural networks of the brain could carry out such computations. We have shown that by adding to the standard model for SNNs one important feature of a substantial fraction of neurons in the neocortex, SFA, SNNs become able to solve such demanding temporal computing tasks.

In particular, we have shown that LSNNs (SNNs that contain neurons with SFA) are able to carry out the 12AX task (Fig. 4.2). This task has been used to test the capability of human subjects to make online decisions while receiving sequences of symbols. These online decisions have to be carried out according to rules that are also encoded — and occasionally changed — by symbols that occur in the same sequence of symbols.

We have shown in Fig. 4.3 that LSNNs can solve another cognitively demanding task on temporally dispersed information: reproducing a previously received symbol sequence, or inverting its order. Since our model consists of spiking neurons, one can directly compare emergent neural codes for the identity and sequence positions of previously encountered symbols with recordings from neurons in the monkey brain for corresponding behavioral tasks. Similarly as in the recordings of Barone and Joseph, 1989; Carpenter et al., 2018, a diversity of neural codes emerge in the model, where neurons encode one or several of the relevant variables – symbol identity, serial position of a symbol, and type of task.

We have also elucidated the computational principle that enables neurons with SFA to support temporal computing tasks. We had considered for that purpose in Fig. 4.1 a simpler temporal computing task where no intermediate updating of stored information was required: All preceding information was provided at a single time point A, and had to be recalled at a well-specified later time point B. We found there that neurons with SFA store information through negative imprinting, i.e., neurons that fire more during time point A fire less at time point B. Hence they encode past information in an activity-silent manner. Such activity-silent form of working memory has been found in the human brain for content that is currently not in the focus of attention Wolff et al., 2017. A prediction of the negative imprinting principle is that decoders that are trained to decode information at time point A are not able to decode this information during a network reactivation at some intermediate time point C between time points A and B, and vice versa (see subsection "Decoding memory from the network activity" in Methods). This prediction had been verified for the human brain in the experiments of Wolff et al., 2017. It is actually well-known that negative imprinting is used by the brain for a particular type of long-term memory called recognition memory: Familiarity of an object is encoded through the reduced firing of a large fraction of neurons in the perirhinal cortex and adjacent areas, see Winters, Saksida, and Bussey, 2008 for a review.

Our model makes a number of concrete suggestions for further experiments. It suggests that a refined decoder that takes negative imprinting into account is able to elucidate the transformation of stored information between time points A (encoding) and an intermediate time point C (network reactivation) in the experiment of Wolff et al., 2017.

Furthermore, the strong role of SFA for temporal computing tasks that we found predicts that SFA is more common in those brain areas that play a key role in temporal computing tasks. At the same time, our detailed analysis of the required alignment between the time scale of SFA and the time scale of working memory duration can be rather loose. Even a random distribution of time constants for SFA works well. Previous experiments have already reported history-dependence of neural firing for up to 20 s Pozzorini, Naud, et al., 2013; Pozzorini, Mensi, et al., 2015, but a more systematic analysis is needed.

Finally, our results raise the question of the extent to which the distribution of time scales of neurons with SFA in a cortical area is related to the intrinsic time scale of that cortical area as measured via intrinsic fluctuations of spiking activity Murray et al., 2014; Wasmuht et al., 2018. Are neurons with SFA essential for defining the intrinsic time scale of an area, or can one find cases where both time scales diverge? We tested the relation between time constants of SFA and the intrinsic time scale of neurons for the case of the STORE-RECALL task (see section 1 of the Suppl. and Fig. S1). We found that the time constants of neurons with SFA had little impact on their intrinsic time scale for this task. We conjecture that the network input has a stronger impact on the intrinsic time scales of neurons for this task.

Altogether, we have shown that a well-known feature of a substantial fraction of neurons in the neocortex — SFA — provides an important new facet for our understanding of computations in SNNs: It enables SNNs to integrate temporally dispersed information seamlessly into ongoing network computations. This paves the way for reaching a key-goal of modeling — to combine detailed experimental data from neurophysiology on the level of neurons and synapses with the brain-like high computational performance of the network.

4.7. Methods

4.7.1. Network models

LIF neuron model, SFA, and training method are described in Chapter 2.

Weight initialization. Initial input and recurrent weights were drawn from a Gaussian distribution $W_{ji} \sim \frac{w_0}{\sqrt{n_{in}}}\mathcal{N}(0,1)$, where n_{in} is the number of afferent neurons and $\mathcal{N}(0,1)$ is the zero-mean unit-variance Gaussian distribution and $w_0 = \frac{1\,\text{Volt}}{R_m}\delta t$ is a normalization constant.

4.7.2. Tasks

20-dimensional STORE-RECALL task. The input to the network consisted of commands STORE and RECALL, and 20 bits which were represented by subpopulations of spiking input neurons. STORE and RECALL commands were represented by 4 neurons each. The 20 bits were represented by population coding where each bit was assigned 4 input neurons (2 for value zero, and 2 for value one). When a subpopulation was active, it would exhibit a Poisson firing with a frequency of 400 Hz. Each input sequence consisted of 10 steps (200 ms each) where a different population encoded bit string was shown during every step. Only during the RECALL period, the input populations, representing the 20 bits, were silent. At every step, the STORE or the RECALL populations were activated interchangeably with probability 0.2 which resulted in the distribution of delays between the STORE-RECALL pairs in the range [200, 1600] ms.

To measure the generalization capability of a trained network, we first generated a test set dictionary of 20 unique feature vectors (random bit strings of length 20) that had at least a Hamming distance of 5 bits among each other. For every training batch, a new dictionary of 40 random bit strings (of length 20) was generated where each string had a Hamming distance of at least 5 bits from any of the bit string in the test set dictionary. This way we ensured that, during training, the network never encountered any bit string similar to one from the test set.

Networks were trained for 4000 iterations with a batch size of 256 and stopped if the error on the training batch was below 1%. We used Adam optimizer Kingma and Ba, 2014 with default parameters and initial learning rate of 0.01 which is decayed every 200 iterations by a factor of 0.8. We also used learning rate ramping, which, for the first 200 iterations, monotonically

increased the learning rate from 0.00001 to 0.01. To avoid unrealistically high firing rates, the loss function contained a regularization term (scaled with coefficient 0.001) that minimizes the squared difference of the average firing rate of individual neurons from a target firing rate of 10 Hz. To improve convergence, we also included an entropy component to the loss (scaled with coefficient 0.3) which was computed as the mean of the entropies of the outputs of the sigmoid neurons. The test performance was computed as average over 512 random input sequences.

We trained LSNNs and SNNs without SFA, consisting of 500 recurrently connected neurons. The membrane time constant was $\tau_m = 20$ ms and the refractory period was 3 ms. Adaptation parameters were $\beta = 4$ mV and $\tau_a = 800$ ms with baseline threshold voltage 10 mV. The synaptic delay was 1 ms. The input to the sigmoidal readout neurons were the neuron traces that were calculated by passing all the network spikes through a low-pass filter with a time constant of 20 ms.

We ran 5 training runs with different random seeds (initializations) for both LSNNs and SNNs without SFA. All runs of the LSNN network converged after $\sim$ 3600 iterations to a training error below 1%. At that point we measured the accuracy on 512 test sequences generated using the previously unseen test bit strings which resulted in test accuracy of 99.09% with a standard deviation of 0.17%. The LIF network was not able to solve the task in any of the runs (all runs resulted in 0% training and test accuracy with zero standard deviation). On the level of individual feature recall accuracy, the best one out of 5 training runs of the LIF network was able to achieve 49% accuracy which is the chance level since individual features are binary bits. In contrast, all LSNNs runs had individual feature level accuracy of above 99.99%.

One-dimensional STORE-RECALL task. The input to the network consisted of 40 input neurons: 10 for STORE, 10 for RECALL, and 20 for population coding of a binary feature. Whenever a subpopulation was active, it would exhibit a Poisson firing with a frequency of 50 Hz. The input sequences of experiments with the expected delay of 2, 4, 8, and 16 s were constructed as a sequence of 20, 40, 80, 120 steps respectively, with each step

lasting for 200 ms. For the experiment with expected delay of 200 ms, the input sequence consisted of 12 steps of 50 ms.

Networks were trained for 400 iterations with a batch size of 64.We used Adam optimizer with default parameters and initial learning rate of 0.01 which was decayed every 100 iterations by a factor of 0.3. The same firing rate regularization term was added to the loss as in the 20-dimensional STORE-RECALL setup (see above). The test performance was computed as the batch average over 2048 random input sequences.

Networks consisted of 60 recurrently connected neurons. The membrane time constant was $\tau_m = 20$ ms and the refractory period was 3 ms. Adaptation parameters were $\beta = 1$ mV with baseline threshold voltage 10 mV. Table 4.1 defines the adaptation time constants and expected delay of the experiments in that section. The synaptic delay was 1 ms. The same sigmoidal readout neuron setup was used as in the 20-dimensional STORE-RECALL setup (see above).

Decoding memory from the network activity. We trained a Support Vector Machine (SVM) to classify the stored memory content from the network spiking activity in the step before the RECALL (200 ms before the start of RECALL command). We performed a cross-validated grid-search to find the best hyperparameters for the SVM which included kernel type {linear, polynomial, RBF} and penalty parameter C of the error term {0.1, 1, 10, 100, 1000}. We trained SVMs on test batches of the 5 different training runs of 20-dimensional STORE-RECALL task. SVMs trained on the period preceding the RECALL command of a test batch achieved an average of 4.38% accuracy with a standard deviation of 1.29%. In contrast, SVMs trained on a period during the RECALL command achieved an accuracy of 100%. This demonstrates that the memory stored in the network is not decodable from the network firing activity before the RECALL input command.

Additionally, analogous to the experiments of Wolff et al., 2017, we trained SVMs on network activity during the encoding (STORE) period and evaluated them on the network activity during reactivation (RECALL), and vice versa. In both scenarios, the classifiers were not able to classify the memory content of the evaluation period (0.0% accuracy).

sMNIST task. The input consisted of sequences of 784 pixel values created by unrolling the handwritten digits of the MNIST dataset, one pixel after the other in a scanline manner as indicated in Fig. S2A. We used 1 ms presentation time for each pixel gray value. Each of the 80 input neurons was associated with a particular threshold for the grey value, and this input neuron fired whenever the grey value crossed its threshold in the transition from the previous to the current pixel.

Networks were trained for 36,000 iterations using the Adam optimizer with batch size 256. The initial learning rate was 0.01 and every 2500 iterations the learning rate was decayed by a factor of 0.8. The same firing rate regularization term was added to the loss as in the STORE-RECALL setup (see above) but with the scaling coefficient of 0.1.

Networks consisted of 220 neurons. The network with SFA had 100 neurons out of 220 with SFA and the rest without. The neurons had a membrane time constant of $\tau_m = 20\,\text{ms}$, a baseline threshold of $v_{\text{th}} = 10\,\text{mV}$, and a refractory period of 5 ms. LIF neurons with SFA had the adaptation time constant $\tau_a = 700\,\text{ms}$ with adaptation strength $\beta = 1.8\,\text{mV}$. The synaptic delay was 1 ms. The output of the network was produced by the softmax of 10 linear output neurons that received the low-pass filtered version of the spikes from all neurons in the network, as shown in the bottom row of Fig. S2B. The low-pass filter had a time constant of 20 ms. For training the network to classify into one of the ten classes we used cross-entropy loss computed between the labels and the softmax of output neurons.

The 12AX task. The input for each training and testing episode consisted of a sequence of 90 symbols from the set {1,2,A,B,C,X,Y,Z}. A single episode could contain multiple occurrences of digits 1 or 2 (up to 23), each time changing the target sequence (A...X or B...Y) after which the network was supposed to output R. Each digit could be followed by up to 26 letters before the next digit appeared. More precisely, the following regular expression describes the string that was produced: `[12][ABCXYZ]{1,10}((A[CZ]{0,6}X|B[CZ]{0,6}Y)|([ABC][XYZ])){1,2}`. Each choice in this regular expression was made randomly.

The network received spike trains from the input population of spiking neurons, producing Poisson spike trains. Possible input symbols were encoded

using one-hot coding. Each input symbol was signaled through a high firing rate of a separate subset of 5 input neurons for 500 ms. The output consisted of two readouts, one for L, one for the R response. During each 500 ms time window, the input to these readouts was the average activity of neurons in the network during that time window. The final output symbol was based on which of the two readouts had the maximum value.

The neurons had a membrane time constant of $\tau_m = 20$ ms, a baseline threshold $v_{\text{th}} = 30$ mV, a refractory period of 5 ms, and synaptic delays of 1 ms. LIF neurons with SFA had an adaptation strength of $\beta = 1.7$ mV, and adaptation time constants were chosen uniformly from $[1, 13500]$ ms.

A cross-entropy loss function was used along with a regularization term (scaled with coefficient 15) that minimizes the squared difference of average firing rate between individual neurons and a target firing rate of 10 Hz. The network was trained using the Adam optimizer for 10,000 iterations with a batch size of 20 episodes and a fixed learning rate of 0.001. An episode consisted of 90 steps, with between 4 to 23 tasks generated according to the task generation procedure described previously. We trained the network with *BPTT* using 5 different network initializations, which resulted in an average test success rate of 97.79% with a standard deviation of 0.42%.

Symbolic computation on strings of symbols. The input to the network consisted of 35 symbols: 31 symbols represented symbols from the English alphabet {a, b, c, d, ... x, y, z, A, B, C, D, E}, one symbol was for "end-of-string" (EOS) '*', one for cue for the output prompt '?', and two symbols to denote whether the task command was duplication or reversal. Each of the altogether 35 input symbols were given to the network in the form of higher firing activity of a dedicated population of 5 input neurons outside of the network ("one-hot encoding"). This population of input neurons fired at a "high" rate (200 Hz) to encode 1, and at a "low" rate (2 Hz) otherwise. The network output was produced by linear readouts (one per potential output symbol, each with a low pass filter with a time constant of 250 ms) that received spikes from neurons in the network, see the row "Output" in Fig. 4.3A. The final output symbol was selected using the readout which had the maximum value at the end of each 500 ms time window (a softmax

instead of the hard argmax was used during training), mimicking winner-take-all computations in neural circuits of the brain Chettih and C. D. Harvey, 2019 in a qualitative manner.

The network was trained to minimize the cross-entropy error between the softmax applied to the output layer and targets. The loss function contained a regularization term (scaled with coefficient 5) that minimizes the squared difference of average firing rate between individual neurons and a target firing rate of 20 Hz.

The training was performed for 50,000 iterations, with a batch size of 50 episodes. We used Adam optimizer with default parameters and a fixed learning rate of 0.001. Each symbol was presented to the network for a duration of 500 ms. The primary metric we used for measuring the performance of the network was success rate, which was defined as the percentage of episodes where the network produced the full correct output for a given string i.e. all the output symbols in the episode had to be correct. The network was tested on 50,000 previously unseen strings.

The network consisted of 192 LIF neurons with SFA and 128 LIF neurons without SFA. All the neurons had a membrane time constant of $\tau_m = 20$ ms, a baseline threshold $v_{\text{th}} = 30$ mV, a refractory period of 5 ms, and a synaptic delay of 1 ms. LIF neurons with SFA in the network had an adaptation strength of $\beta = 1.7$ mV. It was not necessary to assign particular values to adaptation time constants of firing thresholds of neurons with SFA; we simply chose them uniformly randomly to be between 1 ms and 6000 ms, mimicking the diversity of SFA effects found in the neocortex Allen Institute, 2018 in a qualitative manner. All other parameters were the same as in the other experiments. We trained the network using 5 different network initializations (seeds) and tested it on previously unseen strings. Average test success rate was 95.88% with standard deviation 1.39%.

Analysis of spiking data. We used 3-way ANOVA to analyze if a neuron's firing rate is significantly affected by task, serial position in the sequence, symbol identity, or combination of these (similar to Lindsay et al., 2017). We refer to these factors as "conditions". The analysis was performed on the activity of the neurons of the trained network during 50,000 test episodes. For the analysis, neurons whose average firing rate over all episodes was

lower than 2Hz or greater than 60Hz were discarded from the analysis to remove large outliers. This left 279 out of the 320 neurons. From each episode, a serial position from the input period was chosen randomly, and hence each episode could be used only once, i.e., as one data point. This was to make sure that each entry in the 3-way ANOVA was completely independent of other entries, since the neuron activity within an episode is highly correlated. Each data point was labelled with the corresponding triple of (task type, serial position, symbol identity). To ensure that the dataset was balanced, the same number of data points per particular combination of conditions was used, discarding all the excess data points, resulting in a total of 41,850 data points. To categorize a neuron as selective to one or more conditions, or combination of conditions, we observed p-values obtained from $3-$way ANOVA and calculated the effect size ω^2 for each combination of conditions. If the p-value was smaller than 0.001 and ω^2 greater than 0.14 for a particular combination of conditions, the neuron was categorized as selective to that combination of conditions. The ω^2 threshold of 0.14 was suggested by Field, 2013 to select large effect sizes. Each neuron can have a large effect size for more than one combination of conditions. Thus the values shown in Fig. 4.3D sum to > 1. The neuron shown in Fig. 4.3E had the most prominent selectivity for the combination of Task $\times$ Position $\times$ Symbol, with $\omega^2 = 0.394$ and $p < 0.001$. The neuron shown in Fig. 4.3F was categorized as selective to a combination of Position $\times$ Symbol category, with $\omega^2 = 0.467$ and $p < 0.001$. While the 3-way ANOVA tells us if a neuron is selective to a particular combination of conditions, it does not give us the exact task/symbol/position that the neuron is selective to. To find the specific task/symbol/position that the neuron was selective to, Welch's t-test was performed, and a particular combination with maximum t-statistic and $p < 0.001$ was chosen to be shown in Fig. 4.3E,F.

5. Contributions of other biophysical mechanisms to the temporal computing capability of SNNs

Contents

5.1. Introduction

We compare the contribution of SFA to the temporal computing capability of SNNs with the contribution of the two most frequently considered other slow processes in SNNs: facilitating short-term plasticity of synapses (STP-F) and depressing short-term plasticity of synapses (STP-D). In addition we consider a dual version of a neuron with SFA, a neuron model where the firing threshold of a neuron decreases with firing of the neuron. Such neuron model had been considered for example in Fransén et al., 2006.

We compare the contribution of these other three mechanisms to the temporal computing capability of an LSNN (SNN with SFA) for two temporal

computing tasks: one-dimensional STORE-RECALL and a task where the short term memory has to be continuously updated: time series classification (sMNIST).

Facilitating short-term plasticity of synapses, also referred to as paired pulse facilitation, increases the amplitudes of postsynaptic potentials for the later spikes in a spike train, see Methods. Whereas synaptic connections between pyramidal cells in the neocortex are usually depressing H. Markram et al., 2015, it was shown in Y. Wang et al., 2006 that there are facilitating synaptic connections between pyramidal cells in the medial prefrontal cortex of rodents, with a mean time constant of 507 ms (std 37 ms) for facilitation. It was shown in Mongillo, Barak, and Tsodyks, 2008 that if one triples the experimentally found mean time constant for facilitation, then this mechanism supports basic working memory tasks.

Depressing short-term plasticity of synapses, also referred to as paired pulse depression, reduces the amplitude of postsynaptic potentials for later spikes in a spike train. The impact of this mechanism on simple temporal computing tasks had been examined in a number of publications Maass, Natschläger, and Henry Markram, 2002; Buonomano and Maass, 2009; Masse et al., 2019; Hu et al., 2020.

Finally, we considered a dual version of the LIF with SFA neuron model: Its excitability is increased through preceding firing (see Methods). We call this neuron model the enhanced-excitability LIF (ELIF) model. Neurons with such property have been found for example in the entorhinal cortex Fransén et al., 2006. But a transient increase in the excitability of a neuron is also caused by depolarization-mediated suppression of inhibition, and this effect has been observed in many brain areas Kullmann et al., 2012.

The dynamics of the salient hidden variables in these three models is illustrated in Fig. 5.2.

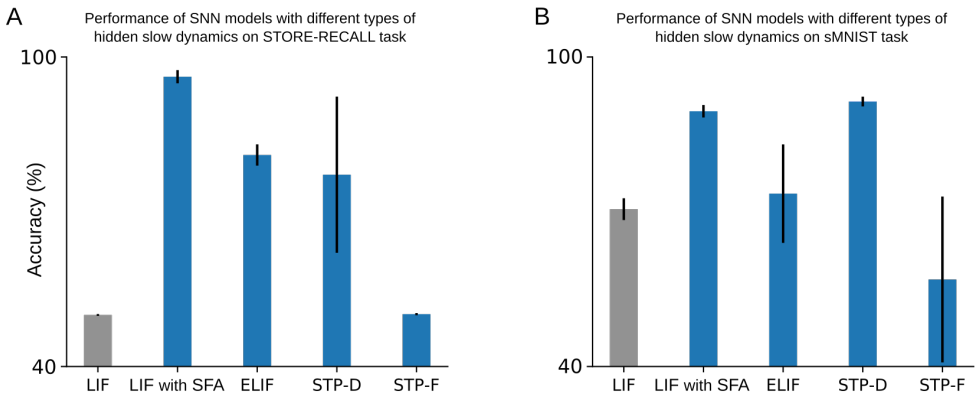

Fig. 5.1.: Temporal computing performance of SNNs with different slow biophysical mechanisms. (**A**) Test set accuracy of five variants of the SNN model on the one-dimensional STORE-RECALL task. Mean accuracy and standard deviation are shown for 10 runs with different network initializations for all 5 network types. (**B**) Test set accuracy of the same five variants of the SNN model for the sMNIST time series classification task. Mean accuracy and standard deviation are shown for a minimum of 4 runs with different network initializations for all 5 network types. LSNNs do very well for both tasks, much better than SNNs with facilitating short term plasticity of synapses (STP-F).

5.2. Comparison of the four mechanisms on the one-dimensional STORE-RECALL task.

We considered five SNNs consisting of 60 neurons, with four of them being endowed with one of the previously discussed biophysical mechanisms, and a fifth baseline SNN without any of these mechanisms. Their performance for this simple temporal computing task is shown in Fig. 5.1A. An LSNN (SNN with SFA) achieved by far the highest accuracy.

5.3. Comparison of the four mechanism for the time series classification task sMNIST.

Fig. 5.1B shows that both LSNNs and SNNs with experimentally reported parameters for depressing short-term synaptic plasticity (STP-D) also achieve very high performance on sMNIST. Furthermore, the performance of SNNs with STP-F is in the same range as the control SNN (LIF) that has no slow mechanism.

Fig. 5.1 also shows that replacing SFA by the inverse mechanism (ELIF) yields substantially worse temporal computing capabilities for both tasks. One possible reason is that information that is stored in the firing threshold of a neuron is better protected in the case of a neuron with SFA, since an increased firing threshold suppresses subsequent accidental firing, and hence accidental modifications of the content that is stored in the firing threshold. In contrast, for an ELIF neuron the information that is stored in the firing threshold is quite vulnerable, since a decreased firing threshold invites accidental firing.

5.4. Methods

5.4.1. Network models

The LIF neuron model, LIF neuron model with SFA, weight initialization, and training methods are the same as in Chapter 4.

LIF neurons with activity-dependant increase in excitability: ELIF neurons. There exists experimental evidence that some neurons fire for the same stimulus more for a repetition of the same sensory stimulus. We refer to such neurons as ELIF neurons, since they are becoming more excitable. Such repetition enhancement was discussed for example in Tartaglia, Mongillo, and Brunel, 2015. But to the best of our knowledge, it has remained open whether repetition enhancement is a network effect, resulting for example from a transient depression of inhibitory synapses onto the cell that is caused by postsynaptic firing Kullmann et al., 2012, or a result of an intrinsic firing property of some neurons. We used a simple model for ELIF neurons that is dual to the above described LIF neuron model with SFA: The threshold is lowered through each spike of the neuron, and then decays exponentially back to its resting value. This can be achieved by using a negative value for β in equation (2.5).

Models for Short-Term Plasticity (STP) of synapses. We modelled the STP dynamic according to the classical model of STP in Mongillo, Barak, and Tsodyks, 2008. The STP dynamics in discrete time, derived from the equations in Mongillo, Barak, and Tsodyks, 2008, are as follows:

$$u'_{ji}(t+\delta t) = \exp\left(\frac{-\delta t}{F}\right) u'_{ji}(t) + U_{ji}(1-u_{ji}(t))z_i(t)\delta t, \tag{5.1}$$

$$u_{ji}(t+\delta t) = U_{ji} + u'_{ji}(t), \tag{5.2}$$

$$r'_{ji}(t+\delta t) = \exp\left(\frac{-\delta t}{D}\right) r'_{ji}(t) + u_{ji}(t)(1-r'_{ji}(t))z_i(t)\delta t, \tag{5.3}$$

$$r_{ji}(t+\delta t) = 1 - r'_{ji}(t), \tag{5.4}$$

$$W^{STP}_{ji}(t+\delta t) = W^{rec}_{ji} u_{ji}(t) r_{ji}(t), \tag{5.5}$$

where $z_i(t)$ is the spike train of the pre-synaptic neuron and W_{ji}^{rec} scales the synaptic efficacy of synapses from neuron i to neuron j. Networks with STP were constructed from LIF neurons with the weight W_{ji}^{rec} in equation (2.2) replaced by the time dependent weight $W_{ji}^{STP}(t)$.

STP time constants of facilitation-dominant and depression-dominant network models were based on values of experimental recordings in Y. Wang et al., 2006 of PFC-E1 ($D = 194 \pm 18$, $F = 507 \pm 37$, $U = 0.28 \pm 0.02$) and PFC-E2 ($D = 671 \pm 17$, $F = 17 \pm 5$, $U = 0.25 \pm 0.02$) synapse types respectively. Recordings in Y. Wang et al., 2006 were performed in medial prefrontal cortex of young adult ferrets. In the sMNIST task for the depression-dominant network model (STP-D) we used values based on PFC-E2, and for facilitation-dominant network model (STP-F) we used values based on PFC-E1, see sMNIST task section below. For the STORE-RECALL task, both facilitation and depression time constants were equally scaled up until the larger time constant matched the requirement of the task, see one-dimensional STORE-RECALL task section below.

5.4.2. Tasks

One-dimensional STORE-RECALL task. The input to the network consisted of 40 input neurons: 10 for STORE, 10 for RECALL, and 20 for population coding of a binary feature. Whenever a subpopulation was active, it would exhibit a Poisson firing with frequency of 50 Hz. For experiments reported in Fig. 5.1A each input sequence consisted of 20 steps (200 ms each) where the STORE or the RECALL populations were activated with probability 0.09 interchangeably which resulted in delays between the STORE-RECALL pairs to be in the range [200, 3600] ms.

Networks were trained for 400 iterations with batch size of 64 in Table 4.1 and 128 in Fig. 5.1A. We used the Adam optimizer with default parameters and initial learning rate of 0.01 which was decayed every 100 iterations by a factor of 0.3. The same firing rate regularization term was added to the loss as in the 20-dimensional STORE-RECALL setup (see above). The test performance was computed as batch average over 2048 random input sequences.

Networks consisted of 60 recurrently connected neurons. The membrane time constant was $\tau_m = 20\,\text{ms}$. In Fig.5.1A, for LIF with SFA and ELIF networks, we used $\beta = 1\,\text{mV}$ and $\beta = -0.5\,\text{mV}$ respectively, with $\tau_a = 2000\,\text{ms}$. Table 4.1 defines the adaptation time constants and expected delay of the experiments in that section. Synapse parameters of STP-D network were $F = 51 \pm 15\,\text{ms}$, $D = 2000 \pm 51\,\text{ms}$ and $U = 0.25$, and of STP-F network $F = 2000 \pm 146\,\text{ms}$, $D = 765 \pm 71\,\text{ms}$ and $U = 0.28$. The baseline threshold voltage was $10\,\text{mV}$ for all models except ELIF for which it was $20\,\text{mV}$. Synaptic delay was $1\,\text{ms}$. The same sigmoidal readout neuron setup was used as in the one-dimensional STORE-RECALL setup (see above).

In Fig. 5.1A the LSNN (SNN with SFA) reached $96 \pm 1.2\%$ accuracy. The performance of SNNs with STP-D or ELIF neurons was consistently lower with $77 \pm 15.1\%$ and $81 \pm 2.0\%$ accuracy respectively. The SNN with STP-F performed at chance level ($50 \pm 0.2\%$), similar as the baseline SNN (LIF) without any additional biophysical mechanisms, even when the time constant for factiliation was increased to a much higher value of $2000\,\text{ms}$ than in the recorded data Y. Wang et al., 2006.

sMNIST task. The input consisted of sequences of 784 pixel values created by unrolling the handwritten digits of the MNIST dataset, one pixel after the other in a scanline manner. We used $1\,\text{ms}$ presentation time for each pixel gray value. Each of the 80 input neurons was associated with a particular threshold for the grey value, and this input neuron fired whenever the grey value crossed its threshold in the transition from the previous to the current pixel.

Networks were trained for 36,000 iterations using the Adam optimizer with batch size 256. The initial learning rate was 0.01 and every 2500 iterations the learning rate was decayed by a factor of 0.8. The same firing rate regularization term was added to the loss as in the STORE-RECALL setup (see above) but with the scaling coefficient of 0.1.

All networks consisted of 220 neurons. Network models labeled LIF with SFA and ELIF in the Fig. 5.1B had 100 neurons out of 220 with SFA or transient excitability respectively. The neurons had a membrane time constant of $\tau_m = 20\,\text{ms}$, a baseline threshold of $v_{\text{th}} = 10\,\text{mV}$, and a refractory period of $5\,\text{ms}$. The adaptation time constants of LIF with SFA and ELIF neurons

were $\tau_a = 700\,\text{ms}$. The adaptation strength of LIF neurons with SFA was $\beta = 1.8\,\text{mV}$, and of ELIF neurons $\beta = -0.9\,\text{mV}$. Synaptic delay was $1\,\text{ms}$. Synapse parameters were $F = 20\,\text{ms}$, $D = 700\,\text{ms}$ and $U = 0.2$ for STP-D model, and $F = 500\,\text{ms}$, $D = 200\,\text{ms}$ and $U = 0.2$ for STP-F model. The output of the network was produced by the softmax of 10 linear output neurons that received the low-pass filtered version of the spikes from all neurons in the network. The low-pass filter had a time constant of $20\,\text{ms}$. For training the network to classify into one of the ten classes we used cross-entropy loss computed between the labels and the softmax of output neurons.

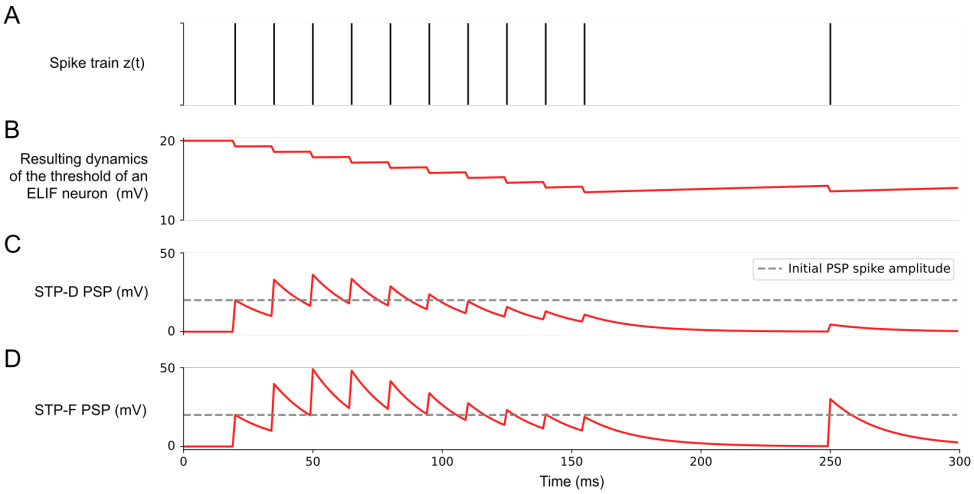

Fig. 5.2.: Illustration of models for an inversely adapting ELIF neuron, and for short-term synaptic plasticity. (**A**) Sample spike train. (**B**) The resulting evolution of firing threshold for an inversely adapting neuron (ELIF neuron). (**C-D**) The resulting evolution of the amplitude of postsynaptic potentials (PSPs) for spikes of the presynaptic neuron for the case of a depression-dominant (STP-D: D$\gg$F) and a facilitation-dominant (STP-F: F$\gg$D) short-term synaptic plasticity.

6. A biologically plausible learning method for recurrent networks of spiking neurons

Contents

Abstract. Recurrently connected networks of spiking neurons underlie the astounding information processing capabilities of the brain. But in spite of extensive research, it has remained open how they can learn through synaptic plasticity to carry out complex network computations. We argue that two pieces of this puzzle were provided by experimental data from neuroscience. A mathematical result in Bellec, Scherr, Subramoney, et al., 2020 tells us how these pieces need to be combined to enable biologically plausible online network learning through gradient descent, in particular deep reinforcement learning. This learning method – called *e-prop* – approaches the performance of backpropagation through time (BPTT), the best-known method for training recurrent neural networks in machine learning. In addition, it suggests a method for powerful on-chip learning in energy-efficient spike-based hardware for artificial intelligence. This chapter introduces the *adaptive e-prop* variant and presents the evaluation results on two tasks.

Acknowledgments and author contributions. This chapter is based on the manuscripts

> Guillaume Bellec*, Franz Scherr*, Anand Subramoney, Elias Hajek, Darjan Salaj, Robert Legenstein, Wolfgang Maass (2020). "A solution to the learning dilemma for recurrent networks of spiking neurons." *Nature Communications*.

To this study, I contributed as co-author developing and conducting the speech recognition related experiments and the adaptive version of *e-prop*. The theoretical development was carried out jointly by GB and FS, after the idea of *e-prop* was first sketched by GB and the research in this direction was initiated by WM. The theorical work around reward-based *e-prop* was mainly developed by FS with contributionsfrom GB and AS. All authors contributed to the design of the experiments. The experiments concerning Section 3.2.2 were developed and conducted by DS and GB. The experiments concerning Section 3.2.3 were developed and conducted by EH, FS and GB. The experiments concerningreward-based *e-prop* in Section 3.2.4 were developed and conducted by FS. The manuscript was written by WM, RL, GB, FS, EH, DS and AS.

6.1. Introduction

Networks of neurons in the brain differ in at least two essential aspects from deep neural networks in machine learning: They are recurrently connected, forming a giant number of loops, and they communicate via asynchronously emitted stereotypical electrical pulses, called spikes, rather than bits or numbers that are produced in a synchronized manner by each layer of a deep feedforward network. We consider the arguably most prominent model for spiking neurons in the brain: leaky integrate-and-fire (LIF) neurons, where spikes that arrive from other neurons through synaptic connections are multiplied with the corresponding synaptic weight, and are linearly integrated by a leaky membrane potential. The neuron fires – i.e., emits a spike – when the membrane potential reaches a firing threshold.

But it is an open problem how recurrent networks of spiking neurons (RSNNs) can learn, i.e., how their synaptic weights can be modified by local rules for synaptic plasticity so that the computational performance of the network improves. In deep learning, this problem is solved for feedforward networks through gradient descent for a loss function E that measures imperfections of current network performance (LeCun, Bengio, and G. Hinton, 2015). Gradients of E are propagated backwards through all layers of the feedforward network to each synapse through a process called backpropagation. Recurrently connected networks can compute more efficiently because each neuron can participate several times in a network computation, and they are able to solve tasks that require integration of information over time or a non-trivial timing of network outputs according to task demands. Therefore the impact of a synaptic weight on the loss function (see Figure 6.1a) is more indirect, and learning through gradient descent becomes substantially more difficult. This problem is aggravated if there are slowly changing hidden variables in the neuron model, as in neurons with spike-frequency adaptation (SFA). Neurons with SFA are quite common in the neocortex Allen Institute, 2018, and it turns out that their inclusion in the RSNN significantly increases the computational power of the network Bellec, Salaj, et al., 2018. In fact, RSNNs trained through gradient descent acquire then similar computing capabilities as networks of LSTM (Long Short-Term Memory) units, the state of the art for recurrent neural networks in machine learning. Because of this functional relation to LSTM networks these RSNN models are referred to as LSNNs Bellec, Salaj, et al., 2018.

In machine learning, one trains recurrent neural networks by unrolling the network into a virtual feedforward network LeCun, Bengio, and G. Hinton, 2015, see Figure 6.1b, and applying the backpropagation algorithm to that (Figure 6.1c). This method is called backpropagation through time (*BPTT*) since it requires propagation of gradients backwards in time.

With a careful choice of the pseudo-derivative for handling the discontinuous dynamics of spiking neurons one can apply *BPTT* also to RSNNs, and RSNNs were able to learn in this way to solve really demanding computational tasks (Bellec, Salaj, et al., 2018, Huh and Sejnowski, 2018). But the dilemma is that *BPTT* requires storing the intermediate states of all neurons during a network computation, and merging these in a subsequent offline process with gradients that are computed backwards in time (see

Figure 6.1c). This makes it very unlikely that *BPTT* is used by the brain (T. P. Lillicrap and Santoro, 2019).

The previous lack of powerful online learning methods for RSNNs also affected the use of neuromorphic computing hardware, which aims at a drastic reduction in the energy consumption of AI implementations. A substantial fraction of this neuromorphic hardware, such as SpiNNaker Furber, Galluppi, et al., 2014 or Intel's Loihi chip Davies et al., 2018, implements RSNNs. Although it does not matter here whether the learning algorithm is biologically plausible, the excessive storage and offline processing demands of *BPTT* make this option unappealing. Hence there also exists a learning dilemma for RSNNs in neuromorphic hardware.

We are not aware of previous work on online gradient descent learning methods for RSNNs, neither for supervised learning nor for reinforcement learning (RL). There exists, however, preceding work on online approximations of gradient descent for non-spiking neural networks based on Williams and Zipser, 1989, which we review in the Discussion Section.

Two streams of experimental data from neuroscience provide clues about the organisation of online network learning in the brain:

Firstly, neurons in the brain maintain traces of preceding activity on the molecular level, for example in the form of calcium ions or activated CaMKII enzymes (Sanhueza and Lisman, 2013). In particular, they maintain a fading memory of events where the presynaptic neuron fired before the postsynaptic neuron, which is known to induce synaptic plasticity if followed by a top-down learning signal (Cassenaer and Laurent, 2012; Yagishita et al., 2014; Gerstner, Lehmann, et al., 2018). Such traces are often referred to as eligibility traces.

Secondly, in the brain, there exists an abundance of top-down signals such as dopamine, acetylcholine, and neural firing (Sajad, Godlove, and Schall, 2019) related to the event-related negativity (ERN), that inform local populations of neurons about behavioral results. Furthermore, dopamine signals (Engelhard et al., 2019; Roeper, 2013) have been found to be specific for different target populations of neurons, rather than being global. We refer in our learning model to such top-down signals as learning signals.

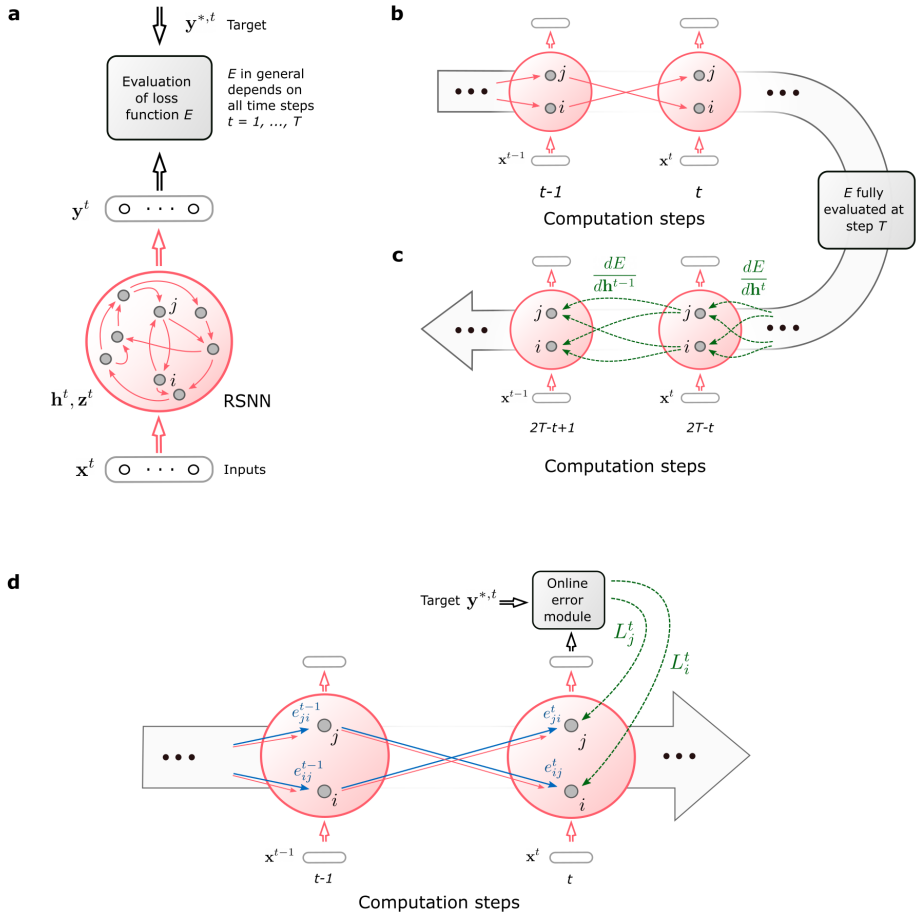

Fig. 6.1.: Schemes for *BPTT* and *e-prop* (**a**) RSNN with network inputs $\mathbf{x}$, neuron spikes $\mathbf{z}$, hidden neuron states $\mathbf{h}$, and output targets $\mathbf{y}^*$, for each time step t of the RSNN computation. Output neurons $\mathbf{y}$ provide a low-pass filter of a weighted sum of network spikes $\mathbf{z}$. (**b**) *BPTT* computes gradients in the unrolled version of the network. It has a new copy of the neurons of the RSNN for each time step t. A synaptic connection from neuron i to neuron j of the RSNN is replaced by an array of feedforward connections, one for each time step t, that goes from the copy of neuron i in the layer for time step t to a copy of neuron j in the layer for time step $t+1$. All synapses in this array have the same weight: the weight of this synaptic connection in the RSNN. (**c**) Loss gradients of *BPTT* are propagated backwards in time and retrograde across synapses in an offline manner, long after the forward computation has passed a layer. **d**) Online learning dynamics of *e-prop*. Feedforward computation of eligibility traces is indicated in blue. These are combined with online learning signals according to equation 1 in Bellec, Scherr, Subramoney, et al., 2020).

A re-analysis of the mathematical basis of gradient descent learning in recurrent neural networks tells us how local eligibility traces and top-down learning signals should be optimally combined – without requiring back-progation of signals through time. The resulting learning method *e-prop* is illustrated in Figure 6.1d. It learns slower than *BPTT*, but tends to approximate the performance of *BPTT*, thereby providing a first solution to the learning dilemma for RSNNs. Furthermore, *e-prop* also works for RSNNs with more complex neuron models, such as LSNNs.

This new learning paradigm elucidates how the brain could learn to recognize phonemes in spoken language, solve temporal credit assignment problems, and acquire new behaviors just from rewards Bellec, Scherr, Subramoney, et al., 2020.

6.2. Variants of *e-prop*

e-prop (Bellec, Scherr, Subramoney, et al., 2020) is an online learning method in a strict sense (see Figure 6.1d) where the weights are updated according to the following equation:

$$\frac{dE}{dW_{ji}} = \sum_t L_j^t \, e_{ji}^t \tag{6.1}$$

where E is the loss function to be minimized and e_{ji}^t is the eligibility trace of the synapse from neuron i to neuron j at time t. The gradient $\frac{dE}{dW_{ji}}$ for the weight W_{ji} of the synapse from neuron i to neuron j tells us how this weight should be changed in order to reduce E.

The ideal value $\frac{dE}{dz_j^t}$ of the learning signal L_j^t is replaced by an approximation, such as $\frac{\partial E}{\partial z_j^t}$, which ignores these indirect influences (this partial derivative $\frac{\partial E}{\partial z_j^t}$ is written with a rounded ∂ to signal that it captures only the direct influence of the spike z_j^t on the loss function E). This approximation takes only currently arising losses at the output neurons k of the RSNN into

account, and routes them with neuron-specific weights B_{jk} to the network neurons j (see Figure 6.2a):

$$L_j^t = \sum_k B_{jk} \underbrace{\left(y_k^t - y_k^{*,t}\right)}_{\substack{\text{deviation of output } k \\ \text{at time } t}} . \tag{6.2}$$

Although this approximate learning signal L_j^t only captures errors that arise at the current time step t, it is combined in equation (6.1) with an eligibility trace e_{ji}^t that may reach far back into the past of neuron j (see Figure 6.3b), thereby alleviating the need to solve the temporal credit assignment problem by propagating signals backwards in time (like in *BPTT*).

There are several strategies for choosing the weights B_{jk} for this online learning signal.

In symmetric e-prop we set it equal to the corresponding weight W_{kj}^{out} of the synaptic connection from neuron j to output neuron k, as demanded by $\frac{\partial E}{\partial z_j^t}$. Note that this learning signal would actually implement $\frac{dE}{dz_j^t}$ exactly in the absence of recurrent connections in the network. Biologically more plausible are two variants of *e-prop* that avoid weight sharing:

In *random e-prop* the values of all weights B_{jk} – even for neurons j that are not synaptically connected to output neuron k – are randomly chosen and remain fixed, similar to Broadcast Alignment for feedforward networks (Timothy P Lillicrap et al., 2016; Nøkland, 2016; Samadi, Timothy P Lillicrap, and Tweed, 2017).

In *adaptive e-prop*, in addition to using random backward weights, we also let B_{jk} evolve through a simple local plasticity rule that mirrors the plasticity rule applied to W_{kj}^{out} for neurons j that are synaptically connected to output neuron k (see subsection 6.3).

6.3. *Adaptive e-prop* and weight decay regularization

In *adaptive e-prop* all neurons are receiving learning signals with random broadcast weights B_{jk} as in random e-prop. Yet, if neuron j sends activity

to the output neuron k via output weights W_{kj}^{out}, the broadcast weights B_{jk} are subsequently adapted. This was implemented by applying an identical weight update to the broadcast and the output weights, i.e. $\Delta B_{jk} = \Delta W_{kj}^{\text{out}}$ and by subtracting $c_{\text{decay}} \cdot W_{kj}^{\text{out}}$ (resp. $c_{\text{decay}} \cdot B_{jk}$) from the weight W_{kj}^{out} (resp. B_{jk}) after each weight update, where $c_{\text{decay}} > 0$ is the regularization factor (see specific experiments for the value of c_{decay}). This weight decay in combination with the mirroring of the weight updates has the effect that, despite different initialization, the output weights and the adaptive broadcast weights converge to similar values. The remaining difference of performance between *symmetric* and *adaptive e-prop* reported in Supplementary Figure B.1 and Supplementary Figure B.3 may be explained by the different initializations.

Adaptive e-prop can be viewed as that version of *e-prop* that exploits all types of biologically plausible learning signals that are available at individual neurons, apart from reward prediction errors or signals from a separately trained RSNN that sends learning signals, see Figure 3 of (Bellec, Scherr, Hajek, et al., 2019).

6.4. Learning phoneme recognition with e-prop

The phoneme recognition task TIMIT (Garofolo et al., 1993) is one of the most commonly used benchmarks for temporal processing capabilities of different types of recurrent neural networks and different learning approaches Greff et al., 2017. It comes in two versions. Both use, as input, acoustic speech signals from sentences that are spoken by 630 speakers from 8 dialect regions of the USA (see the top of Figure 6.2b for a sample segment). In the simpler version, used for example in Greff et al., 2017, the goal is to recognize which of 61 phonemes is spoken in each 10 ms time frame (frame-wise classification). In the more sophisticated version from Graves, Mohamed, and G. Hinton, 2013, which achieved an essential step toward human-level performance in speech-to-text transcription, the goal is to recognize the sequence of phonemes in the entire spoken sentence independently of their timing (sequence transcription). RSNNs consisting only of LIF neurons do not even reach good performance on TIMIT with *BPTT* Bellec, Salaj, et al.,

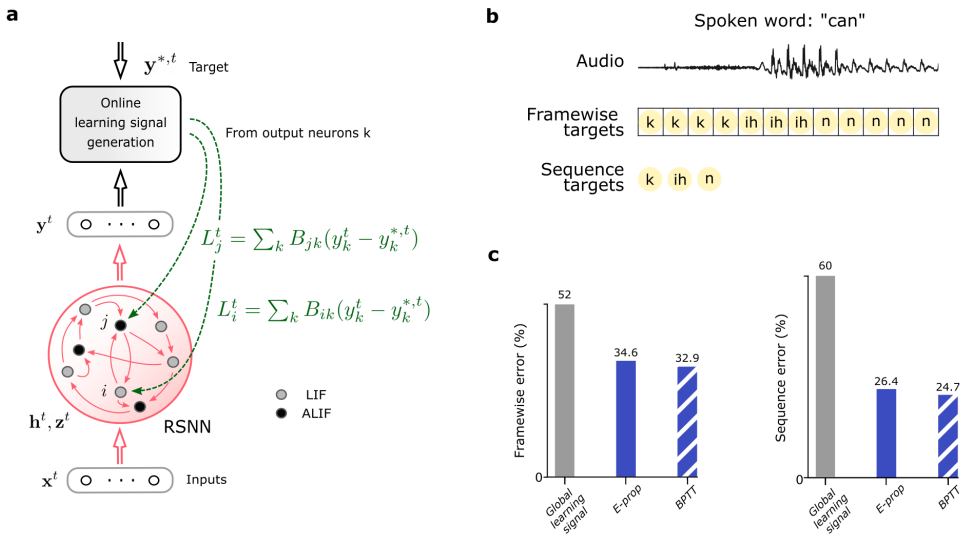

Fig. 6.2.: Comparison of *BPTT* and *e-prop* for learning phoneme recognition (**a**) Network architecture for *e-prop*, illustrated for an LSNN consisting of LIF and ALIF neurons. (**b**) Input and target output for the two versions of TIMIT. (**c**) Performance of *BPTT* and symmetric e-prop for LSNNs consisting of 800 neurons for framewise targets and 2400 for sequence targets (random and *adaptive e-prop* produced similar results, see Supplementary Figure B.1). To obtain the Global learning signal baselines, the neuron-specific feedbacks are replaced with global ones.

2018. Hence we are considering here LSNNs, where a random subset of the neurons is a variation of the LIF model with firing rate adaptation (ALIF neurons), see Chapter 3. The name LSNN is motivated by the fact that this special case of the RSNN model can achieve through training with *BPTT* similar performance as an LSTM network Bellec, Salaj, et al., 2018.

E-prop approximates the performance of *BPTT* on LSNNs for both versions of TIMIT very well, as shown in Figure 6.2c. Furthermore, LSNNs could solve the frame-wise classification task without any neuron firing more frequently than 12 Hz (spike count taken over 32 spoken sentences), demonstrating that they operate in an energy-efficient spike-coding – rather than a rate-coding – regime. For the more difficult version of TIMIT we trained as in Graves, Mohamed, and G. Hinton, 2013 a complex LSNN consisting of a feedforward sequence of three recurrent networks. Our results show that *e-prop* can also handle learning for such more complex network structures very well. In Supplementary Figure B.3 we show for comparison also the performance of *e-prop* and *BPTT* for LSTM networks on the same tasks. These data show that for both versions of TIMIT the performance of *e-prop* for LSNNs comes rather close to that of *BPTT* for LSTM networks. In addition, they show that *e-prop* also provides for LSTM networks a functionally powerful online learning method.

6.5. Solving difficult temporal credit assignment

A hallmark of cognitive computations in the brain is the capability to go beyond a purely reactive mode: to integrate diverse sensory cues over time, and to wait until the right moment arrives for an action. A large number of experiments in neuroscience analyze neural coding after learning such tasks (see e.g. Morcos and Christopher D Harvey, 2016; Engelhard et al., 2019). But it had remained unknown how one can model the learning of such cognitive computations in RSNNs of the brain. In order to test whether *e-prop* can solve this problem, we considered the same task that was studied in the experiments of Morcos and Christopher D Harvey, 2016 and Engelhard et al., 2019. There a rodent moved along a linear track in a virtual environment, where it encountered several visual cues on the left and right, see Figure 6.3a. Later, when it arrived at a T-junction, it had to decide whether to turn left

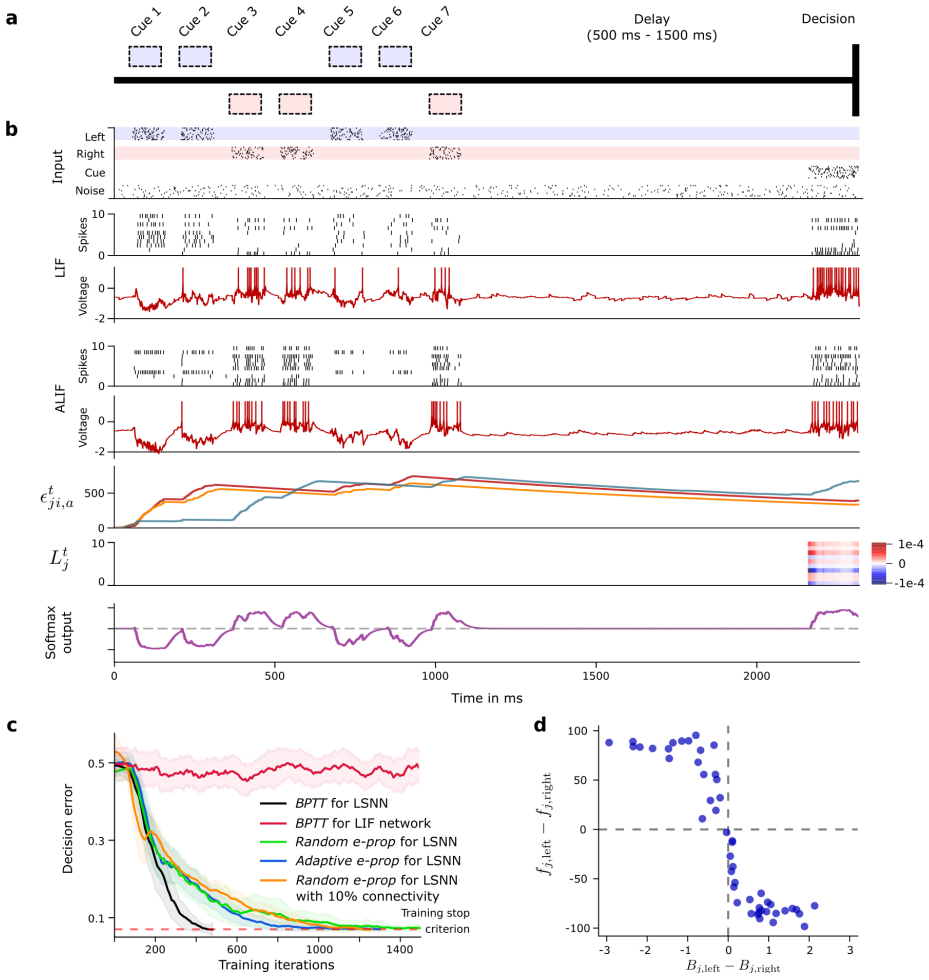

Fig. 6.3.: Solving a task with difficult temporal credit assignment. (**a**) Setup of corresponding rodent experiments of Morcos and Christopher D Harvey, 2016 and Engelhard et al., 2019, see Supplmentary Movie 1. (**b**) Input spikes, spiking activity of 10 out of 50 sample LIF neurons and 10 out of 50 sample ALIF neurons, membrane potentials (more precisely: $v^t_j - A^t_j$) for two sample neurons j, 3 samples of slow components of eligibility traces, sample learning signals for 10 neurons and softmax network output. (**c**) Learning curves for *BPTT* and two *e-prop* versions applied to LSNNs, and *BPTT* applied to an RSNN without adapting neurons (red curve). Orange curve shows learning performance of *e-prop* for a sparsely connected LSNN consisting of excitatory and inhibitory neurons (Dale's law obeyed). The shaded areas are the 95%-confidence intervals of the mean accuracy computed with 20 runs. (**d**) Correlation between the randomly drawn broadcast weights B_{jk} for k = left/right for learning signals in *random e-prop* and resulting sensitivity to left and right input components after learning. $f_{\text{j,left}}$ ($f_{\text{j,right}}$) was the resulting average firing rate of neuron j during presentation of left (right) cues after learning.

or right. It was rewarded when it turned to that side from which it had previously received the majority of visual cues. This task is not easy to learn since the subject needs to find out that it does not matter on which side the last cue was, or in which order the cues were presented. Instead, the subject has to learn to count cues separately for each side and to compare the two resulting numbers. Furthermore, the cues need to be processed properly long before a reward is given. We discuss here the case where a teacher tells the subject at the end of each trial what would have been the right decision. This yields a challenging scenario for any online learning method since non-zero learning signals L_j^t arise only during the last 150 ms of a trial (Figure 6.3b). Hence all synaptic plasticity has to take place during these last 150 ms, long after the input cues have been processed.

Nevertheless, *e-prop* is able to solve this learning problem, see Figure 6.3c. It just needs a bit more time to reach the same performance level as offline learning via *BPTT*. Whereas this task can not even be solved by *BPTT* with a regular RSNN that has no adapting neurons (red curve in Figure 6.3c), all 3 previously discussed variations of *e-prop* can solve it if the RSNN contains adapting neurons.

But how can the neurons in the LSNN learn to record and count the input cues if all the learning signals are identically 0 until the last 150 ms of a 2250 ms long trial (see 2nd to last row of Figure 6.3b)?

For answering this question one should note that firing of a neuron j at time t can affect the loss function E at a later time point $t' > t$ in two different ways: Via route (i) it affects future values of slow hidden variables of neuron j (e.g., its firing threshold), which may then affect the firing of neuron j at t' , which in turn may directly affect the loss function at time t'. Via route (ii) it affects the firing of other neurons j' at t', which directly affects the loss function at time t'.

In symmetric and *adaptive e-prop*, one uses the partial derivative $\frac{\partial E}{\partial z_j^t}$ as learning signal L_j^t for *e-prop* – instead of the total derivative $\frac{dE}{dz_j^t}$ which is not available online. This blocks the flow of gradient information along route ii. But the eligibility trace keeps the flow along route i open. Therefore even symmetric and *adaptive e-prop* can solve the temporal credit assignment problem of Figure 6.3 through online learning: The gradient information

that flows along route i enables neurons to learn how to process the sensory cues at time points t during the first 1050 ms, although this can affect the loss only at time points $t' > 2100$ ms when the loss becomes non-zero.

This is illustrated in the 3rd last row of Figure 6.3b: The slow component $\epsilon^t_{ji,a}$ of the eligibility traces e_{ji} of adapting neurons j decays with the typical long time constant of firing rate adaptation. Since these traces stretch from the beginning of the trial into its last phase, they enable learning of differential responses to left and right input cues that arrived over 1050 ms before any learning signals become non-zero, as shown in the 2nd to last row of Figure 6.3b.

Hence eligibility traces provide so-called highways into the future for the propagation of gradient information. These can be seen as biologically realistic replacements for the highways into the past that *BPTT* employs during its backwards pass.

This analysis also tells us when symmetric e-prop is likely to fail to approximate the performance of *BPTT*: If the forward propagation of gradients along route i cannot reach those later time points t' at which the value of the loss function becomes salient. One can artificially induce this in the experiment of Figure 6.3 by adding to the LSNN – which has the standard architecture shown in Figure 6.2a – hidden layers of a feedforward SNN through which the communication between the LSNN and the readout neurons has to flow. The neurons j' of these hidden layers block route i, while leaving route ii open. Hence the task of Figure 6.3 can still be learnt with this modified network architecture by *BPTT*, but not by symmetric e-prop.

Identifying tasks where the performance of *random e-prop* stays far behind that of BPTT is more difficult, since error signals are sent there also to neurons that have no direct connections to readout neurons. For deep feedforward networks it has been shown in (Bartunov et al., 2018) that Broadcast Alignment, as defined in (Samadi, Timothy P Lillicrap, and Tweed, 2017; Nøkland, 2016), cannot reach the performance of Backprop for difficult image classification tasks. Hence we expect that random e-prop will exhibit similar deficiencies with deep feedforward SNNs on difficult classification tasks. We are not aware of corresponding demonstrations of failures of

Broadcast Alignment for artificial RNNs, although they are likely to exist. Once they are found, they will probably point to tasks where *random e-prop* fails for RSNNs. Currently, we are not aware of any.

Figure 6.3d provides insight into the functional role of the randomly drawn broadcast weights in *random e-prop*: The difference of these weights determines for each neuron j whether it learns to respond in the first phase of a trial more to cues from the left or right. This observation suggests that neuron-specific learning signals for RSNNs have the advantage that they can create a diversity of feature detectors for task-relevant network inputs. Hence a suitable weighted sum of these feature detectors is later able to cancel remaining errors at the network output, similarly as in the case of feedforward networks (Timothy P Lillicrap et al., 2016).

7. Outlook

In this thesis we investigated the role of SFA in the temporal computation and learning capabilities of SNNs. We demonstrated that the LSNNs (SNNs with SFA) have significantly more powerful memory and learning capabilities over SNNs without SFA. In Chapter 3, we showed that LSNNs, in addition to improved memory, are very amenable to optimization with BPTT. With this we show similarities between LSTMs and LSNNs both in terms of accuracies on machine learning benchmark tasks and their friendliness to gradient descent training. These results have been highly attractive to the neuromorphic industry and the LSNN model has already been implemented in SpiNNaker(Furber, Galluppi, et al., 2014) and Loihi(Davies et al., 2018) chips to demonstrate the viability of the platforms for AI applications. The learning paradigm introduced in Chapter 6 is another highly attractive development for the neuromorphic domain as it offers the possibility of on-chip learning.

Following up in Chapter 4 we investigated in more depth the question of how SFA is able to boost the temporal computation of an SNN in different scenarios. There we developed the negative imprint hypothesis which offers an explanation how the SFA is exploited as the mechanism that supports robust memory and computing capabilities. This hypothesis could be used for design of neuroscience experiments which attempt to more accurately capture the relevance of SFA in temporal cognitive tasks.

In Chapter 5, we demonstrated that different biophysical mechanisms have different impact on the temporal computation and learning abilities of SNNs. Interestingly, we show that predominantly depressing short-term plasticity of synapses is another biophysical mechanism that can strongly enhance the computing capabilities of SNNs. This points to the promising direction of investigating further neural and synapse dynamics that might be powerful substrates for other unknown computing capabilities of SNNs. Furthermore,

the combination of the different biophysical mechanisms might lead to new capabilities which produces results greater than the sum of individual contributions of those mechanisms.

Appendix

Appendix A.

Appendix to Chapter 4: Spike frequency adaptation supports network computations on temporally dispersed information

A.1. Autocorrelation based intrinsic time scale of neurons trained on STORE-RECALL task

We wondered whether the adaptive firing threshold of LIF neurons with SFA affects the autocorrelation function of their firing activity — termed intrinsic time scale in Wasmuht et al., 2018. We tested this for an SNN consisting of 200 LIF neurons without and 200 LIF neurons with SFA that was trained to solve a one-dimensional version of the STORE-RECALL task. It turned out that during the delay between STORE and RECALL these intrinsic time constants were in the same range as those measured in the monkey cortex, see Fig. 1C in Wasmuht et al., 2018. Furthermore, neurons of the trained SNN exhibited very similar distributions of these time constants (see Fig. A.1), suggesting that these intrinsic time constants are determined largely by their network inputs, and less by the neuron type.

A.2. sMNIST task with sparsely connected SNN

This task has originally been used as a temporal processing benchmark for ANNs, and has successfully been solved with the Long Short-Term Memory (LSTM) type of ANNs Hochreiter and Schmidhuber, 1997. LSTM units store information in registers – like a digital computer – so that the stored information cannot be perturbed by ongoing network activity. Networks of LSTM units or variations of such units have been widely successful in temporal processing and reach the level of human performance for many temporal computing tasks.

Since LSTM networks also work well for tasks on larger time-scales, for comparing SNNs with LSTM networks, we used a version of the task with 2 ms presentation time per pixel, thereby doubling the length of sequences to be classified to 1568 ms. Grey values of pixels were presented to the LSTM network simply as analog values. A trial of a trained SNN with SFA (with an input sequence that encodes a handwritten digit "3" using population rate coding) is shown in Fig. A.2B. The top row of Fig. A.2B shows a version where the grey value of the currently presented pixel is encoded by population coding, through the firing probability of 80 input neurons. Somewhat better performance was achieved when each of the 80 input neurons was associated with a particular threshold for the grey value, and this input neuron fired whenever the grey value crossed its threshold in the transition from the previous to the current pixel (this input convention was used to produce the results below).

Besides a fully connected network of LIF neurons with SFA, we also tested the performance of a variant of the model, called SC-SNN, that integrates additional constraints of SNNs in the brain: It is sparsely connected (12% of possible connections are present) and consists of 75% excitatory and 25% inhibitory neurons that adhere to Dale's law. By adapting the sparse connections with the rewiring method in Bellec, Kappel, et al., 2018 during *BPTT* training, the SC-SNN was able to perform even better than the fully-connected SNN of LIF neurons with SFA. The resulting architecture of the SC-SNN is shown in Fig. A.2C. Its activity of excitatory and inhibitory neurons, as well as the time courses of adaptive thresholds for (excitatory) LIF neurons with SFA of the SC-SNN are shown in Fig. A.2B. In this setup,

the SFA had $\tau_a = 1400\,\text{ms}$. When we used an SNN with SFA, we improved the accuracy on this task to 96.4% which approaches the accuracy of the artificial LSTM model which reached the accuracy of 98.0%.

We also trained a liquid state machine version of the SNN model with SFA where only the readout neurons are trained. This version of the network reached the accuracy of $63.24 \pm 1.48\%$ over 5 independent training runs.

A.3. Google Speech Commands

We trained SNNs with and without SFA on the keyword spotting task with Google Speech Commands Dataset Warden, 2018 (v0.02). The dataset consists of 105,000 audio recordings of people saying thirty different words. Fully connected networks were trained to classify audio recordings, that were clipped to one second length, into one of 12 classes (10 keywords, as well as two special classes for silence and unknown words; the remaining 20 words had to be classified as "unknown"). Comparison of the maximum performance of trained spiking networks against state-of-the-art artificial recurrent networks is shown in Table A.1. Averaging over 5 runs, the SNN with SFA reached $90.88 \pm 0.22\%$, and the SNN without SFA reached $88.79 \pm 0.16\%$ accuracy. Thus an SNN without SFA can already solve this task quite well, but the inclusion of SFA halves the performance gap to the published state-of-the-art in machine learning. The only other report on a solution to this task with spiking networks is Zenke and Vogels, 2020. There the authors train a network of LIF neurons using surrogate gradients with *BPTT* and achieve $85.3 \pm 0.3\%$ accuracy on the full 35 classes setup of the task. In this setup, the SNN with SFA reached $88.5 \pm 0.16\%$ test accuracy.

Features were extracted from the raw audio using the Mel Frequency Cepstral Coefficient (MFCC) method with 30 ms window size, 1 ms stride, and 40 output features. The network models were trained to classify the input features into one of the 10 keywords (yes, no, up, down, left, right, on, off, stop, go) or to two special classes for silence or unknown word (where the remainder of 20 recorded keywords are grouped). The training, validation and test set were assigned 80, 10, and 10 percent of data respectively while making sure that audio clips from the same person stay in the same set.

All networks were trained for 18,000 iterations using the Adam optimizer with batch size 100. The output spikes of the networks were averaged over time, and the linear readout layer was applied to those values. During the first 15,000 iterations, we used a learning rate of 0.001 and for the last 3000, we used a learning rate of 0.0001. The loss function contained a regularization term (scaled with coefficient 0.001) that minimizes the squared difference of average firing rate between individual neurons and a target firing rate of 10 Hz.

Both SNNs with and without SFA consisted of 2048 fully connected neurons in a single recurrent layer. The neurons had a membrane time constant of $\tau_m = 20\,\text{ms}$, the adaptation time constant of SFA was $\tau_a = 100\,\text{ms}$, adaptation strength was $\beta = 2\,\text{mV}$. The baseline threshold was $v_{\text{th}} = 10\,\text{mV}$, and the refractory period was 2 ms. The synaptic delay was 1 ms.

A.4. Delayed-memory XOR

We also tested the performance of SNNs with SFA on a previously considered benchmark task, where two items in the working memory have to be combined non-linearly: the Delayed-memory XOR task Huh and Sejnowski, 2018. The network is required to compute the exclusive-or operation on the history of input pulses when prompted by a go-cue signal, see Fig. A.3.

The network received on one input channel two types of pulses (up or down), and a go-cue on another channel. If the network received two input pulses since the last go-cue signal, it should generate the output "1" during the next go-cue if the input pulses were different or "0" if the input pulses were the same. Otherwise, if the network only received one input pulse since the last go-cue signal, it should generate a null output (no output pulse). Variable time delays are introduced between the input and go-cue pulses. The time scale of the task was 600 ms which limited the delay between input pulses to 200 ms.

This task was solved in Huh and Sejnowski, 2018, without providing performance statistics, by using a type of neuron that has not been documented in biology — a non-leaky quadratic integrate and fire neuron. We are not

aware of previous solutions by networks of LIF neurons. To compare and investigate the impact of SFA on network performance in the delayed-memory XOR task, we trained SNNs, with and without SFA, of the same size as in Huh and Sejnowski, 2018 — 80 neurons. Across 10 runs, SNNs with SFA solved the task with $95.19 \pm 0.014\%$ accuracy, whereas the SNNs without SFA converged at lower $61.30 \pm 0.029\%$ accuracy.

The pulses on the two input channels were generated with 30 ms duration and the shape of a normal probability density function normalized in the range $[0, 1]$. The pulses were added or subtracted from the baseline zero input current at appropriate delays. The go-cue was always a positive current pulse. The 6 possible configurations of the input pulses ($+$, $-$, $++$, $--$, $+-$, $-+$) were sampled with equal probability during training and testing.

Networks were trained for 2000 iterations using the Adam optimizer with batch size 256. The initial learning rate was 0.01 and every 200 iterations the learning rate was decayed by a factor of 0.8. The loss function contained a regularization term (scaled with coefficient 50) that minimizes the squared difference of the average firing rate of individual neurons from a target firing rate of 10 Hz. This regularization resulted in networks with a mean firing rate of 10 Hz where firing rates of individual neurons were spread in the range [1, 16] Hz.

Both SNNs with and without SFA consisted of 80 fully connected neurons in a single recurrent layer. The neurons had a membrane time constant of $\tau_m = 20\,\text{ms}$, a baseline threshold $v_{\text{th}} = 10\,\text{mV}$, and a refractory period of 3 ms. SFA had an adaptation time constant of $\tau_a = 500\,\text{ms}$ and an adaptation strength of $\beta = 1\,\text{mV}$. The synaptic delay was 1 ms. For training the network to classify the input into one of the three classes, we used the cross-entropy loss between the labels and the softmax of three linear readout neurons. The input to the linear readout neurons were the neuron traces that were calculated by passing all the network spikes through a low-pass filter with a time constant of 20 ms.

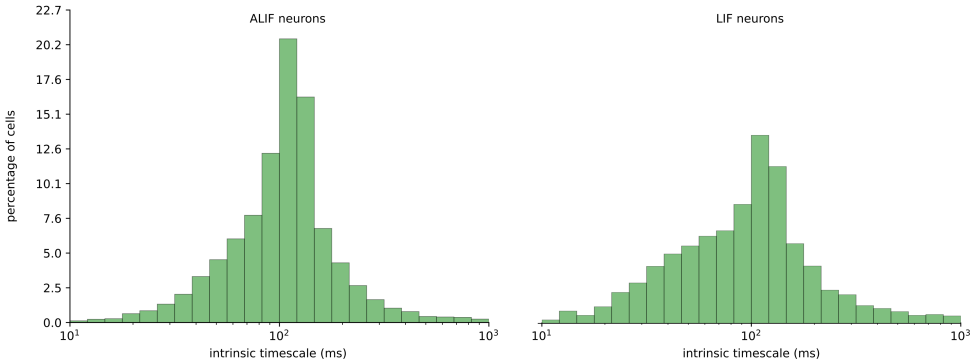

Fig. A.1.: Histogram of the intrinsic time scale of neurons trained on STORE-RECALL task. We trained 64 randomly initialized SNNs consisting of 200 LIF neurons with and 200 without SFA on the single-feature STORE-RECALL task. Measurements of the intrinsic time scale were performed according to Wasmuht et al., 2018 on the spiking data of SNNs solving the task after training. Averaged data of all 64 runs is presented in the histogram. The distribution is very similar for neurons with and without SFA.

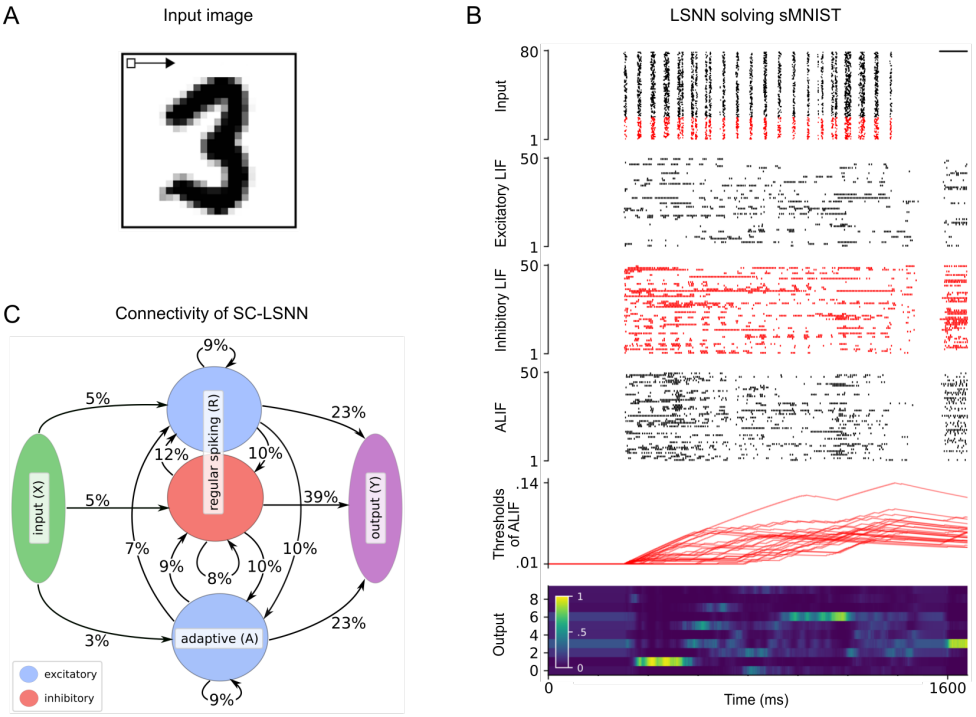

Fig. A.2.: sMNIST time series classification benchmark task. (**A**) Illustration of the pixel-wise input presentation of handwritten digits for sMNIST. (**B**) Rows top to bottom: Input encoding for an instance of the sMNIST task, network activity, and temporal evolution of firing thresholds for randomly chosen subsets of neurons in the SC-SNN, where 25% of the LIF neurons were inhibitory (their spikes are marked in red). The light color of the readout neuron for digit "3" around 1600 ms indicates that this input was correctly classified. (**C**) Resulting connectivity graph between neuron populations of an SC-SNN after BPTT optimization with DEEP R on sMNIST task with 12% global connectivity limit.

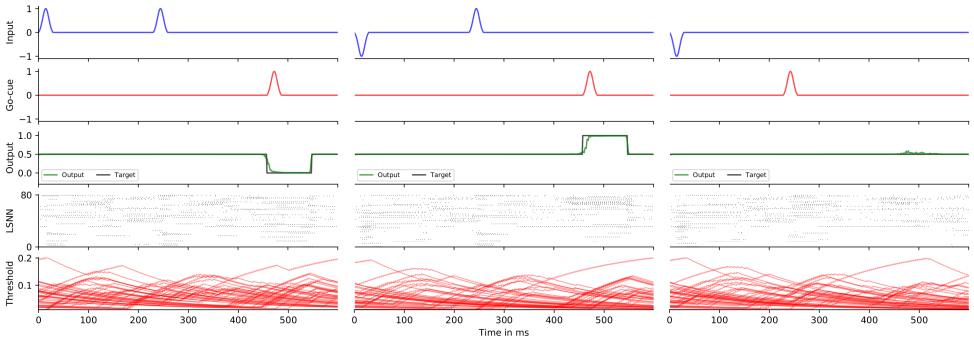

Fig. A.3.: Delayed-memory XOR task. Rows top to bottom: Input signal, Go-cue signal, network readout, network activity, and temporal evolution of firing thresholds.

Model	test accuracy (%)
FastGRNN-LSQ Kusupati et al., 2018	93.18
SNN with SFA	91.21
SNN	89.04

Table A.1.: Google Speech Commands. Accuracy of the spiking network models on the test set compared to the state-of-the-art artificial recurrent model reported in Kusupati et al., 2018. Accuracy of the best out of 5 simulations for SNNs is reported.

Appendix B.

Appendix to Chapter 6: A solution to the learning dilemma for recurrent networks of spiking neurons

B.1. Supplementary Figures

B.1.1. Figure B.1 Comparison of learning algorithms for training LSNNs on the TIMIT task

B.1.2. Figure B.2 Performance of e-prop on the framewise TIMIT task for LSNNs without recurrent connections

B.1.3. Figure B.3 LSTM networks trained with BPTT and e-prop on the TIMIT task

B.1.4. Figure B.4 Impact of the length of the simulation time step on the learning performance of e-prop for the task of Figure 6.3

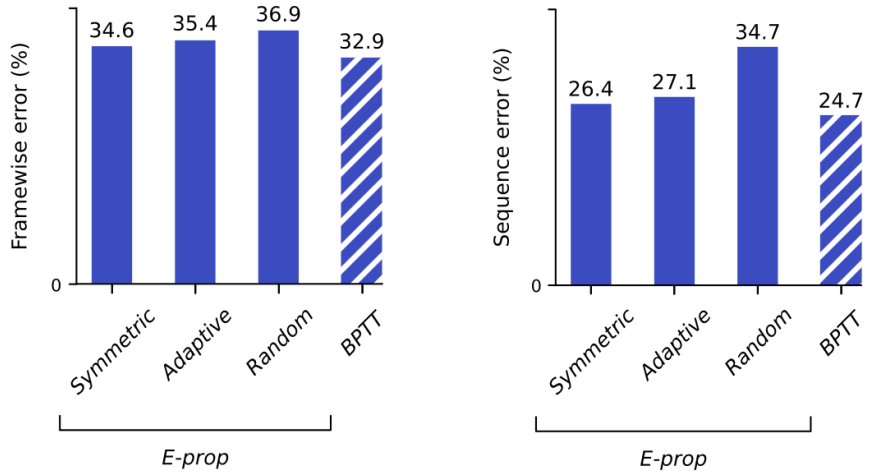

Fig. B.1.: Comparison of learning algorithms for training LSNNs on the TIMIT task. Performance of BPTT and the three versions of e-prop on frame-wise phoneme classification (left) and for phoneme sequence recognition (right).

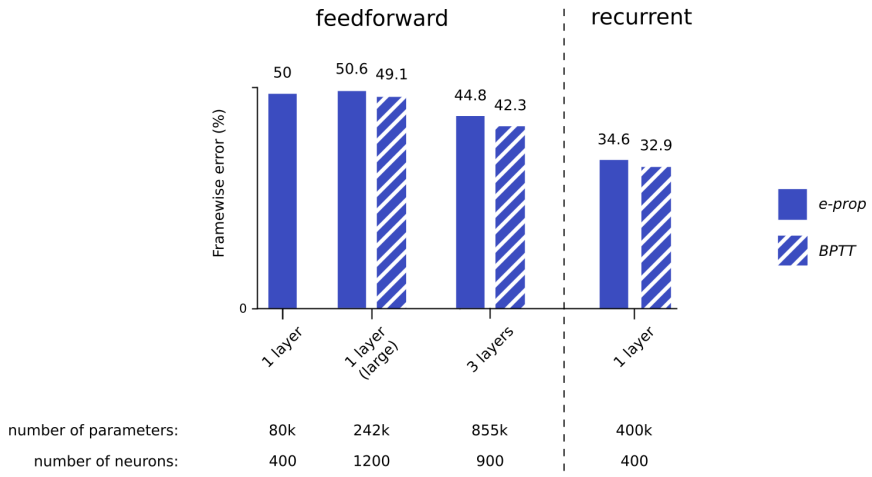

Fig. B.2.: Performance of e-prop on the framewise TIMIT task for variations of the LSNN from Figure 6.2 without recurrent connections (left of dashed line). The result of the left-most bar was achieved with the same hyperparameters as used in Figure 6.2. For the other feedforward architectures we modified hyperparameters to optimize the performance. The results of Figure 6.2 **c)** for LSNNs with recurrent connections are redrawn on the right of the dashed line. One sees that recurrent connections are essential for achieving good performance on this task.

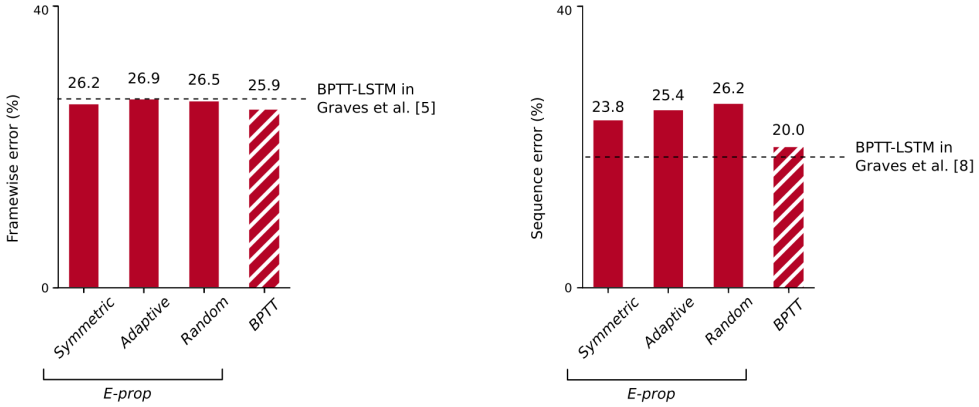

Fig. B.3.: LSTM networks trained with BPTT and e-prop on the TIMIT task. Performance of BPTT and the three versions of e-prop on frame-wise phoneme classification (left) and for phoneme sequence recognition (right). One sees that e-prop approximates BPTT performance also for non-spiking neural networks. The BPTT baselines aim at reproducing the results obtained with LSTM networks in Graves and Schmidhuber, 2005 and Graves, Mohamed, and G. Hinton, 2013. The network architectures and audio pre-processing settings are taken from (Graves and Schmidhuber, 2005; Greff et al., 2017) for framewise phoneme classifcation and from Graves, Mohamed, and G. Hinton, 2013 for phonemene sequence transcription. In comparison to the BPTT-LSTM baselines that we could achieve in this way, 26.9% framewise error rate was reported in Graves and Schmidhuber, 2005 and 18.6% sequence error rate was reported in Graves, Mohamed, and G. Hinton, 2013.

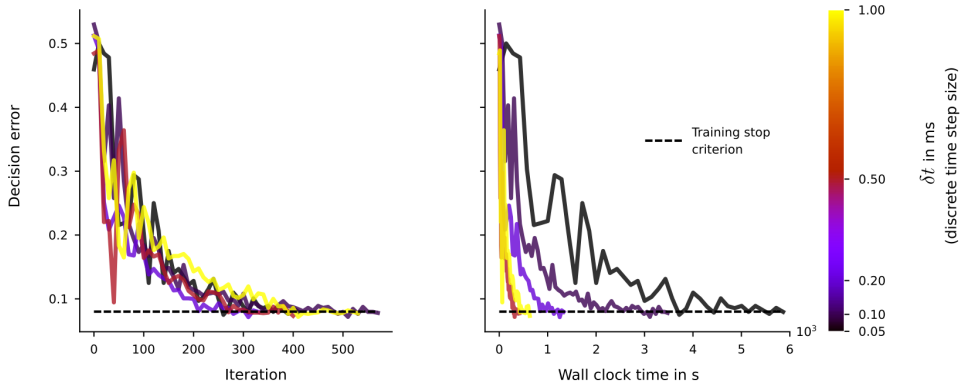

Fig. B.4.: Impact of the length of the simulation time step on the learning performance of e-prop for the task of Figure 6.3 (simplified version with 5 cues instead of 7). Decision error for this task is shown as function of the number of training iterations, and as function of wall clock time. The results are averaged over 5 different seeds. One sees that the length of the simulation time step has no visible impact on the learning performance of e-prop. However, smaller simulation time steps lead to substantially larger simulation time

Bibliography

Allen Institute (Oct. 2017). *Allen Cell Types Database Technical white paper: GLIF models*. Tech. rep. v4. URL: %5Curl%7Bhttp://help.brain-map.org/download/attachments/8323525/GLIFModels.pdf%7D (cit. on p. 13).

Allen Institute (2018). "© 2018 Allen Institute for Brain Science. Allen Cell Types Database, cell feature search. Available from: celltypes.brain-map.org/data." In: (cit. on pp. 9, 11, 12, 14, 33, 71, 85).

Barnett, Mark W and Philip M Larkman (2007). "The action potential." In: *Practical neurology* 7.3, pp. 192–197 (cit. on p. 8).

Barone, P and J-P Joseph (1989). "Prefrontal cortex and spatial sequencing in macaque monkey." In: *Experimental brain research* 78.3, pp. 447–464 (cit. on pp. 49, 59, 60, 64).

Bartunov, Sergey et al. (2018). "Assessing the scalability of biologically-motivated deep learning algorithms and architectures." In: *Advances in Neural Information Processing Systems* (cit. on p. 95).

Bellec, Guillaume, David Kappel, et al. (2018). "Deep Rewiring: Training very sparse deep networks." In: *International Conference on Learning Representations (ICLR)* (cit. on pp. 19, 21, 35, 39, 44, 102).

Bellec, Guillaume, Darjan Salaj, et al. (2018). "Long short-term memory and learning-to-learn in networks of spiking neurons." In: *Advances in Neural Information Processing Systems*, pp. 787–797 (cit. on pp. 13, 85, 90, 92).

Bellec, Guillaume, Franz Scherr, Elias Hajek, et al. (Jan. 2019). "Biologically inspired alternatives to backpropagation through time for learning in recurrent neural nets." In: *arXiv:1901.09049 [cs]*. arXiv: 1901.09049. URL: http://arxiv.org/abs/1901.09049 (cit. on p. 90).

Bellec, Guillaume, Franz Scherr, Anand Subramoney, et al. (2020). "A solution to the learning dilemma for recurrent networks of spiking neurons." In: *Nature Communications* (cit. on pp. 5, 20, 83, 87, 88).

Benda, Jan and Andreas VM Herz (2003). "A universal model for spike-frequency adaptation." In: *Neural computation* 15.11, pp. 2523–2564 (cit. on p. 13).

Benda, Jan, Leonard Maler, and André Longtin (2010). "Linear versus non-linear signal transmission in neuron models with adaptation currents or dynamic thresholds." In: *Journal of Neurophysiology* 104.5, pp. 2806–2820 (cit. on p. 13).

Bengio, Yoshua, Patrice Simard, and Paolo Frasconi (1994). "Learning long-term dependencies with gradient descent is difficult." In: *Neural Networks, IEEE Transactions on* 5.2, pp. 157–166. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=279181 (visited on 10/14/2015) (cit. on p. 35).

Berg, Esta A (1948). "A simple objective technique for measuring flexibility in thinking." In: *The Journal of general psychology* 39.1, pp. 15–22 (cit. on p. 56).

Buchweitz, Augusto (Apr. 2008). "Models of Working Memory: Mechanisms of Active Maintenance and Executive Control." In: (cit. on p. 2).

Buonomano, Dean V and Wolfgang Maass (2009). "State-dependent computations: spatiotemporal processing in cortical networks." In: *Nature Reviews Neuroscience* 10.2, pp. 113–125 (cit. on p. 74).

Carpenter, Adam F et al. (2018). "Encoding of serial order in working memory: neuronal activity in motor, premotor, and prefrontal cortex during a memory scanning task." In: *Journal of Neuroscience* 38.21, pp. 4912–4933 (cit. on pp. 49, 59, 60, 63, 64).

Cassenaer, S. and G. Laurent (2012). "Conditional modulation of spike-timing-dependent plasticity for olfactory learning." In: *Nature* 482.7383, p. 47 (cit. on p. 86).

Chettih, S. N. and C. D. Harvey (2019). "Single-neuron perturbations reveal feature-specific competition in V1." In: *Nature* 567.7748, pp. 334–340 (cit. on p. 71).

Costa, Rui et al. (2017). "Cortical microcircuits as gated-recurrent neural networks." In: *Advances in Neural Information Processing Systems*, pp. 272–283 (cit. on p. 22).

Courbariaux, Matthieu et al. (2016). "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1." In: *arXiv preprint arXiv:1602.02830* (cit. on pp. 15, 19).

Cowan, Nelson (2008). "Chapter 20 What are the differences between long-term, short-term, and working memory?" In: *Progress in Brain Research*. Elsevier, pp. 323–338. DOI: 10.1016/s0079-6123(07)00020-9. URL: https://doi.org/10.1016/s0079-6123(07)00020-9 (cit. on p. 3).

Davies, Mike et al. (2018). "Loihi: A neuromorphic manycore processor with on-chip learning." In: *IEEE Micro* 38.1, pp. 82–99 (cit. on pp. 20, 49, 86, 97).

Deneve, Sophie (2008). "Bayesian spiking neurons I: inference." In: *Neural computation* 20.1, pp. 91–117 (cit. on p. 13).

DePasquale, Brian, Mark M Churchland, and LF Abbott (2016). "Using firing-rate dynamics to train recurrent networks of spiking model neurons." In: *arXiv preprint arXiv:1601.07620* (cit. on p. 20).

Diamond, Adele (Jan. 2013). "Executive Functions." In: *Annual Review of Psychology* 64.1, pp. 135–168. DOI: 10.1146/annurev-psych-113011-143750. URL: https://doi.org/10.1146/annurev-psych-113011-143750 (cit. on p. 3).

Duan, Yan et al. (2016). "RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning." In: *arXiv preprint arXiv:1611.02779* (cit. on pp. 19, 25, 26, 30).

Eliasmith, Chris (2013). *How to build a brain: A neural architecture for biological cognition*. Oxford University Press (cit. on p. 20).

Engelhard, Ben et al. (2019). "Specialized coding of sensory, motor and cognitive variables in VTA dopamine neurons." In: *Nature*, p. 1 (cit. on pp. 86, 92, 93).

Ermentrout, Bard (1998). "Linearization of FI curves by adaptation." In: *Neural computation* 10.7, pp. 1721–1729 (cit. on p. 13).

Esser, Steven K. et al. (Nov. 2016). "Convolutional networks for fast, energy-efficient neuromorphic computing." en. In: *Proceedings of the National Academy of Sciences* 113.41, pp. 11441–11446. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.1604850113 (cit. on pp. 15, 19, 20).

Field, Andy (2013). *Discovering statistics using IBM SPSS statistics*. sage (cit. on pp. 63, 72).

Fitz, Hartmut et al. (2020). "Neuronal spike-rate adaptation supports working memory in language processing." In: *Proceedings of the National Academy of Sciences* 117.34, pp. 20881–20889 (cit. on p. 13).

Fransén, Erik et al. (2006). "Mechanism of graded persistent cellular activity of entorhinal cortex layer v neurons." In: *Neuron* 49.5, pp. 735–746 (cit. on pp. 73, 74).

Furber, Steve B, Francesco Galluppi, et al. (2014). "The spinnaker project." In: *Proceedings of the IEEE* 102.5, pp. 652–665 (cit. on pp. 49, 86, 97).

Furber, Steve B, David R Lester, et al. (2013). "Overview of the spinnaker system architecture." In: *IEEE Transactions on Computers* 62.12, pp. 2454–2467 (cit. on p. 20).

Garofolo, J. S. et al. (1993). "DARPA TIMIT acoustic-phonetic continous speech corpus CD-ROM." In: *NASA STI/Recon Technical Report N, DOI: [http://doi.org/10.6028/nist.ir.4930]* (cit. on p. 90).

Gerstner, Wulfram and Werner Kistler (2002). *Spiking Neuron Models: An Introduction*. New York, NY, USA: Cambridge University Press. ISBN: 0521890799 (cit. on p. 8).

Gerstner, Wulfram and Werner M Kistler (2002). *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press (cit. on p. 10).

Gerstner, Wulfram, Werner M Kistler, et al. (2014). *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press (cit. on p. 13).

Gerstner, Wulfram, Marco Lehmann, et al. (July 2018). "Eligibility Traces and Plasticity on Behavioral Time Scales: Experimental Support of NeoHebbian Three-Factor Learning Rules." In: *Frontiers in Neural Circuits* 12. ISSN: 1662-5110. DOI: 10.3389/fncir.2018.00053. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6079224/ (cit. on p. 86).

Glass, James, Arthur Smith, and Andrew K. Halberstadt (Feb. 1999). "Heterogeneous Acoustic Measurements and Multiple Classifiers for Speech Recognition." In: (cit. on p. 37).

Glorot, Xavier and Yoshua Bengio (2010). "Understanding the difficulty of training deep feedforward neural networks." In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256 (cit. on p. 41).

Gouwens, Nathan W et al. (2018). "Systematic generation of biophysically detailed models for diverse cortical neuron types." In: *Nature communications* 9.1, pp. 1–13 (cit. on p. 13).

Graves, Alex, Abdel-rahman Mohamed, and Geoffrey Hinton (2013). "Speech recognition with deep recurrent neural networks." In: *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, pp. 6645–6649 (cit. on pp. 90, 92, 112).

Graves, Alex and Jürgen Schmidhuber (2005). "Framewise phoneme classification with bidirectional LSTM and other neural network architectures." In: *Neural Networks* 18.5, pp. 602–610 (cit. on p. 112).

Greff, Klaus et al. (2017). "LSTM: A search space odyssey." In: *IEEE transactions on neural networks and learning systems* (cit. on pp. 23, 90, 112).

Gutkin, B. and F. Zeldenrust (2014). "Spike frequency adaptation." In: *Scholarpedia* 9.2. revision #143322, p. 30643. DOI: 10.4249/scholarpedia.30643 (cit. on p. 13).

György Buzsáki, MD (2019). *The brain from inside out*. Oxford University Press (cit. on p. 2).

Harvey, Christopher D, Philip Coen, and David W Tank (2012). "Choice-specific sequences in parietal cortex during a virtual-navigation decision task." In: *Nature* 484.7392, pp. 62–68 (cit. on p. 62).

Hasson, Uri, Janice Chen, and Christopher J Honey (2015). "Hierarchical process memory: memory as an integral component of information processing." In: *Trends in cognitive sciences* 19.6, pp. 304–313 (cit. on p. 19).

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory." In: *Neural computation* 9.8, pp. 1735–1780 (cit. on p. 102).

Hochreiter, Sepp, A Steven Younger, and Peter R Conwell (2001). "Learning to learn using gradient descent." In: *International Conference on Artificial Neural Networks*. Springer, pp. 87–94 (cit. on p. 26).

Hodgkin, Alan L, Andrew F Huxley, and Bernard Katz (1952). "Measurement of current-voltage relations in the membrane of the giant axon of Loligo." In: *The Journal of physiology* 116.4, pp. 424–448 (cit. on p. 10).

Hu, B. et al. (2020). "Adaptation supports short-term memory in a visual change detection task." In: *bioRxiv* (cit. on p. 74).

Huh, Dongsung and Terrence J Sejnowski (2018). "Gradient descent for spiking neural networks." In: *Advances in Neural Information Processing Systems*, pp. 1433–1443 (cit. on pp. 20, 21, 56, 85, 104, 105).

Kappel, David, Stefan Habenschuss, et al. (2015). "Network Plasticity as Bayesian Inference." In: *PLOS Computational Biology* 11.11, e1004485. (Visited on 12/05/2016) (cit. on p. 21).

Kappel, David, Robert Legenstein, et al. (2018). "Reward-based stochastic self-configuration of neural circuits." In: *eNEURO* (cit. on pp. 19, 21).

Kilpatrick, Zachary P and Bard Ermentrout (2011). "Sparse gamma rhythms arising through clustering in adapting neuronal networks." In: *PLoS Comput Biol* 7.11, e1002281 (cit. on p. 13).

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization." In: *arXiv preprint arXiv:1412.6980* (cit. on pp. 40, 42, 66).

Kruijne, Wouter et al. (2020). "Flexible Working Memory through Selective Gating and Attentional Tagging." In: *Neural Computation* 0.0. PMID: 33080159, pp. 1–40. DOI: 10.1162/neco_a_01339. eprint: https://doi.org/10.1162/neco_a_01339. URL: https://doi.org/10.1162/neco_a_01339 (cit. on p. 57).

Kullmann, Dimitri M et al. (2012). "Plasticity of inhibition." In: *Neuron* 75.6, pp. 951–962 (cit. on pp. 74, 77).

Kusupati, Aditya et al. (2018). "Fastgrnn: A fast, accurate, stable and tiny kilobyte sized gated recurrent neural network." In: *Advances in Neural Information Processing Systems*, pp. 9017–9028 (cit. on p. 108).

Lashley, Karl Spencer (1951). *The problem of serial order in behavior*. Vol. 21. Bobbs-Merrill Oxford, United Kingdom (cit. on p. 59).

Le, Quoc V., Navdeep Jaitly, and Geoffrey E. Hinton (2015). "A Simple Way to Initialize Recurrent Networks of Rectified Linear Units." In: *CoRR* abs/1504.00941. arXiv: 1504.00941. URL: http://arxiv.org/abs/1504.00941 (cit. on p. 22).

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). "Deep learning." In: *Nature* (cit. on p. 85).

Levenstein, Daniel et al. (Apr. 2020). "On the role of theory and modeling in neuroscience." In: *arXiv:2003.13825 [q-bio]*. URL: http://arxiv.org/abs/2003.13825 (visited on 04/02/2020) (cit. on p. 2).

Lillicrap, T. P. and A. Santoro (2019). "Backpropagation through time and the brain." In: *Current Opinion in Neurobiology* 55, pp. 82–89 (cit. on p. 86).

Lillicrap, Timothy P et al. (2016). "Random synaptic feedback weights support error backpropagation for deep learning." In: *Nature Communications* 7, p. 13276 (cit. on pp. 89, 96).

Lindsay, Grace W et al. (2017). "Hebbian learning in a random network captures selectivity properties of the prefrontal cortex." In: *Journal of Neuroscience* 37.45, pp. 11021–11036 (cit. on p. 71).

Liu, Y. et al. (2019). "Human replay spontaneously reorganizes experience." In: *Cell* 178.3, pp. 640–652 (cit. on pp. 49, 59, 60).

Maass, Wolfgang (1997). "Networks of spiking neurons: the third generation of neural network models." In: *Neural networks* 10.9, pp. 1659–1671 (cit. on pp. 2, 10).

Maass, Wolfgang, Thomas Natschläger, and Henry Markram (2002). "Real-time computing without stable states: A new framework for neural computation based on perturbations." In: *Neural computation* 14.11, pp. 2531–2560 (cit. on p. 74).

MacDonald III, Angus W (2008). "Building a clinically relevant cognitive task: case study of the AX paradigm." In: *Schizophrenia bulletin* 34.4, pp. 619–628 (cit. on p. 56).

Marcus, Gary F (2003). *The Algebraic Mind: Integrating Connectionism and Cognitive Science*. MIT Press (cit. on pp. 4, 59, 60).

Marder, Eve et al. (1996). "Memory from the dynamics of intrinsic membrane currents." In: *Proceedings of the national academy of sciences* 93.24, pp. 13481–13486 (cit. on p. 13).

Markram, H. et al. (2015). "Reconstruction and simulation of neocortical microcircuitry." In: *Cell* 163.2, pp. 456–492 (cit. on p. 74).

Martinolli, Marco, Wulfram Gerstner, and Aditya Gilra (July 2018). "Multi-Timescale Memory Dynamics Extend Task Repertoire in a Reinforcement Learning Network With Attention-Gated Memory." eng. In: *Frontiers in computational neuroscience* 12. PMC6055065[pmcid], pp. 50–50. ISSN: 1662-5188. DOI: 10.3389/fncom.2018.00050. URL: https://doi.org/10.3389/fncom.2018.00050 (cit. on p. 57).

Masse, N. Y. et al. (2019). "Circuit mechanisms for the maintenance and manipulation of information in working memory." In: *Nature Neuroscience*, p. 1 (cit. on p. 74).

Mensi, Skander et al. (2012). "Parameter extraction and classification of three cortical neuron types reveals two distinct adaptation mechanisms." In: *Journal of neurophysiology* 107.6, pp. 1756–1775 (cit. on p. 14).

Miller, G.A. (1960). *Plans and the structure of behavior*. A Holt - Dryden book. Holt. ISBN: 9780030100758. URL: https://books.google.at/books?id=3UgAAAAAMAAJ (cit. on p. 2).

Mongillo, Gianluigi, Omri Barak, and Misha Tsodyks (2008). "Synaptic theory of working memory." In: *Science* 319.5869, pp. 1543–1546 (cit. on pp. 34, 74, 77).

Morcos, Ari S and Christopher D Harvey (2016). "History-dependent variability in population dynamics during evidence accumulation in cortex." In: *Nat. Neuro.* (cit. on pp. 92, 93).

Morris, Richard (1984). "Developments of a water-maze procedure for studying spatial learning in the rat." In: *Journal of neuroscience methods* 11.1, pp. 47–60 (cit. on p. 31).

Murray, John D et al. (2014). "A hierarchy of intrinsic timescales across primate cortex." In: *Nature neuroscience* 17.12, p. 1661 (cit. on p. 65).

Nicola, Wilten and Claudia Clopath (2017). "Supervised learning in spiking neural networks with FORCE training." In: *Nature communications* 8.1, p. 2208 (cit. on p. 20).

Nøkland, Arild (2016). "Direct feedback alignment provides learning in deep neural networks." In: *NIPS*, pp. 1037–1045 (cit. on pp. 89, 95).

O'Reilly, Randall C and Michael J Frank (2006). "Making working memory work: a computational model of learning in the prefrontal cortex and basal ganglia." In: *Neural computation* 18.2, pp. 283–328 (cit. on pp. 56, 57).

Perich, Matthew G, Juan A Gallego, and Lee E Miller (2018). "A neural population mechanism for rapid learning." In: *Neuron* (cit. on p. 25).

Pozzorini, Christian, Skander Mensi, et al. (2015). "Automated high-throughput characterization of single neurons by means of simplified spiking models." In: *PLoS computational biology* 11.6 (cit. on pp. 13, 14, 65).

Pozzorini, Christian, Richard Naud, et al. (2013). "Temporal whitening by power-law adaptation in neocortical neurons." In: *Nature neuroscience* 16.7, p. 942 (cit. on pp. 13, 14, 65).

Qiao, Ning et al. (2015). "A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses." In: *Frontiers in neuroscience* 9, p. 141 (cit. on p. 20).

Rajan, Kanaka and L. F. Abbott (2006). "Eigenvalue spectra of random matrices for neural networks." In: *Physical review letters* 97.18, p. 188104. URL: https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.97.188104 (visited on 06/08/2017) (cit. on p. 35).

Reddi, Sashank J, Satyen Kale, and Sanjiv Kumar (2018). "On the convergence of adam and beyond." In: *International Conference on Learning Representations* (cit. on p. 42).

Roeper, J. (2013). "Dissecting the diversity of midbrain dopamine neurons." In: *Trends in neurosciences* 36.6, pp. 336–342 (cit. on p. 86).

S. Goldman-Rakic, Patricia (Oct. 1992). "Working Memory and the Mind." In: 267, pp. 110–7 (cit. on p. 2).

Sajad, Amirsaman, David C. Godlove, and Jeffrey D. Schall (Feb. 2019). "Cortical microcircuitry of performance monitoring." En. In: *Nature Neu-*

roscience 22.2, p. 265. ISSN: 1546-1726. DOI: 10.1038/s41593-018-0309-8. URL: https://www.nature.com/articles/s41593-018-0309-8 (cit. on p. 86).

Samadi, Arash, Timothy P Lillicrap, and Douglas B Tweed (2017). "Deep learning with dynamic spiking neurons and fixed feedback weights." In: *Neural computation* 29.3, pp. 578–602 (cit. on pp. 89, 95).

Sanhueza, Magdalena and John Lisman (2013). "The CaMKII/NMDAR complex as a molecular memory." In: *Molecular brain* 6.1, p. 10 (cit. on p. 86).

Schemmel, Johannes et al. (2010). "A wafer-scale neuromorphic hardware system for large-scale neural modeling." In: *Circuits and systems (ISCAS), proceedings of 2010 IEEE international symposium on*. IEEE, pp. 1947–1950 (cit. on p. 20).

Schulman, John et al. (2017). "Proximal policy optimization algorithms." In: *arXiv preprint arXiv:1707.06347* (cit. on pp. 32, 44).

Stokes, Mark G. (2015). "'Activity-silent' working memory in prefrontal cortex: a dynamic coding framework." In: *Trends in Cognitive Sciences* 19.7, pp. 394–405. (Visited on 04/26/2017) (cit. on p. 34).

Subramoney, Anand et al. (2018). "Recurrent networks of spiking neurons learn to learn; in preparation." In: (cit. on pp. 25, 34).

Sussillo, David and LF Abbott (2014). "Random walk initialization for training very deep feedforward networks." In: *arXiv preprint arXiv:1412.6558* (cit. on p. 35).

Tartaglia, Elisa M, Gianluigi Mongillo, and Nicolas Brunel (2015). "On the relationship between persistent delay activity, repetition enhancement and priming." In: *Frontiers in psychology* 5, p. 1590 (cit. on p. 77).

Teeter, Corinne et al. (2018). "Generalized leaky integrate-and-fire models classify multiple neuron types." In: *Nature communications* 9.1, pp. 1–15 (cit. on p. 13).

Tsao, A. et al. (2018). "Integrating time from experience in the lateral entorhinal cortex." In: *Nature* 561.7721, pp. 57–52 (cit. on p. 62).

Turrigiano, GINA G, E Marder, and LF Abbott (1996). "Cellular short-term memory from a slow potassium conductance." In: *Journal of neurophysiology* 75.2, pp. 963–966 (cit. on p. 13).

Vasilaki, Eleni et al. (2009). "Spike-based reinforcement learning in continuous state and action space: when policy gradient methods fail." In: *PLoS computational biology* 5.12, e1000586 (cit. on p. 31).

Wang, Jane X, Zeb Kurth-Nelson, Dharshan Kumaran, et al. (2018). "Prefrontal cortex as a meta-reinforcement learning system." In: *Nature Neuroscience* (cit. on pp. 19, 25, 30).

Wang, Jane X, Zeb Kurth-Nelson, Dhruva Tirumala, et al. (2016). "Learning to reinforcement learn." In: *arXiv preprint arXiv:1611.05763* (cit. on pp. 19, 25, 26, 30).

Wang, Xiao-Jing (1998). "Calcium coding and adaptive temporal computation in cortical pyramidal neurons." In: *Journal of Neurophysiology* (cit. on p. 13).

Wang, Yun et al. (2006). "Heterogeneity in the pyramidal network of the medial prefrontal cortex." In: *Nature neuroscience* 9.4, p. 534 (cit. on pp. 74, 78, 79).

Warden, Pete (2018). "Speech commands: A dataset for limited-vocabulary speech recognition." In: *arXiv preprint arXiv:1804.03209* (cit. on pp. 55, 103).

Wasmuht, Dante Francisco et al. (2018). "Intrinsic neuronal dynamics predict distinct functional roles during working memory." In: *Nature communications* 9.1, p. 3499 (cit. on pp. 65, 101, 106).

Werbos, P. J. (Oct. 1990). "Backpropagation through time: what it does and how to do it." In: *Proceedings of the IEEE* 78.10, pp. 1550–1560. ISSN: 0018-9219. DOI: 10.1109/5.58337 (cit. on p. 15).

Williams, Ronald J. and David Zipser (1989). "A learning algorithm for continually running fully recurrent neural networks." In: *Neural computation* 1.2, pp. 270–280. URL: http://www.mitpressjournals.org/doi/abs/10.1162/neco.1989.1.2.270 (visited on 10/22/2015) (cit. on p. 86).

Winters, B. D., L. M. Saksida, and T. J. Bussey (2008). "Object recognition memory: neurobiological mechanisms of encoding, consolidation and retrieval." In: *Neuroscience & Biobehavioral Reviews* 32.5, pp. 1055–1070 (cit. on p. 64).

Wolff, Michael J et al. (2017). "Dynamic hidden states underlying working-memory-guided behavior." In: *Nature Neuroscience* 20.6, p. 864 (cit. on pp. 53, 64, 68).

Yagishita, Sho et al. (2014). "A critical time window for dopamine actions on the structural plasticity of dendritic spines." In: *Science* 345.6204, pp. 1616–1620 (cit. on p. 86).

Zenke, Friedemann and Tim P Vogels (2020). "The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks." In: *BioRxiv* (cit. on p. 103).