university of groningen

faculty of science and engineering

INCLUDING STDP TO ELIGIBILITY PROPAGATION IN MULTI-LAYER RECURRENT SPIKING NEURAL NETWORKS

WERNER VAN DER VEEN

MSc. Artificial Intelligence
Faculty of Science and Engineering
University of Groningen

Supervised by Dr. Herbert Jaeger & Dr. Marco Wiering

May 3, 2021

ABSTRACT

Spiking neural networks (SNNs) in neuromorphic systems are more energy efficient compared to deep learning–based methods, but there is no clear competitive learning algorithm for training such SNNs. Eligibility propagation (e-prop) offers an efficient and biologically plausible way to train competitive recurrent SNNs in low-power neuromorphic hardware. In this report, previous performance of e-prop on a speech classification task is reproduced, and the effects of including STDP-like behavior are analyzed. Including STDP to the ALIF neuron model improves the classification performance, but this is not the case for the Izhikevich e-prop neuron. Finally, it was found that e-prop implemented in a single-layer recurrent SNN consistently outperforms a multi-layer variant.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

The human brain is one of the most complex systems in the universe. Its approximately 86 billion neurons (Azevedo et al., 2009) and 100–500 trillion synapses (Drachman, 2005) are capable of abstract reasoning, pattern recognition, memorization, and sensory experience—while consuming only about 20 W of power (Drubach, 2000; Sokoloff, 1960).

An early goal of artificial intelligence has been to construct systems that exhibit similar intelligent traits (Turing, 1948). One of the proposed methods was to emulate the network of biological neurons in the human brain using simple units called perceptrons (Rosenblatt, 1958). These perceptrons were inspired by Hebbian learning (Hebb, 1949), which was a new (and later corroborated) theory on biological learning. After the popularization in the 1980s of trainable Hopfield networks (Hopfield, 1982) and backpropagation (Rumelhart, G. E. Hinton, and Williams, 1986), which enabled learning linearly inseparable tasks, artificial neural networks (ANNs) and the connectionist approach were embraced with a new appreciation.

These ANNs are networks of small computational units that can be trained to perform specific pattern recognition tasks. Backpropagation has proven to work well in training ANNs with multiple layers, most popularly in the field of deep learning (DL), which has become a dominant field in artificial intelligence. This popularity is partly due to exponentially increasing computing power and data storage capabilities, as well as the rise of the Internet, which has also provided ample training data. Some variations on ANNs have shown to improve learning performance, such as using convolutional (CNN) and recurrent (RNN) neural networks, both of which are, like the perceptron, inspired by the architecture of the human brain (Fukushima and Miyake, 1982; Hubel and Wiesel, 1968; LeCun, Bengio, et al., 1995; Lukoševičius and Jaeger, 2009). These types of networks approach or exceed human level performance in some areas (Schmidhuber, 2015).

ENERGY LIMITS However, DL-based methods are starting to show diminishing returns; training some state-of-the-art models can require so much data and computing power that only a small number of organizations has the resources to train and deploy them. For example, one of the current top submissions of the "Labeled Faces in the Wild" face verification task is a deep ANN by Paravision that was trained using a dataset of 10 million face images of 100 thousand individuals[1]. Beside very large datasets, deep ANNs also require a significant amount of power to train. For instance, ResNet (Kaiming He et al., 2016) has been trained for 3 weeks on a 8-GPU server consuming about 1 GWh. This

1 See `http://vis-www.cs.umass.edu/lfw/results.html#everai`. Last accessed January 2021.

high power consumption precludes computations in mobile low-power or small-scale devices, which now require at least a connection to a cloud computing server.

The energy consumption of DL contrasts strongly with that of the human brain, which can learn patterns using far less energy and data. This is because despite the biologically inspired foundation, deep ANNs are fundamentally different from the brain, which is an inherently time-dependent dynamical system (Sacramento et al., 2018; Woźniak et al., 2020) that relies on biophysical processes, recurrence, and feedback of its physical substrate for computation (Bhalla, 2014; Sterling and Laughlin, 2015). Deep ANNs are implemented on von Neumann architectures (Von Neumann, 1993), i. e., a system with a central processing unit (CPU) and separate memory, which are significantly different from the working model of the brain (Schuman et al., 2017).

One reason for the inefficiency of deep ANNs is that their implementations suffer from the von Neumann bottleneck (Zenke and E. O. Neftci, 2021), which involves a limited throughput between the CPU and memory—a data operation cannot physically co-occur with fetching instructions to process that data because they share the same communication system. Parallelization on GPUs has alleviated this bottleneck to some extent, but the human brain is more efficient as it is embedded in a physical substrate whose neurons operate and communicate fully in parallel (A Pastur-Romay et al., 2017) using sparsely occurring *spikes* (Bear, Connors, and Paradiso, 2020), and where no explicit data processing instructions exist. A spike can be represented as a binary value which causes the synapse to increase the membrane potential in the efferent neuron to change by a fixed value (Bear, Connors, and Paradiso, 2020). Connections in ANNs are represented abstractly by large weight matrices, which are all multiplied with neuron activation values at every propagation cycle. In the brain, a synapse spikes sparsely and thereby saves energy while conveniently including an informative temporal component.

A second reason is that backpropagation requires two passes over the ANN: the first to compute the network output given an input, and the second to propagate the output error back into the network to move the weights between neurons in the direction of the negative gradient. Backpropagation in RNNs is often performed by unrolling the network in a feedforward ANN in a process called backpropagation through time (BPTT). The human brain, in contrast, is unlikely to use backpropagation, BPTT, or gradients of the output error (Lillicrap and Santoro, 2019).

SPIKING NEURAL NETWORKS Spiking neural networks (SNNs) (Gerstner and Kistler, 2002; Maass, 1997) are another step towards biological plausibility of connectionist models. SNNs use neurons that do not relay continuous activation values at every propagation cycle, but spike once when they reach a threshold value. This makes SNNs potentially much more energy efficient than ANNs that use backpropagation. SNNs are competitive to ANNs in terms of accuracy and computational

power, as well in their ability to display precise spike timings (Lobo et al., 2020). Their sparse firing regimes also offer improved interpretability of their behavior as compared to traditional ANNs (Soltic and Kasabov, 2010), which is desired in areas such as medicine or aviation.

However, SNNs have not been as popular as ANNs. One reason for this is that spike-based activation is not differentiable. As a consequence, backpropagation cannot be directly used to move in the negative direction of the error gradient, although some attempts have been made to bridge this divide (Bellec, Scherr, Hajek, et al., 2019; Bohte, Kok, and La Poutre, 2002; Hong et al., 2010; J. H. Lee, Delbruck, and Pfeiffer, 2016; Ourdighi and Benyettou, 2016; Sacramento et al., 2018; Whittington and Bogacz, 2019; Y. Xu et al., 2013) and to make backpropagation more biologically plausible.

Similarly, it has been demonstrated that approximations of BPTT can be applied to recurrent SNNs (RSNNs) (Bellec, Salaj, et al., 2018; Huh and Sejnowski, 2017). Both single- and multi-layer SNNs have shown good performance in visual processing (Escobar et al., 2009; Kheradpisheh et al., 2018; D. Liu and Yue, 2017) and speech recognition (Dong, Xuhui Huang, and B. Xu, 2018; Tavanaei and Maida, 2017). However, none of these algorithms are biologically plausible. While DL was rapidly becoming popular during the 2010s, there was no clear learning algorithm for SNNs that could compete with ANNs. A second reason for the relative unpopularity of SNNs is that they are generally emulated in von Neumann architectures, undermining their energy efficiency advantages.

NEUROMORPHIC COMPUTING SNN learning algorithms are particularly useful in the upcoming field of neuromorphic computing (NC) (Mitra, Fusi, and Indiveri, 2008), in which analog very-large-scale integration (VLSI) systems are used to implement neural systems. On the surface, it can be understood as running neural networks not abstracted in a digital system, but physically embedded in a dedicated analog medium. A central advantage of NC is energy efficiency (Hasler and Akers, 1990; J.-C. Lee and Sheu, 1990; Tarassenko et al., 1990). This energy efficiency, combined with NC's massive parallelism (Monroe, 2014), makes VLSIs particularly relevant for implementing SNNs.

Like SNNs, neuromorphic systems typically use sparse, event-based communication between devices and physically colocalized memory and computation (E. O. Neftci, 2018; Sterling and Laughlin, 2015). Although colocalized memory and computation has also been implemented in digital machines, such as Google's TPU[2], Graphcore's IPU[3], or Cerebras' CS-1[4], neuromorphic systems are more efficient for running ANNs (Merolla et al., 2014; Rajendran et al., 2019). The energy consumption of CMOS artificial neurons is several orders of magnitude lower than that of neurons in an ANN, and even 2–3 times lower than the energy consumption of biological neurons (Elbez et al., 2020), offering a possible escape from the increasing energy costs of DL models.

2 See `https://cloud.google.com/tpu/docs/tpus`. Last accessed January 2021
3 See `https://www.graphcore.ai/products/ipu`. Last accessed January 2021
4 See `https://cerebras.net/product/#chip`. Last accessed January 2021

Because of this massive parallelism, high energy efficiency, and good ability to implement cognitive functions, neuromorphic systems are attracting strong interest. In particular, SNNs emerged as an ideal biologically inspired NC paradigm for realizing energy-efficient on-chip intelligence hardware (Davies et al., 2018; Merolla et al., 2014), suitable for running fast and complex SNNs on low-power devices. For instance, a competitive image classification performance was reached with a 6-order of magnitude speedup in a leaky integrate-and-fire (LIF) SNN in field-programmable gate arrays, compared to digital simulations (G. Zhang et al., 2020).

BIOLOGICAL LEARNING To run an SNN on neuromorphic hardware, a *local* and *online* learning algorithm is needed. The precondition of locality refers to the idea that a neuron or synapse can only access information or communication with which it is physically connected. For instance, the inner state of a neuron can only be influenced by itself, or by the spikes it receives from afferent neurons. Similarly, a synapse can only spike or change its weight based on signals from the afferent and efferent neuron. This is a direct consequence of the colocalization of processing and memory. The precondition of being online can be regarded as temporal locality—neurons and synapses can only access information that physically exists at the same point in time. They cannot access information about past or future events, except if explicit memory traces of a past event are retained over time. In that case, past events can affect the neuron's current behavior.

The brain also adheres to these two constraints. Some of the more common learning rules in ANNs are based on a form of Hebbian learning, which is a major factor in biological learning and memory consolidation (Schuman et al., 2017). Classical Hebbian learning is often summarized by "cells that fire together, wire together", if there is a causal relationship between these cells, such as a postsynaptic potential on a connecting synapse. Direct application of Hebbian learning in a spiking neural network will generally lead to a positive feedback loop, because "wiring cells together", or increasing the synaptic strength, will in turn increase the likelihood that they also fire together (Zenke, Gerstner, and Ganguli, 2017). Furthermore, classical Hebbian learning describes no way for a synapse to weaken.

Spike-timing-dependent plasticity (STDP) (Abbott and Nelson, 2000; Caporale and Dan, 2008) is a type of Hebbian learning that incorporates temporal causality on a synapse from neuron A to neuron B: if B spikes right after neuron A, then the synapse is strengthened, but if B spikes right before A, it is weakened. It is widely known that STDP is a fundamental learning principle in the human brain (Caporale and Dan, 2008; Kandel et al., 2000), including perceptual systems in the sensory cortex (S. Huang et al., 2014). STDP by itself can be used as an unsupervised learning algorithm or to forming associations in classical conditioning (Diehl and Cook, 2015; C.-H. Kim et al., 2018). Furthermore, it has been demonstrated to form associations between memory traces in SNNs, which are crucial for cognitive brain function (Pokorny

et al., 2020). To allow supervised learning, or operant conditioning, a learning signal is required to influence the direction of the synapse weight change: a positive learning signal will reinforce the association (long-term potentiation), and a negative learning signal weakens it (long term depression) (Lobov et al., 2020). STDP with a learning signal is known as reward-modulated STDP (R-STDP) (Legenstein, Pecevski, and Maass, 2008) in the field of SNNs and three-factor Hebbian learning in neuroscience (Frémaux and Gerstner, 2016). Three-factor Hebbian learning has been demonstrated to outperform its classical two-factor counterpart in a localization-and-retrieval task (Porr and Wörgötter, 2007). A possible reason for this performance difference is that modulatory signals "may provide the attentional and motivational significance for long-term storage of a memory in the brain" and stabilize classical Hebbian learning (Bailey et al., 2000).

Neurotransmitters are used to modulate the learning signal in the brain. Dopamine, for instance, which has a central behavioral and functional role in the primary motor cortex (Barnes et al., 2005; Dang et al., 2006), has been shown to modulate synapses through dendritic spine enlargement during a very narrow time window (Dang et al., 2006). It is behaviorally related to novelty and reward prediction (S. Li et al., 2003; Schultz, 2007) by gating neuroplasticity of corticostriatal (Reynolds, Hyland, and Wickens, 2001; Reynolds and Wickens, 2002) and ventral tegmental (VTA) synapses (Bao, Chan, and Merzenich, 2001). In the VTA, dopaminergic neurons respond to learning signals in a highly localized manner that is specific for local populations of neurons (Engelhard et al., 2019). This is also the case in other areas of the midbrain (Roeper, 2013).

However, R-STDP by itself does not solve the credit assignment problem, which relates to neuromodulation of synapses after a learning signal is presented with some delay. In that case, when the learning signal is presented, the neurons have long spiked, and it is not clear which synapses elicited the behavior that is rewarded or punished. Recent research suggests that the brain uses *eligibility traces* (Florian, 2007; Izhikevich, 2007) to solve the credit assignment problem (Gerstner, Lehmann, et al., 2018; Stolyarova, 2018). In particular, the synaptic CaMKII protein complex is activated during the induction of long-term potentiation (LTP) of biological synapses if the presynaptic neuron spikes shortly before a postsynaptic neuron (Sanhueza and Lisman, 2013). This LTP is maintained over behavioral time spans, and gradually fades. When followed by a learning signal in the form of a neurotransmitter, synaptic plasticity is induced (Cassenaer and Laurent, 2012; Gerstner, Lehmann, et al., 2018; Yagishita et al., 2014).

Over the past decade, eligibility traces have been researched in the context of a wide range of topics, such as biological learning, spiking neural networks, and neuromorphic computing. Synaptic plasticity was demonstrated using eligibility traces in deep feedforward SNNs (Kaiser, Mostafa, and E. Neftci, 2020; E. O. Neftci et al., 2017; Zenke and Ganguli, 2018) and could be implemented in feedforward VLSIs. In Zenke and Ganguli (2018) it is asserted that these methods are also

applicable for RSNNs. Eligibility traces have also been shown to solve difficult credit assignment problems in SNNs using R-STDP (Bellec, Scherr, Subramoney, et al., 2020; Legenstein, Pecevski, and Maass, 2008) and in RNNs (Kaiwen He et al., 2015), and have a predictable learning effect (Legenstein, Pecevski, and Maass, 2008).

ELIGIBILITY PROPAGATION Eligibility propagation (e-prop) (Bellec, Scherr, Subramoney, et al., 2020) is a local and online learning algorithm for RSNNs that can be mathematically derived as an approximation to BPTT (see also Section 2.4). The main aspect that distinguishes e-prop from other eligibility trace–based algorithms is that the particular computation of the eligibility trace depends on multiple hidden states of a neuron. The property that a neuron can have multiple hidden states means that there are many types of neuron models that can be used in e-prop.

In e-prop, the learning signal is a local variation on random broadcast alignment, which propagates the error directly back onto the neurons with a random weight, resembling the function of a neuromodulator in the brain. This has been suggested to provide a diversity of feature detectors for task-relevant network inputs (Bellec, Scherr, Subramoney, et al., 2020). This form of broadcast alignment can perform as effectively as backpropagation in some tasks in feedforward ANNs (Lillicrap, Cownden, et al., 2016; Nøkland, 2016) and multi-layer SNNs (Clopath et al., 2010; Samadi, Lillicrap, and Tweed, 2017), but performs poorly in deep feedforward ANNs for complex image classification tasks (Bartunov et al., 2018).

The local and online properties of e-prop make it a biologically plausible learning algorithm that can be implemented on VLSIs. E-prop has been demonstrated to work for a large variety of tasks, including classifying phones (i. e., speech sounds), for which it performs competitively with RNNs that use BPTT and the popular LSTM neuron model (Graves, Mohamed, and G. Hinton, 2013).

The fading eligibility trace in e-prop is similar to STDP in that the weight change is smaller if there is a longer delay between a presynaptic and postsynaptic spike. However, e-prop is essentially independent of STDP, because it does not explicitly relate the order of the pre- and postsynaptic spike to the synaptic weight update. However, in Bellec, Scherr, Subramoney, et al. (2020) e-prop is remarked to start showing STDP-like properties if the synaptic delay of a spike is prolonged.

So far, only the LIF and adaptive LIF (ALIF) neuron models have been used in e-prop, which do not show STDP-like properties by default. In Traub et al. (2020), a functional modification was made to the LIF model such that STDP can occur. In particular, STDP occurs when the neuron model provides a negated gradient signal in the case when a presynaptic signal arrives too late. This resembles the biological phenomenon of error-related negativity (ERN) (Nieuwenhuis et al., 2001), which is a negative brain response that immediately follows an erroneous behavioral response and peaks after 80–150 ms with an amplitude that depends on the intent and motivation of a person. Traub et al. (2020) also showed

this effect for the Izhikevich neuron (Izhikevich, 2003). However, these STDP-modified neurons were shown only in a single-synapse demo to illustrate the STDP properties, not in a full learning task.

MULTI-LAYER RSNNS The discovery of backpropagation allowed gradient descent–based training of multi-layer ANNs, which significantly increased their performance. Although it is unlikely to use backpropagation, the human brain is hierarchically structured such that early layers process simple information and deep layers process more abstract information. Similarly, multi-layer CNNs also show higher levels of abstraction in deeper layers of the network. For instance, early convolutional filters identify lines and edges, while deeper filters identify more complex shapes. In RNNs, stacking recurrent layers results in a similar abstraction—but it is temporal instead of spatial (Gallicchio, Micheli, and Pedrelli, 2017; Hermans and Schrauwen, 2013). Deeper RNN layers exhibit slower time dynamics and longer memory spans than shallow layers (Gallicchio, 2018), suggesting that they ignore small variations in the input signal and integrate larger temporal patterns. It is unclear if these findings extrapolate to RSNNs.

RESEARCH OBJECTIVES State-of-the-art SNN learning algorithms perform well on a variety of tasks, but have so far not shown the efficiency and learning performance of the human brain. SNNs are most efficient when embedded in a neuromorphic system, requiring a learning algorithm that is local and online. E-prop is an example of such an algorithm, but it has not yet been used in conjunction with neuron models other than LIF and ALIF. These neuron models do not show STDP-like behavior, which is a fundamental learning principle in the brain, and has a close connection to biological eligibility traces, and may therefore improve the accuracy and efficiency of e-prop. For this reason, in this research I continue the trend of emulating biological processes by for the first time modifying the e-prop network to use neuron models that show STDP-like behavior. Analyzing the effects of including STDP-like behavior to the neuron models in an e-prop network is the primary research objective in this report.

Two neuron models that display STDP are used. The first model is the ALIF-STDP, which is a new crossover neuron model of the ALIF neuron (used in Bellec, Scherr, Subramoney, et al. (2020)) and the STDP-LIF neuron (derived but not verified for e-prop in Traub et al. (2020)). The second STDP-like neuron model is the Izhikevich neuron model, which was also derived in Traub et al. (2020), and is slightly modified in this research to produce stable eligibility traces over time.

So far, only the performance of e-prop models with a single fully-connected pool of neurons has been described. Whereas multi-layered CNNs and RNNs can sometimes process abstract information more easily, it is not clear if this also holds for SNNs or e-prop models. The secondary research objective is analyzing the effects of a multi-layered e-prop architecture.

STRUCTURE OF THIS REPORT In the remainder of this report, the e-prop framework is described in Chapter 2. Then, Chapter 3 describes the method used to implement the TIMIT phone classification task and modify the e-prop algorithm to a multi-layer framework with different neuron models, particularly the STDP-ALIF and Izhikevich models. This modified e-prop framework is implemented and experimentally verified. The results are presented and discussed in Chapter 4. These results show that including STDP in ALIF neuron models can indeed improve the learning performance and leads to a higher classification accuracy. However, this does not hold for the Izhikevich neuron, suggesting that this neuron model is not suited for e-prop in its current form. Furthermore, the use of multiple stacked recurrent layers slows down the learning speed, and so does not provide an efficient e-prop architecture. Finally, Chapter 5 summarizes and concludes this report.

2 THEORETICAL FRAMEWORK

This chapter describes the theoretical framework of eligibility propagation expounded in previous literature, which is then developed further in Chapter 3.

2.1 ELIGIBILITY PROPAGATION MODEL

In Bellec, Scherr, Subramoney, et al. (2020), an eligibility propagation (e-prop) model $\mathcal{M}$ of a neuron j in a feedforward or recurrent network is defined by a tuple $\langle M, f \rangle$, where M is a function

$$\mathbf{h}_j^t = M\left(\mathbf{h}_j^{t-1}, \mathbf{z}^{t-1}, \mathbf{x}^t, \mathbf{W}_j\right) \tag{2.1}$$

that defines the hidden state $\mathbf{h}_j^t$ at a discrete time step t, where $\mathbf{z}^{t-1}$ is the observable state of all neurons at the previous time step (i. e., the binary spike values), $\mathbf{x}^t$ is the model input vector at time t, and $\mathbf{W}_j$ is the weight vector of afferent (i. e., "incoming") synapses. The hidden state of a neuron contains all variables that are used for a specific neuron type, e. g., an activation value, or a variable that models a neuron's refractory period after a spike. In short, Equation 2.1 indicates that the hidden state is affected primarily by spikes of other neurons $\mathbf{z}^{t-1}$, and the current input to the model $\mathbf{x}^t$, which are both weighted by trainable network weights $\mathbf{W}_j^{\text{rec}} \subset \mathbf{W}_j$ and $\mathbf{W}_j^{\text{in}} \subset \mathbf{W}_j$, respectively.

The function f in $\mathcal{M}$ describes the update of the observable state of a neuron j at time t:

$$z_j^t = f\left(\mathbf{h}_j^t\right), \tag{2.2}$$

such that the spikes elicited by a neuron only depend on its hidden state. For instance, a neuron j may spike at time t (i. e., $z_j^t = 1$) if its activity, which is contained in the hidden state, reaches a threshold value.

The purpose of an e-prop model is that it can be trained to perform a learning task, such as classification. As described in the remainder of this chapter, the weight matrix $\mathbf{W}$, which comprises the weight values of all synapses in the model, is trained such that the input vectors $\mathbf{x}^t$ yield a good prediction of the classes they belongs to.

The formalizations described in Equations 2.1 and 2.2 indicate that e-prop is a *local* training method, because a neuron's observable state depends only on its own hidden state, which in turn depends only on observable signals that are directly connected to it. E-prop is also an *online* training method, because both the hidden and observable state of a neuron depend only on information that is still available; the observable state is updated at the same time step as the hidden state, and the hidden state is updated according to information which is then present in the afferent neurons.

The precise formulations of M and f depend on the neuron models that are used in the e-prop model.

2.2 NEURON MODELS

LIF NEURON In Bellec, Scherr, Subramoney, et al. (2020), the LIF neuron model is formulated in the context of e-prop, along with a variant (viz. ALIF) that has an adaptive threshold based on the neuron's spiking frequency. The observable state of a LIF model is given by

$$z_j^t = H\left(v_j^t - v_{\text{th}}\right), \tag{2.3}$$

where H is the Heaviside step function, v_j^t is the activity of neuron j at discrete time t, and v_{th} is the threshold constant. (Note that this and all other used hyperparameters are listed in Table A.2.) From Equation 2.3 it follows that a neuron spikes ($z_j^t = 1$) if its activity reaches the activity threshold, and remains silent ($z_j^t = 0$) otherwise. These spikes are the only communication between neurons in the e-prop model.

The hidden state h_j^t of a LIF neuron model contains only an activity value v_j^t that evolves over time according to the equation

$$v_j^{t+1} = \alpha v_j^t + \sum_{i \neq j} W_{ji}^{\text{rec}} z_i^t + \sum_i W_{ji}^{\text{in}} x_i^{t+1} - z_j^t v_{\text{th}}, \tag{2.4}$$

where W_{ji}^{rec} is a synapse weight from neuron i to neuron j, and α is a constant decay factor. In Equation 2.4, the first term models the decay of the activity value over time. The second and third term model the input of the neuron from other neurons, or from the input to the network, respectively. The fourth term ($-z_j^t v_{\text{th}}$) ensures that the activity of the neuron drops when it spikes. Furthermore, z_j^t is explicitly fixed to 0 for T^{refr} time steps after a spike to model neuronal refractoriness.

In biological neurons, the refractory period consists of an "absolute" phase, during which eliciting a new spike is impossible, and a subsequent "relative" phase, during which the threshold is temporarily increased (Purves, 2008). Clamping z_j^t to 0 emulates only this absolute phase, and is therefore only a crude approximation to model biological refractoriness. The refractory period is built into the equations of the Izhikevich neuron model described in Section 3.2.2.2, which is therefore arguably a more biologically plausible neuron model.

ALIF NEURON The ALIF neuron model introduces a threshold adaptation variable a_j^t to the hidden state of the LIF neuron, such that $\mathbf{h}_j^t \stackrel{\text{def}}{=} \left[v_j^t, a_j^t\right]$. In an ALIF neuron, the spiking threshold increases after a spike, and otherwise decreases back to a baseline threshold v_{th} in the continued absence of spikes.

This resembles *spike frequency adaptation* (SFA), a common feature of neocortical pyramidal neurons (Benda and Herz, 2003). SFA is a homeostatic control mechanism that affects the spiking frequency based on the recent spiking activity, such that neurons that spike relatively

infrequently become more sensitive, and vice versa. Ahmed et al. (1998) found that a single time constant is a good fit to characterize the threshold's exponential decay to a steady state.

The observable state of an ALIF neuron is therefore described by

$$z_j^t = H\left(v_j^t - v_{\text{th}} - \beta a_j^t\right) \tag{2.5}$$

and

$$a_j^{t+1} = \rho a_j^t + z_j^t, \tag{2.6}$$

where $0 < \rho < 1$ is an adaptation decay constant and $\beta \geq 0$ is an adaptation strength constant. Equation 2.6 indicates that the adaptive threshold increases at every spike, and decays back to v_{th} in the absence of spikes. The decay factor ρ of the threshold adaptation is higher than the decay factor α of the neuron activity, such that the immediate firing behavior of a neuron is affected on a shorter time scale than the threshold adaptation, which is better suited to reflect the working memory of a neuron and track longer temporal dependencies in the input data than the activation decay. The interaction between the neuron activity, adaptive threshold and spiking behavior is illustrated in Figure 2.1.

The LIF neuron is a spacial case of an ALIF neuron, for which $\beta = 0$, effectively canceling the effect of the threshold adaptation value a_j^t on the observable state z_j^t in Equation 2.5. Therefore, only the e-prop derivations for the ALIF neurons will be described in the following sections. From this point, references to LIF neurons in this report will refer to this special case.

2.3 NETWORK TOPOLOGY

The e-prop network structure as used in this report consists of the following main components:

1. An input layer $\mathbf{x}^t$.
2. A recurrent layer containing N neurons that are connected to all other neurons in this layer by weights $\mathbf{W}^{\text{rec}}$. This layer is also connected to the input layer by weights $\mathbf{W}^{\text{in}}$.
3. An output layer $\mathbf{y}^t$ connected to the recurrent layer by weights $\mathbf{W}^{\text{out}}$.

Since one of the goals of this report is to evaluate multi-layer topologies, the recurrent layer component is modified in Section 3.2.1 to support architectures with a feedforward series of recurrent layers.

An input vector $\mathbf{x}^t$ at time step t is fed to a pool of N recurrent neurons with hidden states $\mathbf{h}^t$ and observable states $\mathbf{z}^t$ through input weights $\mathbf{W}^{\text{in}}$. The recurrent weights $\mathbf{W}^{\text{rec}}$ connect neurons with each other, but no self-loops exist. Therefore, the recurrent neurons also receive inputs from the observable states of the afferent neurons. 25% of these neurons are LIF neurons (i. e., $\beta = 0$) and the others are ALIF

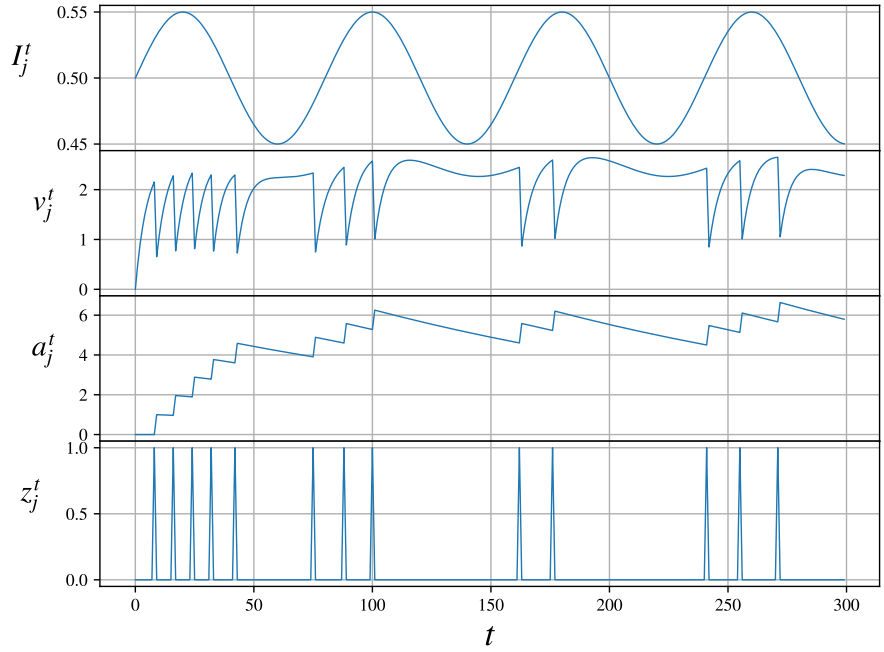


Figure 2.1: A simulated ALIF neuron j receives a sinusoidal input I for 300 time steps t. This figure illustrates the adaptive threshold a, which increases at every spike z, requiring a higher activity v for a next spike. When a spike occurs, v decreases by v_{th}. Note that the first wave of the sinusoid elicits a stronger spike train than subsequent waves, demonstrating the homeostatic effect of the adaptive threshold. Note also that on a short time scale, spikes tend to occur primarily in the upward phases of the sinusoid, suggesting that ALIF neurons are well-suited to respond to changes in the input signals.

neurons. The output weights $\mathbf{W}^{\text{out}}$ process the observable states of the neurons through a softmax function into a logits layer π^t. These logits are compared with the target output $\pi^{*,t}$ and multiplied with broadcast weights $\mathbf{B}^t$ to obtain a learning signal L_j^t for every neuron j in the pool. Figure 2.2 illustrates the basic architecture of a single-layer e-prop model.

Like in Bellec, Scherr, Subramoney, et al. (2020), weights are initialized by sampling them from a Gaussian distribution $\mathcal{N}\left(0, \sqrt{N}\right)$ where N is the number of afferent neurons. For instance, the weights $\mathbf{W}^{\text{in}}$ between the input and the first layer are sampled from $\mathcal{N}\left(0, \sqrt{39}\right)$ if there are 39 input features. Likewise, each of the neurons has $N-1$ afferent recurrent weights, so the recurrent weights within a layer are sampled from $\mathcal{N}\left(0, \sqrt{N-1}\right)$.

A randomly selected 80% of synaptic weights is then set to a value of 0, as well as the synapses that connect a neuron to itself, rendering them ineffective.

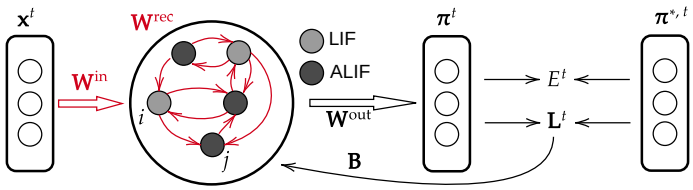


Figure 2.2: A basic illustration of a single-layer network architecture. An input vector $\mathbf{x}^t$ at time step t is fed to a pool of N recurrent neurons with hidden states $\mathbf{h}^t$ and observable states $\mathbf{z}^t$ through input weights $\mathbf{W}^{\text{in}}$. The recurrent weights $\mathbf{W}^{\text{rec}}$ connect neurons with each other, but no self-loops exist. A randomly selected 25% of these neurons is a LIF neuron (i. e., $\beta = 0$) and the others are ALIF neurons. The output weights $\mathbf{W}^{\text{out}}$ process the observable states of the neurons through a softmax function into a logits layer π^t. These logits are compared with the target output $\pi^{*,t}$ and multiplied with broadcast weights $\mathbf{B}^t$ to obtain a learning signal L_j^t for every neuron j in the pool. Note that weights illustrated in red are e-prop weights, i. e., they track eligibility traces.

2.4 DERIVING E-PROP FROM RNNS

Eligibility propagation is a local and online training method that can be derived from backpropagation through time (BPTT). In BPTT, an RNN is unfolded in time, such that the backpropagation method used in feedforward neural networks can be applied to compute the gradients of the cost with respect to the network weights.

In this section, the main equation of e-prop

$$\frac{dE}{dW_{ji}} = \sum_t \frac{dE}{dz_j^t} \cdot \left[\frac{dz_j^t}{dW_{ji}}\right]_{\text{local}}, \tag{2.7}$$

where $\cdot$ denotes the dot product, is derived from the classical factorization of the loss gradients in an unfolded RNN as in Bellec, Scherr, Subramoney, et al. (2020):

$$\frac{dE}{dW_{ji}} = \sum_{t' \leq T} \frac{dE}{d\mathbf{h}_j^{t'}} \cdot \frac{\partial \mathbf{h}_j^{t'}}{\partial W_{ji}}, \tag{2.8}$$

where summation indicates that weights are shared. Recall that for ALIF neurons, $\mathbf{h}_j^t \stackrel{\text{def}}{=} \left[v_j^t, a_j^t\right]$. This is also the true for ALIF neurons that use $\beta = 0$ to disable their threshold adaptability.

By applying the chain rule, the first factor $\frac{dE}{d\mathbf{h}_j^{t'}}$ can be decomposed into a series of learning signals $L_j^t = \frac{dE}{dz_j^t}$ and local factors $\frac{\partial \mathbf{h}_j^{t-t'}}{\partial \mathbf{h}_j^t}$ for all

t starting from the event horizon t', which is the oldest time step that information is used from:

$$\frac{dE}{d\mathbf{h}_j^{t'}} = \underbrace{\frac{dE}{dz_j^{t'}}}_{L_j^{t'}} \frac{\partial z_j^{t'}}{\partial \mathbf{h}_j^{t'}} + \frac{dE}{d\mathbf{h}_j^{t'+1}} \frac{\partial \mathbf{h}_j^{t'+1}}{\partial \mathbf{h}_j^{t'}}. \tag{2.9}$$

Note that this equation is recursive. If Equation 2.9 is substituted into the classical factorization (Equation 2.8), the full history of the synapse $i \to j$ is integrated, and a recursive expansion is obtained that has $\frac{dE}{d\mathbf{h}_j^{T+1}}$ as its terminating case:

$$\frac{dE}{dW_{ji}} = \sum_{t'} \left(L_j^{t'} \frac{\partial z_j^{t'}}{\partial \mathbf{h}_j^{t'}} + \frac{dE}{d\mathbf{h}_j^{t'+1}} \frac{\partial \mathbf{h}_j^{t'+1}}{\partial \mathbf{h}_j^{t'}} \right) \cdot \frac{\partial \mathbf{h}_j^{t'}}{\partial W_{ji}} \tag{2.10}$$

$$= \sum_{t'} \left(L_j^{t'} \frac{\partial z_j^{t'}}{\partial \mathbf{h}_j^{t'}} + \left(L_j^{t'+1} \frac{\partial z_j^{t'+1}}{\partial \mathbf{h}_j^{t'+1}} + (\cdots) \frac{\partial \mathbf{h}_j^{t'+2}}{\partial \mathbf{h}_j^{t'+1}} \right) \frac{\partial \mathbf{h}_j^{t'+1}}{\partial \mathbf{h}_j^{t'}} \right) \cdot \frac{\partial \mathbf{h}_j^{t'}}{\partial W_{ji}}. \tag{2.11}$$

The recursive parenthesized factor can be written into a second factor indexed by t:

$$\frac{dE}{dW_{ji}} = \sum_{t'} \sum_{t \geq t'} L_j^t \frac{\partial z_j^t}{\partial \mathbf{h}_j^t} \frac{\partial \mathbf{h}_j^t}{\partial \mathbf{h}_j^{t-1}} \cdots \frac{\partial \mathbf{h}_j^{t+1}}{\partial \mathbf{h}_j^{t'}} \cdot \frac{\partial \mathbf{h}_j^{t'}}{\partial W_{ji}}. \tag{2.12}$$

By exchanging the summation indices, the learning signal L_j^t is pulled out from the inner summation.

Within the inner summation, the terms $\frac{\partial \mathbf{h}_j^{t+1}}{\partial \mathbf{h}_j^t}$ are collected in an *eligibility vector* $\boldsymbol{\epsilon}_{ji}^t$ and multiplied with the learning signal L_j^t at every time step t. This is crucial for understanding why e-prop is an online training method—local gradients are computed based on traces that are directly accessible at the current time step t, and the eligibility vector operates as a recursively updated "memory" to track previous local hidden state derivatives:

$$\boldsymbol{\epsilon}_{ji}^t = \frac{\partial \mathbf{h}_j^t}{\partial \mathbf{h}_j^{t-1}} \cdot \boldsymbol{\epsilon}_{ji}^{t-1} + \frac{\partial \mathbf{h}_j^t}{\partial W_{ji}}. \tag{2.13}$$

This is why the ρ and α parameters, which define the decay rate in hidden states and the corresponding eligibility vectors, should be set according to the required working memory in the learning task. The eligibility vector and the hidden state have the same dimension: $\left\{ \boldsymbol{\epsilon}_{ji}^t, \mathbf{h}_j^t \right\} \subset \mathbb{R}^d$, where $d = 2$ for the ALIF and Izhikevich neuron models.

The *eligibility trace* e_{ji}^t is a product of $\frac{\partial z_j^t}{\partial \mathbf{h}_j^t}$ and the eligibility vector, resulting in the gradient that can be immediately applied at every time

step t, or accumulated and integrated locally on a synapse (see Section 2.5.2 for details):

$$\frac{dE}{dW_{ji}} = \sum_t \frac{dE}{dz_j^t} \underbrace{\frac{\partial z_j^t}{\partial \mathbf{h}_j^t} \underbrace{\sum_{t \geq t'} \frac{\partial \mathbf{h}_j^t}{\partial \mathbf{h}_j^{t-1}} \cdots \frac{\partial \mathbf{h}_j^{t+1}}{\partial \mathbf{h}_j^{t'}} \cdot \frac{\partial \mathbf{h}_j^{t'}}{\partial W_{ji}}}_{\boldsymbol{\epsilon}_{ji}^t}}_{e_{ji}^t} . \tag{2.14}$$

This is the main e-prop equation.

2.5 LEARNING PROCEDURE

The e-prop equation (Equation 2.14) can be applied to any neuron type with any number of hidden states. In this section, the derivation for ALIF neurons will be detailed.

2.5.1 *Eligibility trace*

Recall the hidden state update equations from Section 2.2:

$$v_j^{t+1} = \alpha v_j^t + \sum_{i \neq j} W_{ji}^{\text{rec}} z_i^t + \sum_i W_{ji}^{\text{in}} x_i^{t+1} - z_j^t v_{\text{th}} \tag{2.4 revisited}$$

and

$$a_j^{t+1} = \rho a_j^t + z_j^t \tag{2.6 revisited}$$

and the update of the observable state

$$z_j^t = H\left(v_j^t - v_{\text{th}} - \beta a_j^t\right). \tag{2.5 revisited}$$

The hidden state $\mathbf{h}_j^t$ of an ALIF neuron j is therefore a vector containing its activation and threshold adaptation:

$$\mathbf{h}_j^t = \begin{pmatrix} v_j^t \\ a_j^t \end{pmatrix}. \tag{2.15}$$

This hidden state is associated with a two-dimensional eligibility vector

$$\boldsymbol{\epsilon}_{ji}^t \stackrel{\text{def}}{=} \begin{pmatrix} \epsilon_{ji,v}^t \\ \epsilon_{ji,a}^t \end{pmatrix} \tag{2.16}$$

that relates to a synapse from any afferent neuron i to neuron j.

Recall from Chapter 1 that the eligibility trace slowly fades after a spike has occurred on a synapse, such that a delayed learning signal can still modify the synaptic strength accordingly, solving the credit assignment problem. Intuitively, the eligibility vector computes the correct contribution of each of the components of the hidden state. For

a LIF neuron, the only component is the activation value, and so it is simply a low-pass filter of the spikes of the afferent neuron.

For the default ALIF neuron, however, the hidden state derivative $\frac{\partial \mathbf{h}_j^{t+1}}{\partial \mathbf{h}_j^t}$ must be computed to derive the eligibility vector. This hidden state derivative is expressed by a 2×2 matrix of partial hidden state derivatives:

$$\frac{\partial \mathbf{h}_j^{t+1}}{\partial \mathbf{h}_j^t} = \begin{pmatrix} \frac{\partial v_j^{t+1}}{\partial v_j^t} & \frac{\partial v_j^{t+1}}{\partial a_j^t} \\ \frac{\partial a_j^{t+1}}{\partial v_j^t} & \frac{\partial a_j^{t+1}}{\partial a_j^t} \end{pmatrix}. \tag{2.17}$$

The presence of z_j^t, and its relation to the Heaviside step function $H(\cdot)$ in the hidden state updates in Equation 2.4 and Equation 2.6 seems problematic for computing these partial derivatives, because the derivative $\frac{\partial z_j^t}{\partial v_j^t}$ is nonexistent. This is overcome by replacing it with a simple nonlinear function called a pseudo-derivative. Outside of the refractory period of a neuron j, this pseudo-derivative has the form

$$\psi_j^t = \gamma \max\left(0, 1 - \left|\frac{v_j^t - v_{\text{th}} - \beta a_j^t}{v_{\text{th}}}\right|\right), \tag{2.18}$$

where γ is a dampening constant, which is set to 0 during the neuron's refractory period. Like in Esser et al. (2016), this pseudo-derivative is 1 at time steps where the neuron spikes, and linearly decays to zero in the positive and negative direction. The synaptic weight can only change when the pseudo-derivative is nonzero.

Now, the partial derivatives in the hidden state derivative can be computed by replacing the Heaviside function Equation (in 2.5) by the pseudo-derivative ψ_j^t:

$$\frac{\partial v_j^{t+1}}{\partial v_j^t} = \alpha \tag{2.19}$$

$$\frac{\partial v_j^{t+1}}{\partial a_j^t} = 0 \tag{2.20}$$

$$\frac{\partial a_j^{t+1}}{\partial v_j^t} = \psi_j^t \tag{2.21}$$

$$\frac{\partial a_j^{t+1}}{\partial a_j^t} = \rho - \psi_j^t \beta. \tag{2.22}$$

These partial derivatives can be used to compute the eligibility vector:

$$\begin{pmatrix} \epsilon_{ji,v}^{t+1} \\ \epsilon_{ji,a}^{t+1} \end{pmatrix} = \begin{pmatrix} \frac{\partial v_j^{t+1}}{\partial v_j^t} & \frac{\partial v_j^{t+1}}{\partial a_j^t} \\ \frac{\partial a_j^{t+1}}{\partial v_j^t} & \frac{\partial a_j^{t+1}}{\partial a_j^t} \end{pmatrix} \cdot \begin{pmatrix} \epsilon_{ji,v}^{t} \\ \epsilon_{ji,a}^{t} \end{pmatrix} + \begin{pmatrix} \frac{\partial v_j^{t+1}}{\partial W_{ji}} \\ \frac{\partial a_j^{t+1}}{\partial W_{ji}} \end{pmatrix} \tag{2.23}$$

$$= \begin{pmatrix} \alpha & 0 \\ \psi_j^t & \rho - \psi_j^t \beta \end{pmatrix} \cdot \begin{pmatrix} \epsilon_{ji,v}^{t} \\ \epsilon_{ji,a}^{t} \end{pmatrix} + \begin{pmatrix} z_i^{t-1} \\ 0 \end{pmatrix} \tag{2.24}$$

$$= \begin{pmatrix} \alpha \cdot \epsilon_{ji,v}^t + z_i^{t-1} \\ \psi_j^t \epsilon_{ji,v}^t + \left(\rho - \psi_j^t \beta\right) \epsilon_{ji,a}^t \end{pmatrix}. \tag{2.25}$$

Intuitively, these eligibility vector components can be seen as the contribution of the hidden state component to the increase of the eligibility trace. For instance, the activation eligibility component $\epsilon_{ji,v}^t$ of a synapse $i \to j$ at time step t is, like in the LIF neuron, a low-pass filter of the afferent spikes z_i.

The threshold adaptation eligibility component $\epsilon_{ji,a}^t$ is less intuitive, but acts as a correction factor for the more slowly decaying threshold adaptation. Its first term $\psi_j^t \epsilon_{ji,v}^t$ causes it to increase when a neuron has recently spiked and when the activation is already increasing again. Therefore, it is higher for synapses that have a higher spike frequency. The second term threshold adaptation eligibility component is a decay corrected for the adaptation strength β.

This eligibility vector update can be recursively applied. For eligibility vectors of synapses that are efferent to input neurons, the input value x_i^t is used in place of z_i^{t-1} in Equation 2.24. Note that the current time index t is used for input neurons to satisfy the online learning principle defined in the model definition in Equation 2.1; neurons receive input from the input at time t, and from the spikes of other neurons emitted at time $t-1$. Furthermore, the absence of $\epsilon_{ji,a}^t$ in the computation of $\epsilon_{ji,v}^{t+1}$ facilitates online training in emulations in non–von Neumann machines, because $\epsilon_{ji,a}^{t+1}$ can be computed before $\epsilon_{ji,v}^{t+1}$, relieving the need to store a temporary copy of its value. In later sections, it is demonstrated that this does not necessarily hold for other neuron models, such as the Izhikevich neuron.

The eligibility vector needs to be multiplied with the partial derivative of the observable state with respect to the hidden state to obtain the eligibility trace:

$$e_{ji}^t = \boldsymbol{\epsilon}_{ji}^t \cdot \frac{\partial z_j^t}{\partial \mathbf{h}_j^t}. \tag{2.26}$$

Again, the Heaviside function in Equation 2.5 is replaced by ψ_j^t:

$$\frac{\partial z_j^t}{\partial \mathbf{h}_j^t} = \begin{pmatrix} \frac{\partial z_j^t}{\partial v_j^t} \\ \frac{\partial z_j^t}{\partial a_j^t} \end{pmatrix} \tag{2.27}$$

$$= \begin{pmatrix} \psi_j^t \\ -\beta \psi_j^t \end{pmatrix}. \tag{2.28}$$

Therefore, the eligibility trace is computed by

$$e_{ji}^t = \begin{pmatrix} \epsilon_{ji,v}^t \\ \epsilon_{ji,a}^t \end{pmatrix} \cdot \begin{pmatrix} \frac{\partial z_j^t}{\partial v_j^t} \\ \frac{\partial z_j^t}{\partial a_j^t} \end{pmatrix} \tag{2.29}$$

$$= \begin{pmatrix} \epsilon_{ji,v}^t \\ \epsilon_{ji,a}^t \end{pmatrix} \cdot \begin{pmatrix} \psi_j^t \\ -\beta\psi_j^t \end{pmatrix} \tag{2.30}$$

$$= \psi_j^t \left(\epsilon_{ji,v}^t - \beta\epsilon_{ji,a}^t\right). \tag{2.31}$$

This means that the eligibility trace can be understood as a low-pass filter of the afferent spikes, with a correction for the efferent neuron's threshold adaptation: a neuron with a higher threshold builds up an eligibility trace more slowly than its more sensitive counterparts. Figure 2.3 illustrates the behavior of the synaptic variables in an ALIF neuron described above.

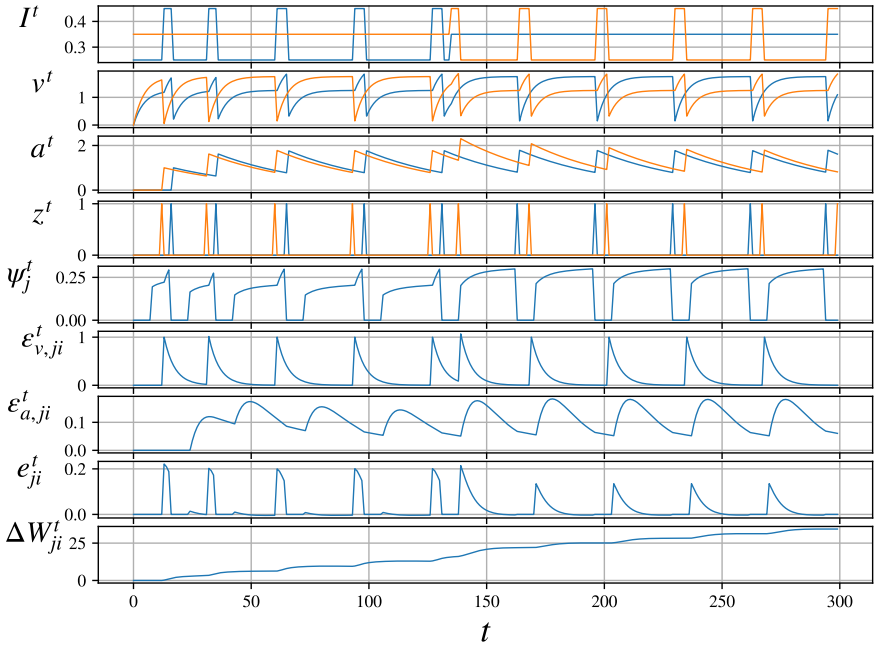


Figure 2.3: A single-synapse simulation of the evolution of the full hidden state of the ALIF neuron. The blue lines indicate the postsynaptic neuron j, and the orange lines indicate the presynaptic neuron i. The injected current I^t increases the voltage v_j^t and is deliberately controlled to produce the spike pattern z_j^t where the postsynaptic neuron spikes after the presynaptic neuron during the first half, and vice versa during the second half of the plot. The learning signal L_j^t is kept at a constant value and is omitted for clarity, such that the relation between the eligibility trace e_{ji}^t and the accumulated weight change ΔW_{ji}^t can be clearly observed. Note that the synapse weight increases regardless of the order of spikes, indicating an absence of STDP in the standard e-prop ALIF neuron.

2.5.2 *Gradients*

Gradient descent is used to apply the weight updates, such that weights are updated by a small fraction η in the negative direction of the estimated gradient of the loss function with respect to the model weights:

$$\Delta W = -\eta \widehat{\frac{dE}{dW_{ji}}} \stackrel{\text{def}}{=} -\eta \sum_t \frac{\partial E}{\partial z_j^t} e_{ji}^t. \tag{2.32}$$

Note that for clarity, this section describes e-prop using stochastic gradient descent. In the actual implementations in Bellec, Scherr, Subramoney, et al. (2020) and this research, the Adam optimization algorithm (Kingma and Ba, 2014) is used (see Section 3.4).

ERROR METRIC In the TIMIT frame-wise phone classification task, there are $K = 61$ output neurons y_k^t where $k \in [1 \,..\, K]$. These are computed according to

$$\hat{y}_k^t = \kappa \hat{y}_k^{t-1} + \sum_j W_{kj}^{\text{out}} z_j^t \tag{2.33}$$

and

$$y_k^t = \hat{y}_k^t + b_k, \tag{2.34}$$

where $\kappa \in [0, 1]$ is the decay factor for the output neurons, W_{kj}^{out} is the weight between neuron j and output neuron k, and b_k is the bias value. The decay factor κ acts as a low-pass filter, smoothing the output values over time and implemented based on the observation that output frame classes typically persist for multiple time steps.

The softmax function $\sigma(\cdot)$ computes the predicted probability π_k^t for class k at time t:

$$\pi_k^t = \sigma_k\left(y_1^t, \ldots, y_K^t\right) = \frac{\exp\left(y_k^t\right)}{\sum_{k'} \exp\left(y_{k'}^t\right)}. \tag{2.35}$$

This predicted probability is compared with the one-hot vector corresponding to the target class label $\pi_k^{*,t}$ at time step t using the cross entropy loss function

$$E = -\sum_{t,k} \pi_k^{*,t} \log \pi_k^t, \tag{2.36}$$

thereby obtaining the accumulated loss E at time step t.

Since the learning signal L_j^t is defined as the partial derivative of the error E with respect to the observable state z_j^t of a neuron j afferent to an output neuron k, we can use

$$L_j^t = \frac{\partial E}{\partial z_j^t} = \sum_k B_{jk} \sum_{t' \geq t} \left(\pi_k^{t'} - \pi_k^{*,t'}\right) \kappa^{t'-t}, \tag{2.37}$$

where B_{jk} is a feedback weight from neuron k back to output neuron j. There are multiple strategies for choosing feedback weights. Bellec, Scherr, Subramoney, et al. (2020) noted that a constantly uniform weight matrix yields poor performance, which has been empirically verified in my project. However, when the feedback weight matrix is initialized from a zero-centered normal distribution, it can remain either constant, mirror $(W^{\text{out}})^\top$, or update according to $(\Delta W^{\text{out}})^\top$. These three variants are referred to in Bellec, Scherr, Subramoney, et al. (2020) as *random*, *symmetric*, and *adaptive* e-prop, respectively. In this paper, symmetric e-prop is used (i.e., $B_{jk} \overset{\text{def}}{=} W^{\text{out}}_{kj}$) unless explicitly stated otherwise.

Note that the term $\kappa^{t'-t}$ in Equation 2.37 is a filter that compensates for the decay factor of output neurons. Note also that this equation does not allow online learning, because future time steps t' are accessed. However, if a low-pass filter with factor κ is applied on the eligibility trace, it will cancel out the effects of the future time steps on the learning signal, and the estimated loss gradient can be approximated. This low-pass filter of the eligibility trace can be implemented in an online fashion by including it as a hidden synaptic variable $\bar{e}^t_{ji}$. Recall that the estimated loss gradient $\widehat{\frac{dE}{dW_{ji}}}$ is approximated by $\sum_t \frac{\partial E}{\partial z^t_j} e^t_{ji}$. Therefore, after inserting Equation 2.37 in Equation 2.32, the weight update is computed by

$$\Delta W_{ji} = -\eta \sum_{t'} \frac{\partial E}{\partial z^{t'}_j} e^{t'}_{ji} \tag{2.38}$$

$$= -\eta \sum_{t'} \sum_k B_{jk} \sum_{t' \geq t} \left(\pi^{t'}_k - \pi^{*,t}_k \right) \kappa^{t'-t} e^{t'}_{ji} \tag{2.39}$$

$$= -\eta \sum_{k,t'} B_{jk} \sum_{t' \geq t} \left(\pi^{t'}_k - \pi^{*,t}_k \right) \kappa^{t'-t} e^{t'}_{ji} \tag{2.40}$$

$$= -\eta \sum_t \underbrace{\sum_k B_{jk} \left(\pi^t_k - \pi^{*,t}_k \right)}_{=L^t_j} \underbrace{\sum_{t' \leq t} \kappa^{t'-t} e^{t'}_{ji}}_{\overset{\text{def}}{=} \bar{e}^t_{ji}}, \tag{2.41}$$

where W_{ji} is an input or recurrent weight. By implementing $\bar{e}_{ji}$ as a low-pass filter (with factor κ) of the eligibility trace, the weight update in Equation 2.41 is implemented as a local and online learning algorithm.

The training algorithm for the output weights W^{out} and bias b can be directly derived from gradient descent:

$$\Delta W^{\text{out}}_{kj} = -\eta \sum_t \left(\pi^t_k - \pi^{*,t}_k \right) \sum_{t' \leq t} \kappa^{t'-t} z^t_j \tag{2.42}$$

and

$$\Delta b_k = -\eta \sum_t \left(\pi^t_k - \pi^{*,t}_k \right). \tag{2.43}$$

3 METHOD

3.1 DATA PREPROCESSING

In this section, the content of the TIMIT speech corpus is described, as well as the preprocessing method that transforms the speech signals into usable features in the e-prop framework.

3.1.1 *The TIMIT speech corpus*

TIMIT is a speech corpus that contains phonemically transcribed speech (Garofolo et al., 1993), comprising 6300 sentences, 10 spoken by each of the 630 speakers. To include a broad range of dialects these speakers are sampled from 8 different geographical regions in the United States (as categorized in Labov, Ash, and Boberg (2008)) in which they lived during their childhood years. Table 3.1 breaks down the precise composition of the dialect distribution.

DIALECT REGION	#MALE	#FEMALE	TOTAL
1 (New England)	31 (63%)	18 (27%)	49 (8%)
2 (Northern)	71 (70%)	31 (30%)	102 (16%)
3 (North Midland)	79 (67%)	23 (23%)	102 (16%)
4 (South Midland)	69 (69%)	31 (31%)	100 (16%)
5 (Southern)	62 (63%)	36 (37%)	98 (16%)
6 (New York City)	30 (65%)	16 (35%)	46 (7%)
7 (Western)	74 (74%)	26 (26%)	100 (16%)
8	22 (67%)	11 (33%)	33 (5%)
All	438 (70%)	192 (30%)	630 (100%)

Table 3.1: Distribution of speakers' dialect regions and sexes. Speakers of the innominate dialect region 8 relocated often during their childhood.

The sentence text can be categorized into 2 *dialect* sentences, 450 *phonetically compact* sentences, and 1890 *phonetically diverse* sentences.

The dialect sentences, which are spoken by all speakers, are designed to expose the dialectical variants of the speakers. The phonetically compact sentences are designed to include many pairs of phones. The phonetically diverse sentences are taken from the Brown Corpus (Kucera, Kučera, and Francis, 1967) and the Playwrights Dialog (Hultzsch et al., 1964) in order to maximize the number of allophones (i. e., different phones used to pronounce the same phoneme). Table 3.2 lists an overview of the distribution of the number of speakers per sentence type.

SENTENCE TYPE	#SENTENCES	#SPEAKERS	TOTAL
Dialect	2	630	1260
Compact	450	7	3150
Diverse	1890	1	1890
Total	2342		6300

Table 3.2: Distribution of sentence types.

Each of the sentences is encoded in as a waveform signal in `.wav` format, and is accompanied by a corresponding text file indicating which phones are pronounced in the waveform, and between which pairs of sample points.

3.1.2 *Data splitting*

The TIMIT dataset is split into a training, validation and testing set as in Graves and Schmidhuber (2005) and Bellec, Scherr, Subramoney, et al. (2020). The training set is used to train the network synaptic weights according to the e-prop algorithm. The validation set is used to obtain a well-performing set of hyperparameters. The testing set is used to evaluate the performance of the network after the hyperparameters are obtained.

The TIMIT corpus documentation offers a suggested partitioning of the training and testing data, which is based on the following criteria:

1. 70%–80% of the data is used for training, and the remaining 20%–30% for testing.
2. No speaker appears in both the training and testing partitions.
3. Both subsets include at least 1 male and 1 female speaker from every dialect region.
4. There is a minimal overlap of text material in the two subsets.
5. The test set should contain all phones in as many allophonic contexts as possible.

In accordance with these criteria, the TIMIT corpus includes a "core" test set that contains 2 male speakers and 1 female speaker from each dialect, summing up to 24 speakers. Each of these speakers read a different set of 5 phonetically compact sentences, and 3 phonetically diverse sentences that were unique for each speaker. Consequently, the test set comprises 192 sentences ($24 \times (5 + 3)$) and was selected such that it contains at least one occurrence of each phone. In this report, the TIMIT core test set is used, thereby meeting the criteria listed above.

The remaining 4096 sentences are randomly partitioned into 3696 training sentences and 400 validation sentences in this research (the TIMIT corpus contains no fixed training/validation set partition).

3.1.3 *Engineering features*

The data preprocessing pipeline is similar to that used in Fayek (2016), which can be summarized by applying a pre-emphasis filter on the waveforms, then slicing the waveform in short frames, taking their short-term power spectra, computing 26 filterbanks, and finally obtain 12 Mel-Frequency Cepstrum Coefficients (MFCCs). Then, these MFCCs are aligned with the phones found in the TIMIT dataset. An example of a waveform signal is given in Figure 3.1.

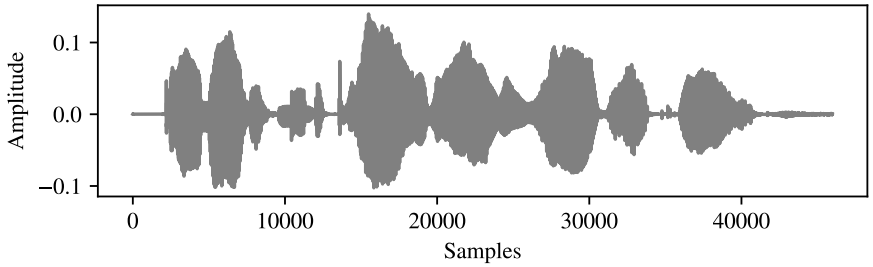


Figure 3.1: Raw TIMIT waveform signal

PRE-EMPHASIS In speech signals, high frequencies generally have smaller magnitudes than lower frequencies. To balance the magnitudes over the range of frequencies in the signal, a pre-emphasis filter $y(t)$ is applied on the waveform signal $x(t)$:

$$y(t) = x(t) - 0.97x(t-1). \tag{3.1}$$

This procedure yields the additional benefit of improving the signal-to-noise ratio. An example of a pre-emphasized signal is given in Figure 3.2.

FRAMING The waveforms, which are sampled at a rate f_s of 16 kHz, cannot be directly used as input to the model, because they are too long—a typical sentence waveform contains in the order of tens of thousands of data points. Furthermore, the individual data points are not very informative, because they reflect the sound wave of the uttered sound, not the characteristics of the source of this sound. These sounds are filtered by the shape of the vocal tract, which manifests itself in the envelope of the short time power spectrum of the sound. This power spectrum representation describes the power of the frequency components of the signal over a brief interval. The frequency components are assumed to be stationary over short intervals, in contrast to the full sentence, which carries its meaning because it is non-stationary. Therefore, the waveform signals are transformed into series of frequency coefficients of short-term power spectra. To obtain these multiple short-term power spectra over the duration of the waveform, it is sliced into brief overlapping frames.

Every 160 samples (equivalent to 10 ms) of a pre-emphasized signal, an interval frame of 400 samples (equivalent to 25 ms) is extracted. This

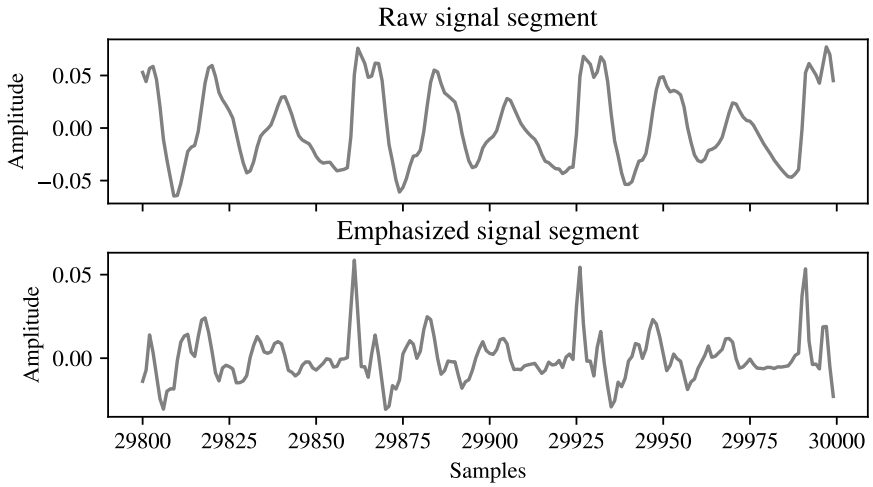


Figure 3.2: A segment of a signal after the pre-emphasis filter of Equation 3.1 was applied to it. The upper panel contains samples 28000–30000 of the signal in Figure 3.1, while the lower panel contains its pre-emphasis filtered counterpart. Note that the filtered signal is less symmetric around the horizontal axis, because the filtered signal is similar to a first derivative—the original signal has sharper increases than decreases, so the filtered signal has stronger extrema towards the positive direction.

means that the frames overlap by 25 ms. The waveform is zero-padded such that the last frame also has 400 samples. By this process, signal frames $x_i(n)$ are obtained, where n ranges over 1–400, and i ranges over the number of frames in the waveform.

Then, a Hamming window with the form

$$w\,[n] = a_0 - a_1 \cos\left(\frac{2\pi n}{N-1}\right), \tag{3.2}$$

is applied where N is the window length of 400 samples, $0 \leq n < N$, $a_0 = 0.53836$, and $a_1 = 0.46164$. A plot of this window is given in Figure 3.3. This window is applied to reduce the spectral leakage, which manifests itself though sidelobes in the power spectra. Applying the Hamming window reduces the sidelobes to near-equiripple conditions, minimizing the leakage (Smith, accessed December 2020).

SHORT-TERM POWER SPECTRA The power spectra P_i are obtained for each frame by first taking the absolute K-point discrete Fourier transform (DFT) of the frame samples $x_i(n)$:

$$X_k = \left| \sum_{n=0}^{N-1} x_i(n) \cdot e^{-\frac{i2\pi}{N} kn} \right|, \tag{3.3}$$

where $K = 512$. This yields the magnitudes of the discrete cosine transform (DCT) of the frames.

The power spectra are obtained using the equation

$$P = \frac{{X_k}^2}{K}, \tag{3.4}$$

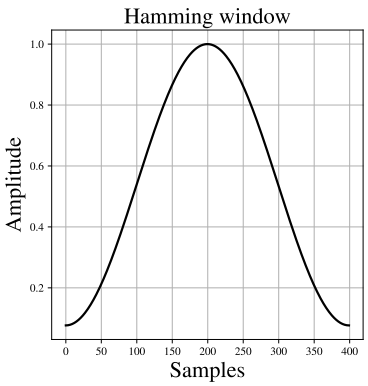


Figure 3.3: The form of the Hamming window applied on signal frames to reduce spectral leakage.

an example of which is shown in Figure 3.4.

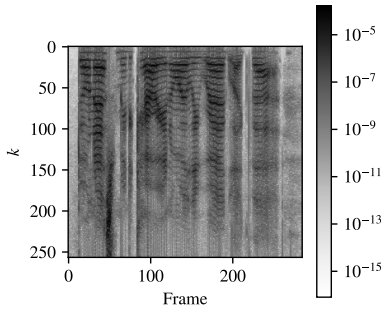


Figure 3.4: The power spectra of a sentence.

MEL FILTERBANK The short-term power spectra are then transformed to Mel-spaced filterbanks. The Mel scale is a scale of pitches that are perceptually equal in distance (Stevens, Volkmann, and Newman, 1937). This is in contrast to the frequency measurement, in which the human cochlea can distinguish lower frequencies more accurately than higher ones. The aim of converting to the Mel scale is to make every filterbank coefficient feature equally informative, thereby improving the learning performance of the model.

The Mel-spaced filterbank is a set of 40 triangular filters that we apply to each frame in P.

To compute the Mel-spaced filterbank, lower and upper band edges of respectively 0 Hz and $f_s/2 = 8$ kHz are selected, and convert these to Mels using

$$m(f) = 2595 \log_{10}\left(1 + \frac{f}{700}\right), \tag{3.5}$$

where f is the frequency in Hz. This yields a lower band edge of 0 Mels and an upper band edge of approximately 2835 Mels.

The 40 filterbanks are obtained by first spacing 42 points $\mathbf{m}$ linearly between these bounds (inclusive), thereby obtaining 40 points spaced exclusively between the bounds.

Then, the vector of Mel frequencies $\mathbf{m}$ is converted back to Hz using

$$\mathbf{f} = 700\left(10^{\mathbf{m}/2595} - 1\right). \tag{3.6}$$

The resulting frequencies $\mathbf{f}$ are rounded to their nearest Fourier transform bins $\mathbf{b}$ using

$$\mathbf{b} = \lfloor (K+1)\mathbf{f}/fs \rfloor. \tag{3.7}$$

The resulting 40 filterbanks with their corresponding Mels and frequencies are listed in Table A.1.

The i^{th} filter in filterbank H_i is a triangular filter that has its lower boundary at b_i Hz, its peak at b_{i+1} Hz, and its upper boundary at b_{i+2} Hz. For other frequencies, they are 0. Therefore, the filterbank can be described by

$$H_i(k) = \begin{cases} 0 & \text{if } k < b_i \\ \frac{k-b_i}{b_{i+1}-b_i} & \text{if } b_i \leq k < b_{i+1} \\ 1 & \text{if } k = b_{i+1} \\ \frac{b_{i+2}-k}{b_{i+2}-b_{i+1}} & \text{if } b_{i+1} < k \leq b_{i+2} \\ 0 & \text{if } b_{i+2} < k \end{cases}, \tag{3.8}$$

where $0 \leq k \leq \frac{K}{2}$. These Mel-spaced filters are shown in Figure 3.5.

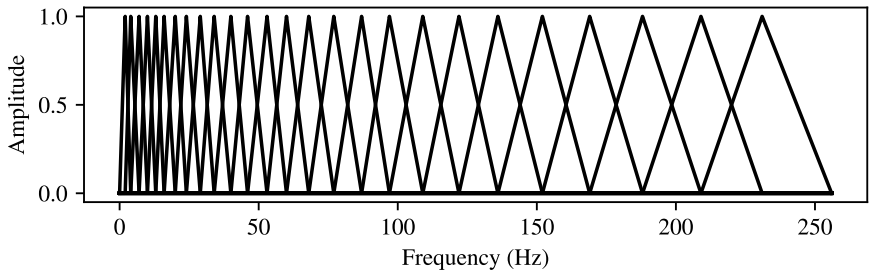


Figure 3.5: The Mel-spaced filterbanks.

After applying the filterbank to the short-term power spectrum, a spectrogram S of the frame sequence (see e. g. Figure 3.6) is obtained.

MEL-FREQUENCY CEPSTRAL COEFFICIENTS Coefficients in the spectrograms are strongly correlated, which would negatively impact the learning performance of the model. Therefore, the DCT is applied again to decorrelate the coefficients and obtain the power cepstrum C of the speech frame:

$$C_k = 2c \sum_{n=0}^{N-1} S(n) \cos\left(\frac{\pi k (2n+1)}{2N}\right), \tag{3.9}$$

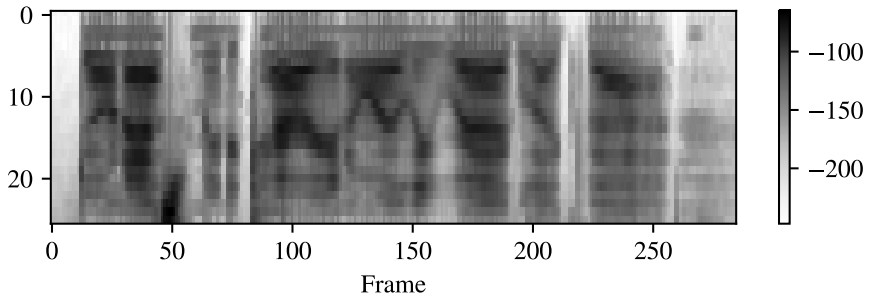


Figure 3.6: An example of the spectrogram of a sentence.

where c is a scaling factor that makes the matrix of coefficients orthonormal:

$$c = \begin{cases} \sqrt{\frac{1}{4N}} & \text{if } k = 0, \\ \sqrt{\frac{1}{2N}} & \text{otherwise.} \end{cases} \tag{3.10}$$

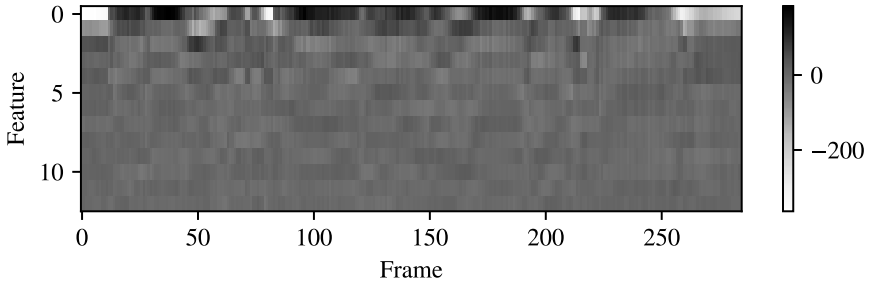
The first coefficient in C is discarded, because it indicates the average power of the input signal and therefore carries little meaning. Coefficients higher than 13 are also discarded, because they represent only fast changes in the spectrogram and increase the complexity of the input signal while adding increasingly less meaning to it. Next, the first and second derivatives of the MFCCs over time are concatenated to the 13 MFCC features, obtaining an input vector of size 39.

Then, these input vectors are balanced by centering each frame around the value 0. An example of these input vectors is given in Figure 3.7. Note that before training a model, all input vectors (including validation and testing data) are standardized channel-wise according to the full training set (see 4.1 for an example input as used by the model).

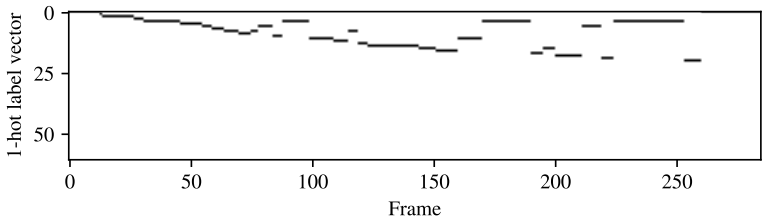
TARGET OUTPUT The target output of the model is a frame-wise representation of the phones that are uttered in a sentence. The TIMIT corpus contains text files indicating in what order phones occur in a sentence, and their starting and ending sample points.

These phones are discretized into frames such that they align correctly with the MFCCs. They are represented in one-hot vector encoding. Since the dataset contains 61 different phones, this is also the length of these vectors.

Figure 3.7 illustrates the waveform data and its frame-wise aligned MFCCs and target output. Note that the full dataset of features is standardized per training data channel; feature channels are first centered around 0, and then divided by their standard deviations. To prevent data leakage, validation and testing data are standardized according to the means and standard deviations in the training data. An example of a standardized input is shown in Figure 4.1.



(a) MFCCs centered around 0 per channel. Note that these MFCCs are standardized channel-wise over the full training dataset, and that the first and second derivations of the MFCC features are omitted.



(b) Target signal encoded as a one-hot vector changing over time. The order of phones along the one-hot vector corresponds to the order in which they are encountered in processing the dataset. This particular example shows a pattern because it is the first processed sentence in the training set.

Figure 3.7: MFCCs features and corresponding target phones.

3.2 ENHANCING E-PROP

In my project I examined two types of enhancements to apply the e-prop learning algorithm on the TIMIT dataset.

The first type is the effect of the neuron model; particularly, the effect of including STDP behavior is analyzed. The second type is the effect of a multi-layered architecture.

The results of these enhancements will answer the research questions posed in Chapter 1, i.e., whether multi-layered architectures or inclusion of STDP in neuron models improves the performance of e-prop.

3.2.1 *Multi-layer architecture*

The multi-layer e-prop architecture can be described in the same formal model as its single-layer counterpart, in which the hidden state is based

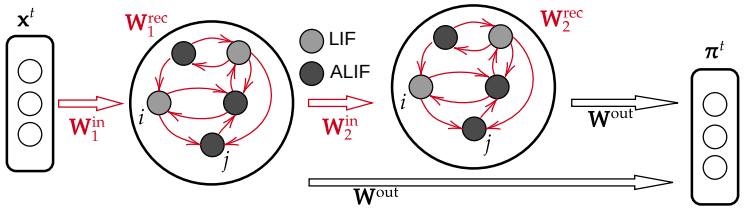


Figure 3.8: An illustration of a multi-layer network architecture. Some details shown in Figure 2.2 are omitted for clarity.

on temporally (i.e., online) and spatially locally available information at a neuron j:

$$\mathbf{h}_j^t = M\left(\mathbf{h}_j^{t-1}, \mathbf{z}^{t-1}, \mathbf{x}^t, \mathbf{W}_j\right). \tag{2.1 revisited}$$

For the multi-layer architecture, however, neurons in deeper layers no longer depend on the input, but on the observable states of the previous layer at the same time step, such that at every time step, a full pass through the network is made. We modify the indexing notation accordingly, in order to directly refer to neurons and weights in a particular layer $r \in [1 \,..\, R]$:

$$\mathbf{h}_{rj}^t = \begin{cases} M\left(\mathbf{h}_{rj}^{t-1}, \mathbf{z}_r^{t-1}, \mathbf{x}^t, \mathbf{W}_{rj}\right) & \text{if } r = 1 \\ M\left(\mathbf{h}_{rj}^{t-1}, \mathbf{z}_r^{t-1}, \mathbf{z}_{r-1}^t, \mathbf{W}_{rj}\right) & \text{otherwise,} \end{cases} \tag{3.11}$$

where $\mathbf{h}_{rj}^t$ (resp. z_{rj}^t) is the hidden state (resp. observable state) of a neuron j in layer r, and $\mathbf{W}_{rj} = \mathbf{W}_{rj}^{\text{in}} \cup \mathbf{W}_{rj}^{\text{rec}}$ is the set of afferent weights to neuron j in layer r.

Similarly, the observable state can be modeled by

$$z_{rj}^t = f\left(\mathbf{h}_{rj}^t\right) \tag{3.12}$$

and the network output by

$$y_k^t = \kappa y_k^{t-1} + \sum_{j,r} W_{rkj}^{\text{out}} z_{rj}^t + b_k, \tag{3.13}$$

where W_{rkj}^{out} is a weight between neuron j in layer r and output neuron k. Note that the summation over r entails that the output layer is connected to all neurons in all layers in the network. This allows trainable broadcast weights in earlier layers, such as those found in symmetric and adaptive e-prop.

MULTI-LAYER ALIF NEURONS An ALIF neuron in a multi-layer architecture is similar to one in a single-layer architecture (see Section 2.2). The only difference, apart from the layer indexing, is its activity update. For a multi-layer ALIF neuron, the activity value is given by

$$v_{rj}^{t+1} = \alpha v_{rj}^t + \sum_{i \neq j} W_{rji}^{\text{rec}} z_i^t + \sum_i W_{rji}^{\text{in}} I - z_{rj}^t v_{\text{th}}, \tag{3.14}$$

where

$$I = \begin{cases} x_i^{t+1} & \text{if } r = 1 \\ z_{r-1,i}^{t+1} & \text{otherwise.} \end{cases} \tag{3.15}$$

3.2.2 *Other neuron types*

In this section, the STDP-ALIF and Izhikevich neuron models are presented. The advantage of these models over the standard ALIF model is that they naturally elicit STDP. Additionally, the Izhikevich model has an implicit refractory mechanism that is built into its system of equations, making it a more biologically plausible model, as opposed to the (STDP-)ALIF model that requires an explicit timer variable in a neuron's hidden state (but it is still local and online).

The Izhikevich e-prop neuron model was first presented by Traub et al. (2020) only in a single-synapse demonstration of its STDP properties. In this report, the performance of the e-prop Izhikevich model is experimentally validated in a learning task for the first time. Traub et al. (2020) also described the STDP-LIF, which is a non-adaptive modification of the standard LIF neuron. Here, its adaptive counterpart, the STDP-ALIF model, is derived and validated as well. This allows for a direct comparison between the ALIF and STDP-ALIF models, such that the effects of STDP can be more precisely analyzed.

3.2.2.1 *STDP-ALIF*

The key change between the ALIF and STDP-ALIF neuron is that the latter is reset to zero at a spike event, and when its refractory period of δt_{ref} time steps ends. Recall that in contrast, the standard ALIF neuron only uses a soft reset by including a term $-z_{rj}^t v_{\text{th}}$ in its activation update equation (Equation 3.14).

The activation update of the STDP-ALIF neuron is therefore:

$$v_{rj}^{t+1} = \alpha v_{rj}^t + \sum_{i \neq j} W_{rji}^{\text{rec}} z_i^t + \sum_i W_{rji}^{\text{in}} I - z_{rj}^t \alpha v_{rj}^t - z_{rj}^{t-\delta t_{\text{ref}}} \alpha v_{rj}^t, \tag{3.16}$$

where, again, I is the network input if $r = 1$, otherwise it is the observable state of neuron i in the preceding layer (see Equation 3.15). Recall that a neuron cannot spike for T^{refr} time steps after its last spike—therefore, the neuron is still suppressed at time step $t - \delta t_{\text{ref}}$ and hence the fourth and fifth terms $-z_{rj}^t \alpha v_{rj}^t$ and $-z_{rj}^{t-\delta t_{\text{ref}}} \alpha v_{rj}^t$ cannot simultaneously be nonzero. Note also that Equation 3.16 will not necessarily set the activation value precisely to 0 at spike time and after the refractory time, as only the activation that caused the neuron to spike will be subtracted, and the new input values described in its second and third term will be added to the new activation value.

The hidden state derivative changes accordingly:

$$\frac{\partial v_{rj}^{t+1}}{\partial v_{rj}^{t}} = \alpha - z_{rj}^{t}\alpha - \alpha z_{rj}^{t-\delta t_{\text{ref}}} \tag{3.17}$$

$$= \alpha\left(1 - z_{rj}^{t} - z_{rj}^{t-\delta t_{\text{ref}}}\right). \tag{3.18}$$

Note that the absence of a_{rj}^{t} in the new activation update entails that the other entries of the hidden state Jacobian are equal to those of the ALIF model, i. e., $\frac{\partial v_{rj}^{t+1}}{\partial a_{rj}^{t}} = 0$, $\frac{\partial a_{rj}^{t+1}}{\partial v_{rj}^{t}} = \psi_{rj}^{t}$, and $\frac{\partial a_{rj}^{t+1}}{\partial a_{rj}^{t}} = \rho - \psi_{rj}^{t}\beta$.

Using these values, we compute the new eligibility trace:

$$\boldsymbol{\epsilon}_{rji}^{t+1} = \frac{\partial \mathbf{h}_{rj}^{t+1}}{\partial \mathbf{h}_{rj}^{t}} \cdot \boldsymbol{\epsilon}_{rji}^{t} + \frac{\partial \mathbf{h}_{rj}^{t+1}}{\partial W_{rji}} \tag{3.19}$$

$$= \begin{pmatrix} \alpha\left(1 - z_{rj}^{t} - z_{rj}^{t-\delta t_{\text{ref}}}\right)\epsilon_{rji,v}^{t} + z_{ri}^{t-1} \\ \psi_{rj}^{t}\epsilon_{rji,v}^{t} + \left(\rho - \psi_{rj}^{t}\beta\right)\epsilon_{rji,a}^{t} \end{pmatrix} \tag{3.20}$$

Note that the observable state of the afferent neuron z_{ri}^{t-1} in Equation 3.20 changes to $z_{r-1,i}^{t}$ if $r > 1$ and $\boldsymbol{\epsilon}_{rji}^{t+1}$ corresponds to a weight between layer $r-1$ and r. If the weight is instead between the network input and the first layer, then this z_{ri}^{t-1} changes to x_{i}^{t}. Note also that this corrects an inconsistency in the STDP-LIF model described in Traub et al. (2020), where the observable state at the current time step t is accessed instead of $t-1$, thereby diverging from the e-prop model in Equation 3.11.

The eligibility trace remains

$$e_{rji}^{t} = \frac{\partial z_{rj}^{t}}{\partial \mathbf{h}_{rj}^{t}} \cdot \boldsymbol{\epsilon}_{rji}^{t} \tag{3.21}$$

$$= \psi_{rj}^{t}\left(\epsilon_{rji,v}^{t} - \beta\epsilon_{rji,a}^{t}\right). \tag{3.22}$$

By resetting the neuron activation at spike time and after the refractory time, STDP is introduced in the system by clamping the pseudo-derivative to a negative value, instead of 0, during the refractory time:

$$\psi_{rj}^{t} = \begin{cases} -\gamma & \text{if } t - t_{z_{rj}} < \delta t_{\text{ref}} \\ \gamma \max\left(0, 1 - \left|\frac{v_{rj}^{t} - v_{\text{th}}}{v_{\text{th}}}\right|\right) & \text{otherwise} \end{cases} \tag{3.23}$$

The factor of the pseudo-derivative and the eligibility vector can therefore produce both positive or negative eligibility traces and gradients (see Figure 3.9).

3.2.2.2 *Izhikevich neuron*

The standard system of equations of the Izhikevich neuron is described by

$$v' = 0.04v^2 + 5v + 140 - a + I \tag{3.24}$$

$$a' = 0.004v - 0.02a, \tag{3.25}$$

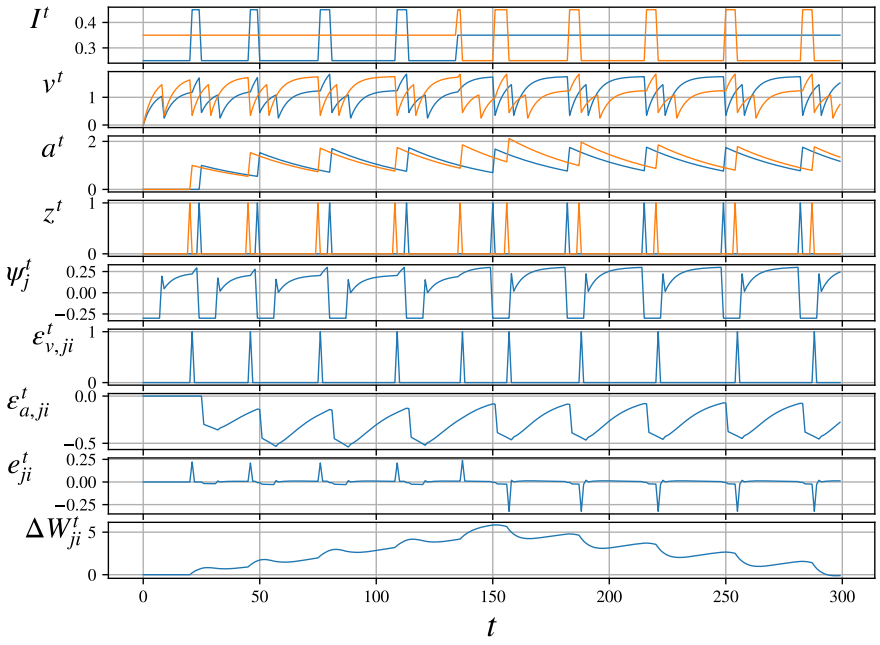


Figure 3.9: A single-synapse simulation of the STDP-ALIF neuron model. Orange and blue lines respectively describe properties of the afferent and efferent neuron.

where v' and a' are the values of the activation value v and recovery variable a at the next time point, and I is the current input to the neuron. Following Traub et al. (2020), the activation reset and refractory period are built into this system of equations by replacing v and a by respectively:

$$\tilde{v}_{rj}^t = v_{rj}^t - \left(v_{rj}^t + 65\right) z_{rj}^t \tag{3.26}$$

$$\tilde{a}_{rj}^t = a_{rj}^t + 2z_{rj}^t, \tag{3.27}$$

such that when a spike event occurs (i. e., $z_{rj}^t = 1$), the activation value is reset to its baseline value of -65, and the recovery variable increases by 2. We describe this "self-resetting" Izhikevich neuron in the context of multi-layer e-prop as follows:

$$v_{rj}^{t+1} = \tilde{v}_{rj}^t + 0.04\left(\tilde{v}_{rj}^t\right)^2 + 5\tilde{v}_{rj}^t + 140 - \tilde{a}_{rj}^t + I_{rj}^t \tag{3.28}$$

$$a_{rj}^{t+1} = \tilde{a}_{rj}^t + 0.004\tilde{v}_{rj}^t - 0.02\tilde{a}_{rj}^t. \tag{3.29}$$

The partial derivatives of the hidden state $\mathbf{h}_{rj}^t$ can then be computed:

$$\frac{\partial v_{rj}^{t+1}}{\partial v_{rj}^t} = \left(1 - z_{rj}^t\right)\left(6 + 0.08 v_{rj}^t\right) \tag{3.30}$$

$$\frac{\partial a_{rj}^{t+1}}{\partial v_{rj}^t} = -1 \tag{3.31}$$

$$\frac{\partial v_{rj}^{t+1}}{\partial a_{rj}^t} = 0.004\left(1 - z_{rj}^t\right) \tag{3.32}$$

$$\frac{\partial a_{rj}^{t+1}}{\partial a_{rj}^t} = 0.98. \tag{3.33}$$

Using these values, we compute the new eligibility trace:

$$\boldsymbol{\epsilon}_{rji}^{t+1} = \frac{\partial \mathbf{h}_{rj}^{t+1}}{\partial \mathbf{h}_{rj}^t} \cdot \boldsymbol{\epsilon}_{rji}^t + \frac{\partial \mathbf{h}_{rj}^{t+1}}{\partial W_{rji}} \tag{3.34}$$

$$= \begin{pmatrix} \left(1 - z_{rj}^t\right)\left(6 + 0.08 v_{rj}^t\right)\epsilon_{rji,v}^t - \epsilon_{rji,a}^t + z_{ri}^{t-1} \\ 0.004\left(1 - z_{rj}^t\right)\epsilon_{rji,v}^t + 0.98\epsilon_{rji,a}^t \end{pmatrix} \tag{3.35}$$

As in Traub et al. (2020), the pseudo-derivative is defined as

$$\psi_{rj}^t = \gamma \ \exp\left(\frac{\min\left(v_{rj}^t, 30\right) - 30}{30}\right), \tag{3.36}$$

such that

$$\begin{pmatrix} \frac{\partial z_{rj}^t}{\partial v_{rj}^t} \\ \frac{\partial z_{rj}^t}{\partial u_{rj}^t} \end{pmatrix} = \begin{pmatrix} \psi_{rj}^t \\ 0 \end{pmatrix}, \tag{3.37}$$

and therefore only $\epsilon_{rji,v}^t$ is used in computing the eligibility trace:

$$e_{rji}^t = \begin{pmatrix} \psi_{rj}^t & 0 \end{pmatrix} \begin{pmatrix} \epsilon_{rji,v}^t \\ \epsilon_{rji,a}^t \end{pmatrix} \tag{3.38}$$

$$= \psi_{rj}^t \epsilon_{rji,v}^t. \tag{3.39}$$

However, when inserting these equations in a single-synapse demo, the eligibility vector assumes extremely high or low values (see Figure 3.10).

This suggests that the Izhikevich neuron does not fit the e-prop framework well. In this report, this is corrected by clipping the value of $\epsilon_{ji,v}^t$ to $[-3, 3]$ and $\epsilon_{ji,a}^t$ to $[-0.005, 0.005]$. This correction yields the desired STDP behavior (see Figure 3.11).

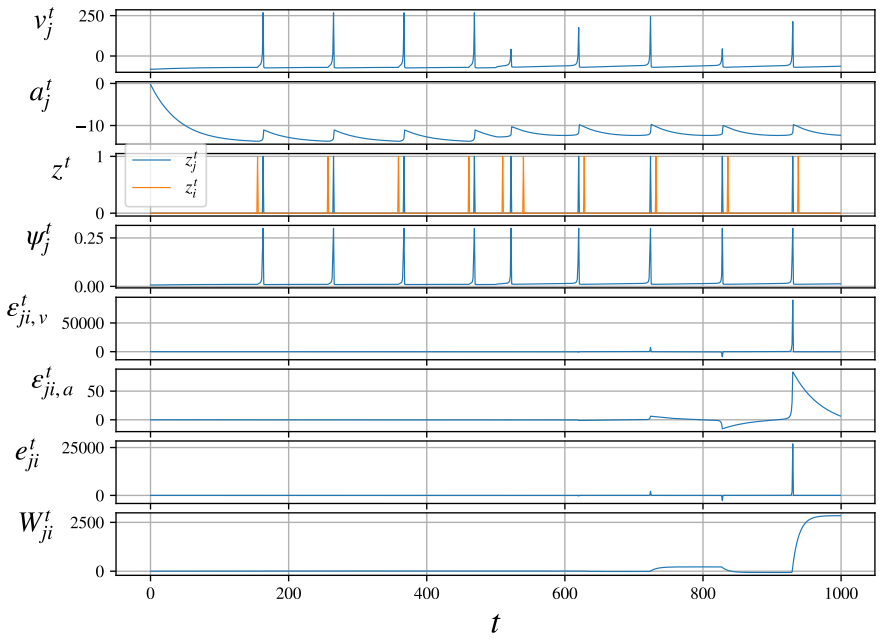


Figure 3.10: A single-synapse simulation of the uncorrected Izhikevich neuron. Note that $\epsilon_{ji,v}^t$ takes on extreme values and that the eligibility vector flips sign at any pair of spikes. Orange and blue lines respectively describe properties of the afferent and efferent neuron.

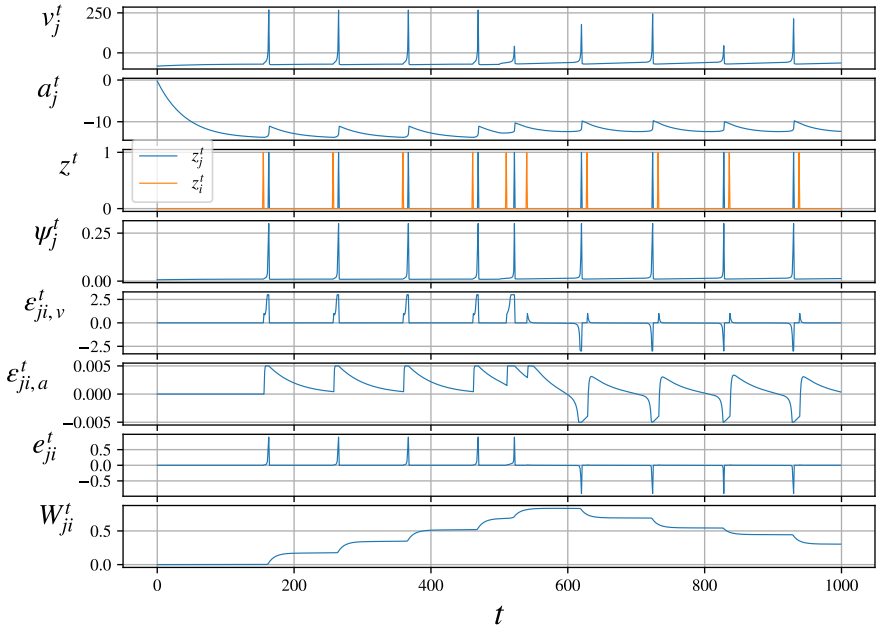


Figure 3.11: A single-synapse simulation of the corrected Izhikevich neuron. Orange and blue lines respectively describe properties of the afferent and efferent neuron.

3.3 REGULARIZATION

Firing rate regularization and L2 regularization are applied to improve the stability of the learning process and the generalizability of the resulting model. These two regularization methods are motivated by biological plausibility, ease of implementation in the e-prop framework, and improved empirical performance.

3.3.1 *Firing rate regularization*

A firing rate regularization term is added to individually modulate the spike frequencies of the neurons. Since spikes in a neuromorphic architecture cost energy, the practical motivation for this regularization term is that it increases the energy efficiency of the model. The biological motivation is that the firing rate of biological neurons is also under homeostatic control (Erickson et al., 2006).

Firing rate is implemented by adding a regularization term E_{reg} to the loss function that penalizes neurons that have a too low or too high firing rate:

$$E_{\text{reg}} = \frac{1}{2} \sum_j \left(f^{\text{target}} - f_{rj}^{\text{av},t} \right)^2, \tag{3.40}$$

where f^{target} is a target firing rate of 10 Hz, and

$$f_{rj}^{\text{av},t} = \frac{1}{t} z_{rj}^{\text{total},t} \tag{3.41}$$

is the running average of the spike frequency, where $z^{\text{total},t}$ accumulates spikes emitted by neuron j in layer r up to (and including) time step t. Note that $z^{\text{total},0} = 0$, i. e., the accumulation resets at each new training sample. By implementing this sum as a hidden variable, e-prop remains an online and local training algorithm when firing rate regularization is implemented. Adhering to these two constraints supports the biological plausibility of the firing rate regularization term. After the training sample, the effect of the firing rate regularization on the weight update is integrated.

Another possibility to compute the average firing rate would be to track an exponentially decaying firing rate. This was not implemented in this research for the following two reasons. First, the effects of the firing rate regularization are integrated only at the end of a training sample (see Equation 3.43), likely causing any fluctuations of the average frequency over time to cancel each other out and result in an effectively similar regularization term as the current, non-decaying frequency calculation. Second, since one of the objectives of this research is to reproduce results obtained in Bellec, Scherr, Subramoney, et al. (2020) and expand on that paper, the regularization term is kept as faithful to theirs as possible. However, analyzing the effects of different firing rate regularization terms might be an interesting direction for future research.

To insert the regularization term into the e-prop framework, we compute the weight update that regularizes the firing rate toward f^{target}

through gradient descent, similarly to the main e-prop weight update in Equation 2.32:

$$\frac{\partial E_{\text{reg}}}{\partial z_{rj}^t} = \left(f^{\text{target}} - f_{rj}^{\text{av},t} \right). \tag{3.42}$$

Note that this regularization loss differs from the firing rate regularization described in Bellec, Scherr, Subramoney, et al. (2020), in which the firing rate is calculated in an offline fashion, by retroactively computing the average firing rate based on all spikes instead of only accumulated spikes. Note also that in Bellec, Scherr, Subramoney, et al. (2020), $\frac{\partial E_{\text{reg}}}{\partial z_{rj}^t}$ is multiplied with the eligibility trace e_{rji}^t, as in Equation 2.32 to obtain the weight update, whereas in this report, the eligibility trace is omitted, resulting in a number of benefits:

1. It allows silent neurons that have infrequently spiking afferent neurons to more easily increase their firing rate, because their low afferent eligibility traces no longer nullify the regularization gradients, and thereby result in a better empirical learning performance.

2. It is more efficient in emulations on von Neumann machines, because the element-wise multiplication of $\frac{\partial E_{\text{reg}}}{\partial z_{rj}^t}$ and the eligibility trace is a relatively large computation on the order $\Theta\left(n^2\right)$ that no longer needs to be computed.

3. It is more intuitive, as only the gradient of the firing rate is used to compute the weight update.

We apply the weight update ΔW_{rji} of the regularization gradient using

$$\Delta_{\text{reg}} W_{rji} = -\eta \; c_{\text{reg}} \sum_t \left(f^{\text{target}} - f_{rj}^{\text{av},t} \right). \tag{3.43}$$

Note that the regularization gradients can be combined and accumulated over time on the same synaptic variable as the normal gradients, facilitating practical implementation of the learning procedure in both software emulations and neuromorphic embeddings:

$$\Delta W_{rji} = -\eta \; \sum_t \left(c_{\text{reg}} \left(f^{\text{target}} - f_{rj}^{\text{av},t} \right) + L_{rj}^t \cdot \bar{e}_{rji}^t \right). \tag{3.44}$$

3.3.2 *L2 regularization*

To regularize weights around 0, a small fraction (parametrized by c_{L2}) of the value of the weight is added to its gradient value at every weight update:

$$\Delta_{\text{L2}} W_{rji} = -\eta \; c_{\text{L2}} \cdot W_{rji}, \tag{3.45}$$

which can be added to the full weight update as an extra term:

$$\Delta W_{rji} = -\eta \left(c_{\text{L2}} \cdot W_{rji} + \sum_t \left(c_{\text{reg}} \left(f^{\text{target}} - f_{rj}^{\text{av},t} \right) + L_{rj}^t \cdot \bar{e}_{rji}^t \right) \right).$$

(3.46)

The statistical motivation for this extra regularization term is that by softly restricting the weights, the network is less likely to overfit on the training data.

The biological motivation is that biological synapses are regularized by a multiplicative factor to decrease the strength of individual synapses to the same proportion (Turrigiano and Nelson, 2000), likely counteracting the run-away effects that the positive feedback of STDP naturally induces (Siddoway, Hou, and Xia, 2014). Moreover, there are natural bounds of the strength of a biological synapse, measured by the amplitude of the postsynaptic potential, because the number of neurotransmitter vesicles and release sites is physically limited (Del Castillo and Katz, 1954). These natural limits are approximately 0.4 mV to 20 mV (Diaz-Rios and Miller, 2006).

3.4 OPTIMIZER

For simplicity, in this report weight updates are described using stochastic gradient descent:

$$\Delta W_{rji} = -\eta \sum_t L^t_{rj} \cdot \bar{e}^t_{rji}. \tag{3.47}$$

However, the results described in Section 4.1 are obtained using Adam (or Adaptive Moment Estimation) (Kingma and Ba, 2014). This optimization method tracks running averages of the gradient and its second moment (resp. M_{rji} and V_{rji}), and fits the local and online constraints of e-prop, because the running averages are tracked per individual synapse. The Adam weight update in the context of multi-layer e-prop is given by:

$$M^{(i+1)}_{rji} = \beta_1 M^{(i)}_{rji} + (1-\beta_1)\, G^{(i)}_{rji} \tag{3.48}$$

$$V^{(i+1)}_{rji} = \beta_2 V^{(i)}_{rji} + (1-\beta_2) \left(G^{(i)}_{rji}\right)^2 \tag{3.49}$$

$$\widehat{M}_{rji} = \frac{M^{(i+1)}_{rji}}{1-\beta_1^{i+1}} \tag{3.50}$$

$$\widehat{V}_{rji} = \frac{V^{(i+1)}_{rji}}{1-\beta_2^{i+1}} \tag{3.51}$$

$$\Delta W^{(i+1)}_{rji} = -\eta \frac{\widehat{M}_{rji}}{\sqrt{\widehat{V}_{rji}} + 10^{-5}}, \tag{3.52}$$

where $G^{(i)}$ is the estimated gradient $\sum_t L^t_{rj} \cdot \bar{e}^t_{rji}$ at weight update i, and $\beta_1 = 0.9$ and $\beta_2 = 0.999$ are the forgetting factors for the gradient and its second moment, respectively. The firing rate and L2 regularization terms are omitted here for clarity. Note that the forgetting factors are not indexed, but raised to the power of $i+1$, in computing the bias-corrected estimates $\widehat{M}_{rji}$ and $\widehat{V}_{rji}$. Note also that minibatches of size 32

are used to more accurately estimate the gradient and enable a stabler descent in the error landscape. The value of $G_{rji}^{(i)}$ is computed as the mean over the minibatch.

Note that the learning rate is linearly ramped up from 0 to η during the first epoch, such that the initial minibatches are used to aggregate good initial momentum buffers, as the variance is higher when fewer minibatches are processed. This "warming up" of the learning rate is a variance reduction technique that has shown beneficial results in training other models (L. Liu et al., 2019). Empirical observations on the resulting learning curves (see Section 4.1) suggest that this procedure does not hamper a rapid initial decrease of the loss function.

4

DISCUSSION

In this chapter, the learning performance and regularization behavior of the ALIF, STDP-ALIF, and Izhikevich neurons are compared and discussed. Then, the effect of stacking multiple recurrent layers on the learning performance and speed is examined. Next, possible future research avenues are discussed.

4.1 RESULTS

Figure 4.1 shows a typical classification result of a full validation sentence.

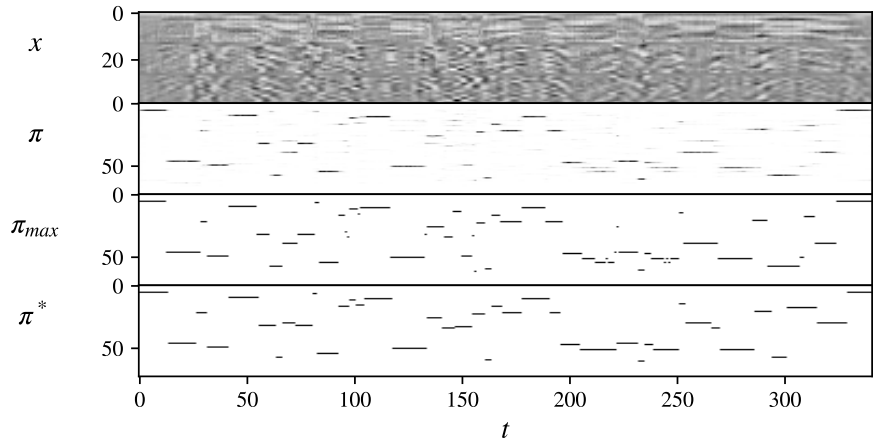


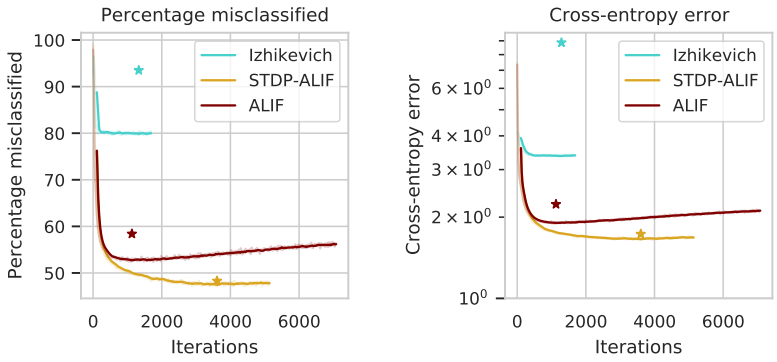
Figure 4.1: An example validation result using a trained ALIF model. The plot in the top row shows the standardized MFCC frames including its first and second derivatives of a sentence changing over time. The plot in the second row shows the probability distributions π of the frame-wise outputs of the model. The plot in the third row indicates the predicted phone $\pi_{\max}$ per frame. The plot in the last row shows the target phones π^*.

4.1.1 *Comparing neuron models*

ACCURACY The main outcome of the neuron model comparison is that in these results, the STDP-ALIF neuron outperforms the ALIF and corrected Izhikevich neuron models in classifying phones in the TIMIT dataset. This suggests that including STDP-like behavior to the ALIF neuron results in a better learning performance, answering the primary research objective posed in Chapter 1. Furthermore, the ALIF and STDP-ALIF neuron perform better than the Izhikevich e-prop neuron model. In Figure 4.2a the Izhikevich neuron reaches a

misclassification rate of 93.8% on the test set, which is only slightly better than constantly predicting the most frequent class. The ALIF neuron model reaches a test misclassification rate of 58.4% in relatively few iterations, after which validation performance starts to decrease. The STDP-ALIF neuron model scores best, reaching a performance of 48.3% after approximately 3500 iterations, suggesting that the addition of the STDP mechanism to the ALIF neuron improves the classification performance. Furthermore, the STDP-ALIF model does not show signs of overfitting as much as the ALIF neuron such as a decreasing validation performance in Figure 4.2a. However, the Izhikevich neuron shows STDP behavior too but performs poorly, suggesting that STDP by itself does not necessarily constitute a well-performing neuron model. The STDP-ALIF neuron may instead work by virtue of another factor, such as its better spike frequency adaptation compared to the Izhikevich neuron model. Note that the test performance was obtained from the model with the best validation accuracy (the used hyperparameters are listed in Table A.2).

Figure 4.2b illustrates the decrease of the cross-entropy score, which for the ALIF and STDP-ALIF neurons is comparable to that of the misclassification rate. The cross-entropy and classification performance of the Izhikevich neuron stalls relatively quickly at poor levels, suggesting that it trains its bias toward more frequent phone classes in the training data rather than learning a general relationship between input MFCCs and classes.



(a) Percentage of samples wrongly classified.

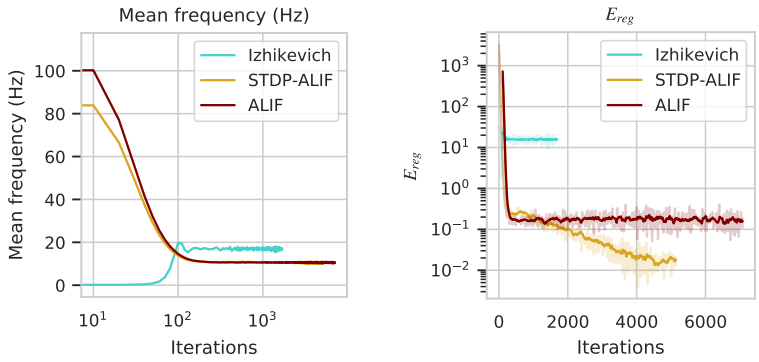
(b) Cross-entropy loss (log-scaled).

Figure 4.2: Classification performance on the validation data for each of the three neuron models in a single-layer e-prop model. The opaque lines indicate the running average of the real validation scores indicated by the transparent lines. The star symbols indicate the performances on the test set, with a misclassification rate of 93.5% for the Izhikevich neuron, 58.4% for the ALIF neuron, and 48.3% for the STDP-ALIF neuron type.

FIRING RATE Figure 4.3a illustrates the effect of the firing regularization term. It can be observed that the ALIF and STDP-ALIF neuron

models are able to quickly modulate their mean spiking frequencies to the desired target frequency of 10 Hz, but the Izhikevich neuron overshoots to a mean spiking frequency of approximately 18 Hz.

Figure 4.3b illustrates the decrease of the regularization error. The regularization error of the Izhikevich and ALIF neuron models quickly converges to fluctuate around a constant value, whereas that of the STDP-ALIF neuron model continues to decrease over time, even after the mean spiking frequency and classification performance have both converged to a plateau.



(a) Mean spiking frequency. Note the logarithmic horizontal axis.

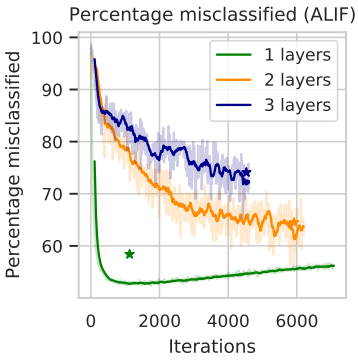
(b) Regularization error. Note the logarithmic vertical axis.

Figure 4.3: Effect of firing rate regularization on the validation data for each of the three neuron models.

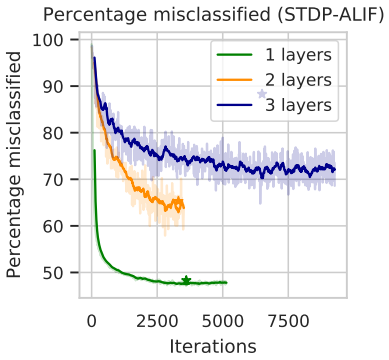
4.1.2 *Comparing network depth*

The comparison between the network depth in Figures 4.4a–4.4c suggests that single-layer e-prop networks train considerably more efficiently and accurately than multi-layer e-prop networks and show less variance among validation runs. This holds for all tested neuron types. The cross-entropy error, spiking frequency, and regularization error are also better for single-layer networks (see Figure A.1).

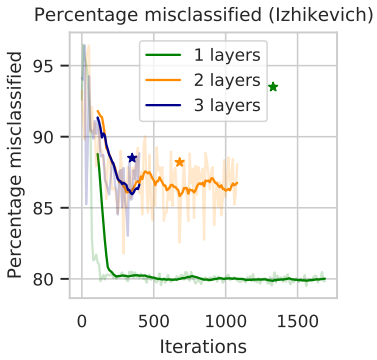
Therefore, rearranging the neurons into a stacked architecture does not appear to improve the classification performance, answering the secondary research objective posed in Chapter 1. In particular, it appears to diminishes the learning speed to a significant extent and render multi-layer e-prop architectures more inefficient than single-layer ones. However, it is not certain that multi-layer architectures are necessarily worse—they train more slowly, but in this report the performance was still improving when their training runs were interrupted due to practical limits. Therefore, particularly for the ALIF neuron, the multi-layer networks might outperform the single-layer networks if future work where low computing power and energy costs are not a priority.



(a) ALIF model.



(b) STDP-ALIF model.



(c) Izhikevich model.

Figure 4.4: Accuracy comparison on the validation data between single- and multi-layer e-prop models.

4.2 POSSIBLE IMPROVEMENTS

There are many hypothetical ways of improving the performance or biological plausibility of e-prop that have not yet been considered in this report. For instance, a likely reason that the learning speed of the multi-layer architectures was slower than their single-layer counterparts is that the weights are poorly initialized. Empirical observation of the learning process suggested that during early epochs, spiking activity faded in deeper layers, because the spiking activity from a preceding layer is generally weaker than the input values the first layer receives. Higher weights in-between layers mitigate this fading activity, but require some search to find a good value. In this report, the firing rate regularization term approximated this value, but learning is more efficient with a better initialization, since initial synaptic weights significantly affect the performance of STDP-based SNNs (J. Kim et al., 2020).

Also in this report, certain parameters such as firing rate targets (f^{target}), activity leak (α), and feedback signals were constant for all neurons, except the threshold adaptivity (β), which was 0 for a randomly selected 25% of the neurons to emulate non-adaptive LIF neurons. Future research could examine the effects of sampling some of these parameters

from a distribution for each neuron, thereby creating a more diverse population of neurons with different time scales. This sampling requires a careful assessment of the time scales related to the learning task; in particular, the network should be able to process the slowest relevant time scale of the task (Jaeger et al., 2021). In the TIMIT classification task, for instance, this could span the whole time fragment, because initial words give semantic context to all subsequent words. However, a more predictive time scale for the TIMIT dataset is on the scale of approximately 8 MFCC frames, which is now accurately captured by the adaptive threshold component in the ALIF neuron model. Temporal dependencies can also be found when moving in the opposite direction—later words and sounds give informative context to earlier words and sounds. In Bellec, Scherr, Subramoney, et al. (2020), this context was captured in a bidirectional network, improving the accuracy by nearly 15%. In this report, this was empirically validated as well, but left undiscussed because a bidirectional network is not biologically plausible, as directly accessing future input values violates the "online" constraint. According to Bellec, Scherr, Subramoney, et al. (2020), e-prop suggests that the experimentally found diverse time constants of the firing activity of populations of neurons in different brain areas (Runyan et al., 2017) are correlated with their capability to handle corresponding ranges of delays in temporal credit assignment for learning. Setting different values for these parameters per layer might also have a beneficial effect; Ahmed et al. (1998) suggested that deeper layers display slower and weaker adaptation rates than early layers.

In the brain, neurons primarily tend to connect to nearby neurons. This suggests that the effects of the topology within a layer might positively affect the learning process. A simple lattice topology might better approximate the connectivity of the brain, decrease the computational complexity in emulations in von Neumann machines, and allow easier on-chip implementations in neuromorphic hardware. Hierarchical clustering of neurons might also have a beneficial effect, as this has been demonstrated to improve R-STDP in SNNs (Weidel, Duarte, and Morrison, 2021) and address the scalability issue of SNNs (Carrillo et al., 2012). Because a neuromorphic system can support complex network operations (Hasler and Akers, 1990), large-scale conductance-based SNNs (Yang, B. Deng, et al., 2019; Yang, J. Wang, et al., 2019) and asynchronous communication in VLSIs through address-event-representation ("A Pulse-Coded Communications Infrastructure for Neuromorphic Systems" 1998; Lazzaro et al., 1993) might be suitable to further customize the connectivity graph of an e-prop architecture.

Other exciting research avenues include connectivity graphs that change over time through a dynamic pruning and growing of weights and neurons. Here, the biological motivation is that the human brain prunes synaptic connections during early development (Huttenlocher et al., 1979). Elbez et al. (2020) demonstrated that 75% of a SNN can be compressed while preserving its performance, but it is not clear if this can be applied in a biologically plausible way in the e-prop framework. However, integrating stochastic synaptic rewiring (Kappel et al., 2018)

into an ALIF network can improve its short-term memory (Bellec, Scherr, Subramoney, et al., 2020).

Finally, synaptic delay might improve the temporal processing power of an e-prop model. In this report, communication between neurons was transmitted as a spike over a synapse with a delay of 1 ms. This delay could differ among synapses, such that potentially informative past inputs are more accurately preserved in synaptic delays, rather than only in eligibility traces and activity loops. This can help deal with tasks that require processing information on multiple time spans (Jaeger et al., 2021). This resembles the variable physical length of myelinated biological synapses and the number of nodes of Ranvier along them, affecting the conductance of the action potential (Bean, 2007).

4.3 FUTURE DIRECTIONS

As neuromorphic computing matures, neuroscience improves, and DL increasingly hits fundamental limitations, there is an exciting future for biologically plausible SNNs. There is much to gain from cross-fertilization between these fields. The popularity of DL was accelerated by accessible platforms to implement and deploy ANNs. Similar high-level simulation platforms are now in active development, which can integrate the typical behavior of memristive device models into crossbar architectures within DL systems (Lammie et al., 2020).

Recent advances in neuromorphic computing indicate this increasing popularity. Neuromorphic architectures have been used for mapless navigation with 75 times lower power and better performance (Tang, Kumar, and Michmizos, 2020); as low-power solutions for simultaneous localization and mapping of mobile robots (Tang, Shah, and Michmizos, 2019), for planning (Fischl et al., 2017), and control (Blum et al., 2017); and self-repairing SNN for fault detection (Zhu et al., 2017). While cross-fertilization between neuromorphic computing and quantum computing is starting to take place (Russek et al., 2016), as quantum superposition and entanglement can be used to process information in parallel and in a high-dimensional state space (Fujii and Nakajima, 2017; Tacchino et al., 2019; Yamamoto et al., 2017), more physics and materials science is required to build efficient neuromorphic architectures (Marković et al., 2020). The same holds for the cross-fertilization between neuroscience and learning rules of biologically plausible SNNs. Nanodevices that emulate biological synapses with learning functions can benefit neuromorphic architectures (Ren et al., 2018; Y. Wang et al., 2018; Yao et al., 2017), particularly the two-terminal memristor (Jo et al., 2010; Z. Wang et al., 2017). However, it has been argued that the learning principles of biological NNs are not explored enough to design engineering solutions (Gorban, Makarov, and Tyukin, 2019; Taherkhani et al., 2018). Feedback connections, for which the brain uses neurotransmitters, may become particularly problematic in large-scale neuromorphic systems. Another issue in analog computation is how to match the system's internal temporal processing to that of its inputs. Emulating neural dynamics on

a physical substrate is more efficient but requires constraints to match the brain's timescales (Jaeger et al., 2021; Mead, 1990). Future work on e-prop could explore a combination with attention-based models in order to cover multiple timescales (Bellec, Scherr, Subramoney, et al., 2020).

5 CONCLUSION

As deep learning models require increasing amount of energy, the upcoming neuromorphic computing paradigm offers various ways to more efficiently run spiking neural networks. Spiking neural networks can learn to perform tasks with a good performance and low energy requirements, but there is no established learning algorithm yet. In this paper, the e-prop learning algorithm for recurrent spiking neural networks was combined with STDP, which is a major component of biological learning.

In this report, the e-prop framework was applied on the TIMIT phone classification task, meeting the objectives listed in Chapter 1. First, the performance of the ALIF neuron was reproduced using the explicit e-prop equations. Next, the STDP-LIF neuron was modified to the STDP-ALIF model that was experimentally verified to outperform the ALIF neuron on the TIMIT learning task. Also, the Izhikevich neuron, which also shows STDP behavior, was shown to be unstable and performing worse than the ALIF and STDP-ALIF neurons. This suggests that STDP does not provide an adequate neuron model by itself, but that e.g. spike frequency adaptation also needs to be taken into account. However, enhancing an already well-performing neuron model to display STDP-like properties can improve the performance.

Finally, the effect of stacking multiple layers was also examined in combination with the ALIF, STDP-ALIF, and Izhikevich neuron model, and did not appear to improve the learning performance in this task.

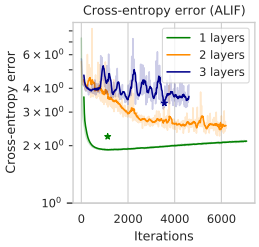
Possible future work on this topic includes research on the effects of more elaborate weight initialization methods, variable hyperparameters for individual neurons, different static or dynamic connectivity graphs, and synaptic delays.

The scientific gain of this research is that the link between STDP and e-prop was more closely examined than in previous literature, and that the inclusion of STDP in e-prop can lead to a more accurate or efficient learning performance. E-prop combined with the STDP-ALIF neuron model remains a framework that offers high potential for biologically plausible learning algorithms for SNNs, which can be particularly well-suited for replicating intelligent behavior in low-power neuromorphic hardware.

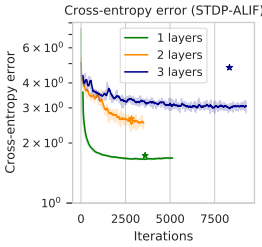
APPENDIX

A.1 IMPLEMENTATION

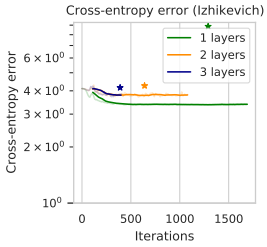
The code is available at `https://github.com/wkvanderveen/maspro`. It does not require any machine learning libraries (except basic PyTorch to initialize arrays on the GPU); all computations are explicitly implemented in pure multidimensional NumPy-like arrays. The code contains a configuration file to set various options for the MFCC preprocessing and e-prop process, including the number of layers, whether to use an uni- or bidirectional network (a legacy option not treated in this report), and the window size in the MFCCs, for example.



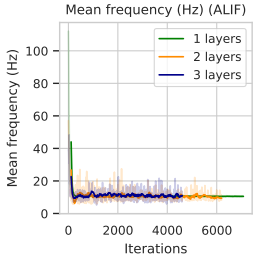
(a) ALIF cross-entropy.



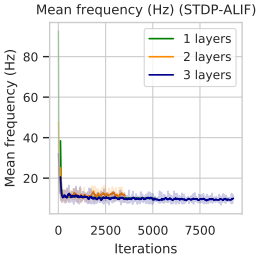
(b) STDP-ALIF cross-entropy.



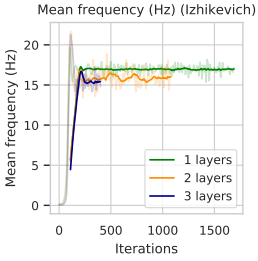
(c) Izhikevich cross-entropy.



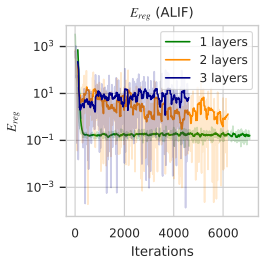
(d) ALIF spike rate.



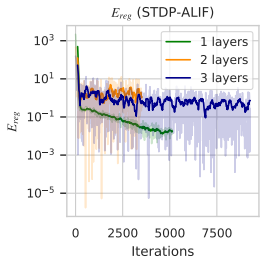
(e) STDP-ALIF spike rate.



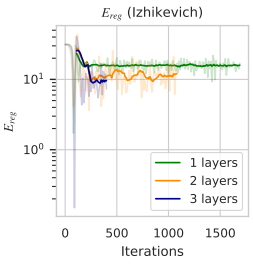
(f) Izhikevich spike rate.



(g) ALIF reg. error.



(h) STDP-ALIF reg. error.



(i) Izhikevich reg. error.

Figure A.1: Cross-entropy rates, mean spiking frequencies, and regularization errors for multi-layer networks.

MELS	HZ	FILTERBANK
0	0	0
105	68.5	2
210	143.7	4
315	226.2	7
420	316.8	10
525	416.3	13
630	525.5	16
735	645.4	20
840	777	24
945	921.5	29
1050	1080.1	34
1155	1254.4	40
1260	1445.4	46
1365	1655.3	53
1470	1885.7	60
1575	2138.6	68
1680	2416.3	77
1785	2721.2	87
1890	3055.9	97
1995	3423.3	109
2100	3826.7	122
2205	4269.5	136
2310	4755.7	152
2415	5289.4	169
2520	5875.3	188
2625	6518.6	209
2730	7224.8	231
2835	8000	256

Table A.1: Conversion table between linearly spaced Mels and their corresponding frequencies and filterbank boundaries.

SYMBOL	DESCRIPTION	VALUE
α	Activity leak	0.8
β	Adaptivity	0.184
ρ	Adaptivity leak	0.975
κ	Output decay	0.8
γ	Pseudoderivative dampening	0.3
v_{th}	Base threshold	0.95
δt_{ref}	Refractory time	2
η	Learning rate	0.01
β_1	Adam momemtum factor 1	0.9
β_2	Adam momemtum factor 2	0.999
c_{reg}	Firing rate regularization	50
c_{L2}	L2 regularization	10^{-5}
f^{target}	Target firing rate	0.01
N	Network size	800*

Table A.2: The full list of hyperparameter values, used in all networks and neuron model types.
*This is the total number of neurons in multi-layer networks (400 per layer in 2-layer networks, and 266 in 3-layer networks).

BIBLIOGRAPHY

A Pastur-Romay, L et al. (2017). "Parallel computing for brain simulation." In: *Current Topics in Medicinal Chemistry* 17.14, pp. 1646–1668.

"A Pulse-Coded Communications Infrastructure for Neuromorphic Systems" (Nov. 1998). In: *Pulsed Neural Networks*. The MIT Press. ISBN: 9780262278768. DOI: 10.7551/mitpress/5704.003.0011. eprint: https://direct.mit.edu/book/chapter-pdf/177002/9780262278768_caf.pdf. URL: https://doi.org/10.7551/mitpress/5704.003.0011.

Abbott, Larry F and Sacha B Nelson (2000). "Synaptic plasticity: taming the beast." In: *Nature neuroscience* 3.11, pp. 1178–1183.

Ahmed, B et al. (1998). "Estimates of the net excitatory currents evoked by visual stimulation of identified neurons in cat visual cortex." In: *Cerebral cortex (New York, NY: 1991)* 8.5, pp. 462–476.

Azevedo, Frederico AC et al. (2009). "Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain." In: *Journal of Comparative Neurology* 513.5, pp. 532–541.

Bailey, Craig H et al. (2000). "Is heterosynaptic modulation essential for stabilizing hebbian plasiticity and memory." In: *Nature Reviews Neuroscience* 1.1, pp. 11–20.

Bao, Shaowen, Vincent T Chan, and Michael M Merzenich (2001). "Cortical remodelling induced by activity of ventral tegmental dopamine neurons." In: *Nature* 412.6842, pp. 79–83.

Barnes, Terra D et al. (2005). "Activity of striatal neurons reflects dynamic encoding and recoding of procedural memories." In: *Nature* 437.7062, pp. 1158–1161.

Bartunov, Sergey et al. (2018). "Assessing the scalability of biologically-motivated deep learning algorithms and architectures." In: *Advances in neural information processing systems*, pp. 9368–9378.

Bean, Bruce P (2007). "The action potential in mammalian central neurons." In: *Nature Reviews Neuroscience* 8.6, pp. 451–465.

Bear, Mark, Barry Connors, and Michael A Paradiso (2020). *Neuroscience: Exploring the brain*. Jones & Bartlett Learning, LLC.

Bellec, Guillaume, Darjan Salaj, et al. (2018). "Long short-term memory and learning-to-learn in networks of spiking neurons." In: *arXiv preprint arXiv:1803.09574*.

Bellec, Guillaume, Franz Scherr, Elias Hajek, et al. (2019). "Biologically inspired alternatives to backpropagation through time for learning in recurrent neural nets." In: *arXiv preprint arXiv:1901.09049*.

Bellec, Guillaume, Franz Scherr, Anand Subramoney, et al. (2020). "A solution to the learning dilemma for recurrent networks of spiking neurons." In: *Nature Communications* 11.1, pp. 1–15.

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

Bhalla, Upinder S (2014). "Molecular computation in neurons: a modeling perspective." In: *Current opinion in neurobiology* 25, pp. 31–37.

Blum, Hermann et al. (2017). "A neuromorphic controller for a robotic vehicle equipped with a dynamic vision sensor." In: *Robotics Science and Systems, RSS 2017*.

Bohte, Sander M, Joost N Kok, and Han La Poutre (2002). "Error-backpropagation in temporally encoded networks of spiking neurons." In: *Neurocomputing* 48.1-4, pp. 17–37.

Caporale, Natalia and Yang Dan (2008). "Spike timing–dependent plasticity: a Hebbian learning rule." In: *Annu. Rev. Neurosci.* 31, pp. 25–46.

Carrillo, Snaider et al. (2012). "Scalable hierarchical network-on-chip architecture for spiking neural network hardware implementations." In: *IEEE Transactions on Parallel and Distributed Systems* 24.12, pp. 2451–2461.

Cassenaer, Stijn and Gilles Laurent (2012). "Conditional modulation of spike-timing-dependent plasticity for olfactory learning." In: *Nature* 482.7383, pp. 47–52.

Clopath, Claudia et al. (2010). "Connectivity reflects coding: a model of voltage-based STDP with homeostasis." In: *Nature Neuroscience* 13.3, p. 344.

Dang, Mai T et al. (2006). "Disrupted motor learning and long-term synaptic plasticity in mice lacking NMDAR1 in the striatum." In: *Proceedings of the National Academy of Sciences* 103.41, pp. 15254–15259.

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

Del Castillo, J and B3 Katz (1954). "Quantal components of the end-plate potential." In: *The Journal of physiology* 124.3, pp. 560–573.

Diaz-Rios, Manuel and Mark W Miller (2006). "Target-specific regulation of synaptic efficacy in the feeding central pattern generator of Aplysia: potential substrates for behavioral plasticity?" In: *The Biological Bulletin* 210.3, pp. 215–229.

Diehl, Peter U and Matthew Cook (2015). "Unsupervised learning of digit recognition using spike-timing-dependent plasticity." In: *Frontiers in Computational Neuroscience* 9, p. 99.

Dong, Meng, Xuhui Huang, and Bo Xu (2018). "Unsupervised speech recognition through spike-timing-dependent plasticity in a convolutional spiking neural network." In: *PloS One* 13.11, e0204596.

Drachman, David A (2005). *Do we have brain to spare?* AAN Enterprises.

Drubach, Daniel (2000). *The brain explained*. Prentice Hall.

Elbez, Hammouda et al. (June 2020). "Progressive Compression and Weight Reinforcement for Spiking Neural Networks." working paper or preprint. URL: `https://hal.archives-ouvertes.fr/hal-02737057`.

Engelhard, Ben et al. (2019). "Specialized coding of sensory, motor and cognitive variables in VTA dopamine neurons." In: *Nature* 570.7762, pp. 509–513.

Erickson, Jeffrey D et al. (2006). "Activity-dependent regulation of vesicular glutamate and GABA transporters: a means to scale quantal size." In: *Neurochemistry international* 48.6-7, pp. 643–649.

Escobar, Maria-Jose et al. (2009). "Action recognition using a bio-inspired feedforward spiking network." In: *International Journal of Computer Vision* 82.3, p. 284.

Esser, Steven K et al. (2016). "Convolutional networks for fast, energy-efficient neuromorphic computing." In: *Proceedings of the national academy of sciences* 113.41, pp. 11441–11446.

Fayek, Haytham M. (2016). *Speech processing for machine learning: filter banks, mel-frequency cepstral coefficients (MFCCs) and what's in-between.* URL: `https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html`.

Fischl, Kate D et al. (2017). "Path planning on the truenorth neurosynaptic system." In: *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, pp. 1–4.

Florian, Răzvan V (2007). "Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity." In: *Neural computation* 19.6, pp. 1468–1502.

Frémaux, Nicolas and Wulfram Gerstner (2016). "Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules." In: *Frontiers in neural circuits* 9, p. 85.

Fujii, Keisuke and Kohei Nakajima (2017). "Harnessing disordered-ensemble quantum dynamics for machine learning." In: *Physical Review Applied* 8.2, p. 024030.

Fukushima, Kunihiko and Sei Miyake (1982). "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition." In: *Competition and Cooperation in Neural Nets*. Springer, pp. 267–285.

Gallicchio, Claudio (2018). "Short-term memory of deep RNN." In: *arXiv preprint arXiv:1802.00748*.

Gallicchio, Claudio, Alessio Micheli, and Luca Pedrelli (2017). "Deep reservoir computing: A critical experimental analysis." In: *Neurocomputing* 268, pp. 87–99.

Garofolo, John S et al. (1993). "DARPA TIMIT acoustic-phonetic continous speech corpus CD-ROM. NIST speech disc 1-1.1." In: *STIN* 93, p. 27403.

Gerstner, Wulfram and Werner M Kistler (2002). *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge University Press.

Gerstner, Wulfram, Marco Lehmann, et al. (2018). "Eligibility traces and plasticity on behavioral time scales: experimental support of neohebbian three-factor learning rules." In: *Frontiers in Neural Circuits* 12, p. 53.

Gorban, Alexander N, Valeri A Makarov, and Ivan Y Tyukin (2019). "The unreasonable effectiveness of small neural ensembles in high-dimensional brain." In: *Physics of life reviews* 29, pp. 55–88.

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.

Graves, Alex and Jürgen Schmidhuber (2005). "Framewise phoneme classification with bidirectional LSTM and other neural network architectures." In: *Neural Networks* 18.5-6, pp. 602–610.

Hasler, P and LA Akers (1990). "VLSI neural systems and circuits." In: *Ninth Annual International Phoenix Conference on Computers and Communications. 1990 Conference Proceedings*. IEEE, pp. 31–37.

He, Kaiming et al. (2016). "Deep residual learning for image recognition." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

He, Kaiwen et al. (2015). "Distinct eligibility traces for LTP and LTD in cortical synapses." In: *Neuron* 88.3, pp. 528–538.

Hebb, Donald Olding (1949). "The organization of behavior; a neuropsychological theory." In: *A Wiley Book in Clinical Psychology* 62, p. 78.

Hermans, Michiel and Benjamin Schrauwen (2013). "Training and analysing deep recurrent neural networks." In: *Advances in neural information processing systems* 26, pp. 190–198.

Hong, Shen et al. (2010). "A cooperative method for supervised learning in spiking neural networks." In: *The 2010 14th International Conference on Computer Supported Cooperative Work in Design*. IEEE, pp. 22–26.

Hopfield, John J (1982). "Neural networks and physical systems with emergent collective computational abilities." In: *Proceedings of the National Academy of Sciences* 79.8, pp. 2554–2558.

Huang, Shiyong et al. (2014). "Associative Hebbian synaptic plasticity in primate visual cortex." In: *Journal of Neuroscience* 34.22, pp. 7575–7579.

Hubel, David H and Torsten N Wiesel (1968). "Receptive fields and functional architecture of monkey striate cortex." In: *The Journal of physiology* 195.1, pp. 215–243.

Huh, Dongsung and Terrence J Sejnowski (2017). "Gradient descent for spiking neural networks." In: *arXiv preprint arXiv:1706.04698*.

Hultzsch, Eugen et al. (1964). *Tables of transitional frequencies of English phonemes*. Urbana: University of Illinois Press.

Huttenlocher, Peter R et al. (1979). "Synaptic density in human frontal cortex-developmental changes and effects of aging." In: *Brain Res* 163.2, pp. 195–205.

Izhikevich, Eugene M (2003). "Simple model of spiking neurons." In: *IEEE Transactions on neural networks* 14.6, pp. 1569–1572.

Izhikevich, Eugene M (2007). "Solving the distal reward problem through linkage of STDP and dopamine signaling." In: *Cerebral cortex* 17.10, pp. 2443–2452.

Jaeger, Herbert et al. (2021). "Dimensions of Timescales in Neuromorphic Computing Systems." In: *arXiv preprint arXiv:2102.10648*.

Jo, Sung Hyun et al. (2010). "Nanoscale memristor device as synapse in neuromorphic systems." In: *Nano Letters* 10.4, pp. 1297–1301.

Kaiser, Jacques, Hesham Mostafa, and Emre Neftci (2020). "Synaptic plasticity dynamics for deep continuous local learning (DECOLLE)." In: *Frontiers in Neuroscience* 14, p. 424.

Kandel, Eric R et al. (2000). *Principles of neural science*. Vol. 4. McGraw-hill New York.

Kappel, David et al. (2018). "A dynamic connectome supports the emergence of stable computational function of neural circuits through reward-based learning." In: *Eneuro* 5.2.

Kheradpisheh, Saeed Reza et al. (2018). "STDP-based spiking deep convolutional neural networks for object recognition." In: *Neural Networks* 99, pp. 56–67.

Kim, Chul-Heung et al. (2018). "Demonstration of unsupervised learning with spike-timing-dependent plasticity using a TFT-type NOR flash memory array." In: *IEEE Transactions on Electron Devices* 65.5, pp. 1774–1780.

Kim, Jangsaeng et al. (2020). "Initial synaptic weight distribution for fast learning speed and high recognition rate in STDP-based spiking neural network." In: *Solid-State Electronics* 165, p. 107742.

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

Kucera, Henry, Henry Kučera, and Winthrop Nelson Francis (1967). *Computational analysis of present-day American English*. Brown University Press.

Labov, William, Sharon Ash, and Charles Boberg (2008). *The atlas of North American English: Phonetics, phonology and sound change*. Walter de Gruyter.

Lammie, Corey et al. (2020). "MemTorch: An open-source simulation framework for memristive deep learning systems." In: *arXiv preprint arXiv:2004.10971*.

Lazzaro, John et al. (1993). "Silicon auditory processors as computer peripherals." In: *IEEE Transactions on Neural Networks* 4.3, pp. 523–528.

LeCun, Yann, Yoshua Bengio, et al. (1995). "Convolutional networks for images, speech, and time series." In: *The Handbook of Brain Theory and Neural Networks* 3361.10, p. 1995.

Lee, J-C and Bing J Sheu (1990). "Parallel digital image restoration using adaptive VLSI neural chips." In: *Proceedings., 1990 IEEE International Conference on Computer Design: VLSI in Computers and Processors*. IEEE, pp. 126–129.

Lee, Jun Haeng, Tobi Delbruck, and Michael Pfeiffer (2016). "Training deep spiking neural networks using backpropagation." In: *Frontiers in neuroscience* 10, p. 508.

Legenstein, Robert, Dejan Pecevski, and Wolfgang Maass (2008). "A learning theory for reward-modulated spike-timing-dependent plasticity with application to biofeedback." In: *PLoS Comput Biol* 4.10, e1000180.

Li, Shaomin et al. (2003). "Dopamine-dependent facilitation of LTP induction in hippocampal CA1 by exposure to spatial novelty." In: *Nature neuroscience* 6.5, pp. 526–531.

Lillicrap, Timothy P, Daniel Cownden, et al. (2016). "Random synaptic feedback weights support error backpropagation for deep learning." In: *Nature Communications* 7.1, pp. 1–10.

Lillicrap, Timothy P and Adam Santoro (2019). "Backpropagation through time and the brain." In: *Current opinion in neurobiology* 55, pp. 82–89.

Liu, Daqi and Shigang Yue (2017). "Fast unsupervised learning for visual pattern recognition using spike timing dependent plasticity." In: *Neurocomputing* 249, pp. 212–224.

Liu, Liyuan et al. (2019). "On the variance of the adaptive learning rate and beyond." In: *arXiv preprint arXiv:1908.03265*.

Lobo, Jesus L et al. (2020). "Spiking neural networks and online learning: An overview and perspectives." In: *Neural Networks* 121, pp. 88–100.

Lobov, Sergey A et al. (2020). "Spatial properties of STDP in a self-learning spiking neural network enable controlling a mobile robot." In: *Frontiers in Neuroscience* 14, p. 88.

Lukoševičius, Mantas and Herbert Jaeger (2009). "Reservoir computing approaches to recurrent neural network training." In: *Computer Science Review* 3.3, pp. 127–149.

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

Marković, Danijela et al. (2020). "Physics for neuromorphic computing." In: *Nature Reviews Physics* 2.9, pp. 499–510.

Mead, Carver (1990). "Neuromorphic electronic systems." In: *Proceedings of the IEEE* 78.10, pp. 1629–1636.

Merolla, Paul A et al. (2014). "A million spiking-neuron integrated circuit with a scalable communication network and interface." In: *Science* 345.6197, pp. 668–673.

Mitra, Srinjoy, Stefano Fusi, and Giacomo Indiveri (2008). "Real-time classification of complex patterns using spike-based learning in neuromorphic VLSI." In: *IEEE Transactions on Biomedical Circuits and Systems* 3.1, pp. 32–42.

Monroe, Don (June 2014). "Neuromorphic Computing Gets Ready for the (Really) Big Time." In: *Commun. ACM* 57.6, pp. 13–15. ISSN: 0001-0782. DOI: 10.1145/2601069. URL: https://doi.org/10.1145/2601069.

Neftci, Emre O (2018). "Data and power efficient intelligence with neuromorphic learning machines." In: *Iscience* 5, pp. 52–68.

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

Nieuwenhuis, Sander et al. (2001). "Error-related brain potentials are differentially related to awareness of response errors: Evidence from an antisaccade task." In: *Psychophysiology* 38.5, pp. 752–760.

Nøkland, Arild (2016). "Direct feedback alignment provides learning in deep neural networks." In: *arXiv preprint arXiv:1609.01596*.

Ourdighi, Asmaa and Abdelkader Benyettou (2016). "An efficient spiking neural network approach based on spike response model for breast cancer diagnostic." In: *Int. Arab J. Inf. Technol.* 13.6B, pp. 1032–1038.

Pokorny, Christoph et al. (2020). "STDP forms associations between memory traces in networks of spiking neurons." In: *Cerebral Cortex* 30.3, pp. 952–968.

Porr, Bernd and Florentin Wörgötter (2007). "Learning with "relevance": using a third factor to stabilize hebbian learning." In: *Neural computation* 19.10, pp. 2694–2719.

Purves, Dale (2008). "Neuroscience (4th ed.)" In: ed. by George J Augustine et al. Sinauer Associates.

Rajendran, Bipin et al. (2019). "Low-power neuromorphic hardware for signal processing applications: A review of architectural and system-level design approaches." In: *IEEE Signal Processing Magazine* 36.6, pp. 97–110.

Ren, Yanyun et al. (2018). "Analytical modeling of organic–inorganic CH3NH3PbI3 Perovskite resistive switching and its application for Neuromorphic recognition." In: *Advanced Theory and Simulations* 1.4, p. 1700035.

Reynolds, John NJ, Brian I Hyland, and Jeffery R Wickens (2001). "A cellular mechanism of reward-related learning." In: *Nature* 413.6851, pp. 67–70.

Reynolds, John NJ and Jeffery R Wickens (2002). "Dopamine-dependent plasticity of corticostriatal synapses." In: *Neural Networks* 15.4-6, pp. 507–521.

Roeper, Jochen (2013). "Dissecting the diversity of midbrain dopamine neurons." In: *Trends in Neurosciences* 36.6, pp. 336–342.

Rosenblatt, Frank (1958). "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6, p. 386.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). "Learning representations by back-propagating errors." In: *nature* 323.6088, pp. 533–536.

Runyan, Caroline A et al. (2017). "Distinct timescales of population coding across cortex." In: *Nature* 548.7665, pp. 92–96.

Russek, Stephen E et al. (2016). "Stochastic single flux quantum neuromorphic computing using magnetically tunable Josephson junctions." In: *2016 IEEE International Conference on Rebooting Computing (ICRC)*. IEEE, pp. 1–5.

Sacramento, João et al. (2018). "Dendritic cortical microcircuits approximate the backpropagation algorithm." In: *arXiv preprint arXiv:1810.11393*.

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.

Sanhueza, Magdalena and John Lisman (2013). "The CaMKII/NMDAR complex as a molecular memory." In: *Molecular brain* 6.1, pp. 1–8.

Schmidhuber, Jürgen (2015). "Deep learning in neural networks: An overview." In: *Neural networks* 61, pp. 85–117.

Schultz, Wolfram (2007). "Behavioral dopamine signals." In: *Trends in neurosciences* 30.5, pp. 203–210.

Schuman, Catherine D et al. (2017). "A survey of neuromorphic computing and neural networks in hardware." In: *arXiv preprint arXiv:1705.06963*.

Siddoway, Benjamin, Hailong Hou, and Houhui Xia (2014). "Molecular mechanisms of homeostatic synaptic downscaling." In: *Neuropharmacology* 78, pp. 38–44.

Smith, Julius O. (accessed December 2020). *Spectral Audio Signal Processing*. online book, 2011 edition. `http://ccrma.stanford.edu/~jos/sasp/`.

Sokoloff, Louis (1960). "The metabolism of the central nervous system in vivo." In: *Handbook of Physiology, section, I, Neurophysiology* 3, pp. 1843–64.

Soltic, Snjezana and Nikola Kasabov (2010). "Knowledge extraction from evolving spiking neural networks with rank order population coding." In: *International Journal of Neural Systems* 20.06, pp. 437–445.

Sterling, Peter and Simon Laughlin (2015). *Principles of neural design*. MIT Press.

Stevens, Stanley Smith, John Volkmann, and Edwin B Newman (1937). "A scale for the measurement of the psychological magnitude pitch." In: *The Journal of the Acoustical Society of America* 8.3, pp. 185–190.

Stolyarova, Alexandra (2018). "Solving the credit assignment problem with the prefrontal cortex." In: *Frontiers in Neuroscience* 12, p. 182.

Tacchino, Francesco et al. (2019). "An artificial neuron implemented on an actual quantum processor." In: *npj Quantum Information* 5.1, pp. 1–8.

Taherkhani, Aboozar et al. (2018). "A supervised learning algorithm for learning precise timing of multiple spikes in multilayer spiking neural networks." In: *IEEE transactions on neural networks and learning systems* 29.11, pp. 5394–5407.

Tang, Guangzhi, Neelesh Kumar, and Konstantinos P Michmizos (2020). "Reinforcement co-learning of deep and spiking neural networks for energy-efficient mapless navigation with neuromorphic hardware." In: *arXiv preprint arXiv:2003.01157*.

Tang, Guangzhi, Arpit Shah, and Konstantinos P Michmizos (2019). "Spiking neural network on neuromorphic hardware for energy-efficient unidimensional slam." In: *arXiv preprint arXiv:1903.02504*.

Tarassenko, Lionel et al. (1990). "Real-time autonomous robot navigation using VLSI neural networks." In: *Proceedings of the 3rd International Conference on Neural Information Processing Systems*, pp. 422–428.

Tavanaei, Amirhossein and Anthony Maida (2017). "Bio-inspired multilayer spiking neural network extracts discriminative features from speech signals." In: *International Conference on Neural Information Processing*. Springer, pp. 899–908.

Traub, Manuel et al. (2020). "Learning Precise Spike Timings with Eligibility Traces." In: *International Conference on Artificial Neural Networks*. Springer, pp. 659–669.

Turing, Alan Mathison (1948). *Intelligent machinery.*

Turrigiano, Gina G and Sacha B Nelson (2000). "Hebb and homeostasis in neuronal plasticity." In: *Current opinion in neurobiology* 10.3, pp. 358–364.

Von Neumann, John (1993). "First Draft of a Report on the EDVAC." In: *IEEE Annals of the History of Computing* 15.4, pp. 27–75.

Wang, Yan et al. (2018). "Photonic synapses based on inorganic perovskite quantum dots for neuromorphic computing." In: *Advanced Materials* 30.38, p. 1802883.

Wang, Zhongrui et al. (2017). "Memristors with diffusive dynamics as synaptic emulators for neuromorphic computing." In: *Nature Materials* 16.1, pp. 101–108.

Weidel, Philipp, Renato Duarte, and Abigail Morrison (2021). "Unsupervised learning and clustered connectivity enhance reinforcement learning in spiking neural networks." In: *Frontiers in Computational Neuroscience* 15, p. 18.

Whittington, James CR and Rafal Bogacz (2019). "Theories of error back-propagation in the brain." In: *Trends in Cognitive Sciences* 23.3, pp. 235–250.

Woźniak, Stanisław et al. (2020). "Deep learning incorporating biologically inspired neural dynamics and in-memory computing." In: *Nature Machine Intelligence* 2.6, pp. 325–336.

Xu, Yan et al. (2013). "A supervised multi-spike learning algorithm based on gradient descent for spiking neural networks." In: *Neural Networks* 43, pp. 99–113.

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.

Yamamoto, Yoshihisa et al. (2017). "Coherent Ising machines—Optical neural networks operating at the quantum limit." In: *npj Quantum Information* 3.1, pp. 1–15.

Yang, Shuangming, Bin Deng, et al. (2019). "Scalable digital neuromorphic architecture for large-scale biophysically meaningful neural network with multi-compartment neurons." In: *IEEE Transactions on Neural Networks and Learning Systems* 31.1, pp. 148–162.

Yang, Shuangming, J. Wang, et al. (2019). "Real-time neuromorphic system for large-scale conductance-based spiking neural networks." In: *IEEE Transactions on Cybernetics* 49, pp. 2490–2503.

Yao, Peng et al. (2017). "Face classification using electronic synapses." In: *Nature Communications* 8.1, pp. 1–8.

Zenke, Friedemann and Surya Ganguli (2018). "Superspike: Supervised learning in multilayer spiking neural networks." In: *Neural Computation* 30.6, pp. 1514–1541.

Zenke, Friedemann, Wulfram Gerstner, and Surya Ganguli (2017). "The temporal paradox of Hebbian learning and homeostatic plasticity." In: *Current Opinion in Neurobiology* 43, pp. 166–176.

Zenke, Friedemann and Emre O Neftci (2021). "Brain-inspired learning on neuromorphic substrates." In: *Proceedings of the IEEE.*

Zhang, Guohe et al. (2020). "A low-cost and high-speed hardware implementation of spiking neural network." In: *Neurocomputing* 382, pp. 106–115.

Zhu, Yuke et al. (2017). "Target-driven visual navigation in indoor scenes using deep reinforcement learning." In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 3357–3364.