ELIGIBILITY PROPAGATION IN MULTI-LAYER RECURRENT SPIKING NEURAL NETWORKS

WERNER VAN DER VEEN

Faculty of Science and Engineering
University of Groningen

June 2021 – classicthesis v4.6

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

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, 2009). One of the proposed methods was to emulate the network of biological neurons in the human brain using simple units named perceptrons (Rosenblatt, 1958), inspired by Hebbian learning (Hebb, 1949), which was a new (and later corroborated) theory on biological learning. This proved to be more difficult than initially hoped, partly due to ostensibly insurmountable problems such as the inability of perceptrons to learn linearly inseparable tasks (Minsky and Papert, 1969). As a consequence, excitement waned and funding dried up (Crevier, 1993), and the field focused on more practical applications, such as expert systems and symbolic reasoning (Haugeland, 1985). After the popularization in the 1980s of trainable Hopfield networks (Hopfield, 1982) and backpropagation (Rumelhart, 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, and the rise of the Internet has also provided abundant 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; 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. The computational processes of self-driving cars, for example, consume on the order of a thousand watts. One of the current top submissions of the Labeled Faces in the Wild (LFW) face verification task is a deep ANN by Paravision 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. A more extreme case is the 11-billion parameter version of Google's T5 model (Raffel et al., 2019), which is estimated to cost more than $1.3 million per training run (Sharir, Peleg, and Shoham, 2020). This difference in power consumption precludes computations in mobile low-power or small-scale devices, which now requires at least a connection to a cloud computing server.

The energy consumption of DL contrasts strongly with the 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 they 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 fully in parallel (A Pastur-Romay et al., 2017) using sparsely occurring postsynaptic potentials (or *spikes*) (Bear, Connors, and Paradiso, 2020), and where no explicit data processing instructions exist. 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. There are also no matrix multiplications in the brain; 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).

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 named backpropagation through time (BPTT). The human brain, in contrast, is unlikely to use

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

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. The concept dates back to the 1980s (Hopfield, 1982); nevertheless, they are sometimes considered as the third generation of ANNs (Maass, 1997). SNNs use neurons that do not relay continuous activation values at every propagation cycle, but spike once when they reach a threshold value. SNNs are competitive to ANNs in terms of accuracy and computational power, as well in they 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 BPTT can be applied to recurrent SNNs (RSNNs) (Bellec, Salaj, et al., 2018; Huh and Sejnowski, 2017). Some RSNN training methods rely on control theory to train a chaotic reservoir of neurons (Gilra and Gerstner, 2017; Thalmeier et al., 2016). Information locality can be preserved in some of there methods (Alemi et al., 2018), i. e., a neuron or synapse can only access information of itself, or the communication of synapses or neurons (resp.) with which it is directly connected. The FORCE training method (Nicola and Clopath, 2017), in contrast, can also train RSNNs but it nonlocal. This body of research led to increased understanding of training SNNs and how to obtain better learning performances. For instance, both single- and multi-layer SNNs have shown good performance in visual processing (Escobar et al., 2009; Kheradpisheh et al., 2018; Liu and Yue, 2017) and speech recognition (Dong, X. Huang, and B. Xu, 2018; Tavanaei and Maida, 2017). 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 discipline 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 ab-

stracted in a digital system, but physically embedded in an 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 parse, event-based communication and physically colocalized memory and computation (E. O. Neftci, 2018; Sterling and Laughlin, 2015). Although colocalized memory and computation has also been implemented in von Neumann 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). Neuromorphic systems are also more tolerant to device variation (Yu et al., 2013).

Because of their massive parallelism, high energy efficiency, good error tolerance, 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 (Zhang et al., 2020).

BIOLOGICAL LEARNING To run an SNN on neuromorphic hardware, a *local* and *online* 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 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 future information, nor past information if it was not explicitly retained.

The brain also adheres to these two constraints. Most learning algorithms for SNNs are rooted in a form of Hebbian learning, which is a major factor in biological learning and memory consolidation. 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

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

positive feedback loop, because "wiring cells together", or increasing the synaptic strength, will in turn increase the likelihood they 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: if neuron B spikes right after neuron A, then the synapse is strengthened. If B spikes right before A, it is weakened. However, this too leads to runaway activity through the same positive feedback loop. 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; 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), outperforming its classical two-factor counterpart (Porr and Wörgötter, 2007) because learning signals are crucial for maintaining long-term memory (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 (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). Acetylcholine is another example of a neuromodulator that gates synaptic plasticity in the cortex and enables state-dependent learning, in which memories are recalled better if the sensory context is similar to that during the memory encoding (Shulz et al., 2000).

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). An eligibility trace of a synapse is elicited when a neuron spikes, and fades away slowly enough that a delayed learning signal can still modulate the synaptic plasticity through R-STDP (Cassenaer and Laurent, 2012; Gerstner, Lehmann, et al., 2018; Yagishita et al., 2014). Synaptic plasticity was demonstrated using synaptic plasticity 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, but Zenke and Ganguli, 2018 notes that these methods are 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). R-STDP in SNNs has been used to solve a number of tasks, including training a stable lane keeping controller (Bing, Meschede, Chen, et al., 2020), but can suffer from catastrophic forgetting and lack of policy evaluation in mapless navogation (Bing, Meschede, K. Huang, et al., 2018).

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 2.2.3). It uses local learning signals and eligibility traces for any type of neuron. 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). 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 on classifying phones (i. e., speech sounds), for which it performs competitively with LSTMs, a popular RNN architecture that uses BPTT.

So far, only the LIF and adaptive LIF (ALIF) neuron models have been used in e-prop. 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. Multi-layer CNNs 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. Deeper layers exhibit slower time dynamics than shallow layers, suggesting that they ignore small variations in the input signal and integrate larger temporal patterns. It is unclear if these findings extrapolate to RSNNs.

THIS RESEARCH In this report, the performance of e-prop in Bellec, Scherr, Subramoney, et al. (2020) is reproduced. Whereas in Bellec, Scherr, Subramoney, et al. (2020) e-prop was implemented using automatic differentiation, in this report it is done explicitly, using the e-prop learning equations. Furthermore, the two new STDP neuron models of Traub et al. (2020) are experimentally verified in the TIMIT phone classification task. One of these models, the LIF-STDP, will be modified to an adaptive version named ALIF-STDP. Finally, the effect of stacking multiple RSNN layers in an e-prop with each of the three neuron models is examined.

Chapter 2 describes the basic version of the e-prop framework. Chapter 3 describes the method used to implement the TIMIT learning task and modify the e-prop algorithm to a multi-layer framework with different neuron models. The results are presented in Chapter 4 and discussed in Chapter 5. Finally, Chapter 6 concludes this report.

2

RELATED WORK

2.1 THREE-FACTOR HEBBIAN LEARNING

-3F Hebbian learning - STDP is observed - What (math, plot?), how biologically - But how can these synaptic changes lead to long-term behaviorial changes? - STDP requires modulatory signals (bailey2000heterosynaptic)

2.1.1 *Spike-timing dependent plasticity*

- Clopath rule - R-STDP - (and other variants if they're relevant)

2.1.2 *Learning signal*

- Biological plausilibity, (how does it happen in brain?) - Error-related negativity (see Bellec1)

2.1.2.1 *Broadcasting*

- Broadcasting methods - Broadcast alignment (see Bellec1) - In brain

2.1.3 *Eligibility traces*

- Why necessary? - Bioplausibility

2.2 ELIGIBILITY PROPAGATION

2.2.1 *Network model*

An eligibility propagation model $\mathcal{M}$ 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 $\mathbf{h}_j^t$ of a neuron j at a discrete time step t, where $\mathbf{z^{t-1}}$ is the observable state of all neurons at the previous time step, $\mathbf{x}^t$ is the model input vector at time t, and $\mathbf{W}_j$ is the weight vector of afferent synapses. The update of the observable state of a neuron j at time t is defined by

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

This formalization means that e-prop is a *local* training method, because a neuron's observable state depends only on its own hidden state, and the hidden state 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 variables in the previous time point.

2.2.2 *Neuron models*

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

connect to synaptic scaling

cite

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

where H is the Heaviside step function, and v_{th} is the threshold constant. Consequently, the observable state $z_j^t \in \{0, 1\}$ is binary, and denotes the spike activity of a neuron. These spikes are the only communication between neurons in the model. The LIF neuron model has a single state h_j^t that contains only an activity value v_j^t and 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, α is a constant decay factor. Whenever j spikes (i.e., $z_j^t = 1$), the activity of the neuron is reduced to a value near 0 by the term $-z_j^t v_{\text{th}}$. Furthermore, z_j^t is fixed to 0 for T^{refr} time steps to model neuronal refractoriness that is also present in biological neurons.

cite

caption and reference to fig

ALIF NEURON The adaptive LIF (ALIF) neuron introduces a threshold adaptation variable a_j^t to the hidden state of the neuron, such that $\mathbf{h}_j^t \overset{\text{def}}{=} \left[v_j^t, a_j^t\right]$. In an ALIF neuron, the spiking threshold increases after a spike, and otherwise decreases back to a baseline threshold v_{th}. 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 ρ is an adaptation decay constant.

In this paper, the LIF neuron is generalized as an ALIF neuron for which $\beta = 0$, effectively cancelling the effect of the threshold adaptation value a_j^t on the observable state z_j^t in Equation **??**. Therefore, only the e-prop derivations for the ALIF neurons will be described in the following sections.

- Mention SFA - Mention GLIF (Bellec2)

- DEMO FIGURE

2.2.3 *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 subsection, the main equation of e-prop

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

is derived from the classical factorization of the loss gradients in an unfolded RNN:

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

where summation indicates that weights are shared.

We now decompose the first term 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 after the event horizon t':

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

Note that this equation is recursive. If we substitute Equation 2.9 into the classical factorization (Equation 2.8), we obtain a recursive expansion that has $\frac{dE}{d\mathbf{h}_j^{T+1}}$ as its terminating case:

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

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

We write the term in parentheses into a second term indexed by t:

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

We then exchange the summation indices to pull out the learning signal L_j^t 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* ϵ_{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:

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

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 elibibility vector and the hidden state have the same dimension: $\left\{\epsilon_{ji}^t, \mathbf{h}_j^t\right\} \subset \mathbb{R}^d$, where $d = 2$ for all neuron types described in this report.

The *eligibility trace* e_{ji}^t is a product of $L_j^t = \frac{dE}{dz_j^t}$ 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 **??** for details):

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

This is the main e-prop equation.

2.2.4 *Learning procedure*

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

2.2.4.1 *Eligibility trace*

Recall the hidden state update equations from Section 2.2.2:

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

and

$$a_j^{t+1} = \rho a_j^t + z_j^t \qquad \text{(2.6 revisited)}$$

and the update of the observable state

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

The ALIF neuron model has a two-dimensional hidden state

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

associated with a neuron j and a two-dimensional eligibility vector

$$\epsilon_{ji}^t \overset{\text{def}}{=} \begin{pmatrix} \epsilon_{ji,v}^t \\ \epsilon_{ji,a}^t \end{pmatrix} \tag{2.16}$$

associated with a synapse from neuron i to neuron j.

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

$$\frac{\mathbf{h}_j^{t+1}}{\mathbf{h}_j^t} = \begin{pmatrix} [1.5]\frac{\partial v_j^{t+1}}{\partial v_j^t} & \frac{\partial v_j^{t+1}}{\partial a_j^t} \\ \frac{\partial a_j^{t+1}}{\partial v_j^t} & \frac{\partial a_j^{t+1}}{\partial a_j^t} \end{pmatrix} \tag{2.17}$$

The presence of z_j^t, and its relation with 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 derivates, 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.

Now, the partial derivatives in the hidden state derivative can be computed:

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

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

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

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

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

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

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

$$= \begin{pmatrix} [1.5]\alpha \cdot \epsilon_{ji,v}^t + z_i^{t-1} \\ \psi_j^t \epsilon_{ji,v}^t + \left(\rho - \psi_j^t \beta\right) \epsilon_{ji,a}^t \end{pmatrix} \tag{2.25}$$

This eligibility vector 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 sent 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.

Multiplying the eligibility vector with the partial derivative of the observable state with respect to the hidden state results in the eligibility trace:

$$e_{ji}^t = \begin{pmatrix} \epsilon_{ji,v}^t \\ \epsilon_{ji,a}^t \end{pmatrix} \cdot \begin{pmatrix} \frac{\partial z_j^t}{\partial v_j^t} \\ \frac{\partial z_j^t}{\partial a_j^t} \end{pmatrix} \tag{2.26}$$

$$= \begin{pmatrix} \epsilon_{ji,v}^t \\ \epsilon_{ji,a}^t \end{pmatrix} \cdot \begin{pmatrix} \psi_j^t \\ -\beta \psi_j^t \end{pmatrix} \tag{2.27}$$

$$= \psi_j^t \left(\epsilon_{ji,v}^t - \beta \epsilon_{ji,a}^t\right) \tag{2.28}$$

2.2.4.2 *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:

$$\Delta W = -\eta \frac{\widehat{dE}}{dW_{ji}} \stackrel{\text{def}}{=} -\eta \sum_t \frac{\partial E}{\partial z_j^t} e_{ji}^t. \tag{2.29}$$

Note that for clarity, this section describes e-prop using the stochastic gradient descent. In the actual implementation, 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

$$y_k^t = \kappa y_k^{t-1} + \sum_j W_{kj}^{\text{out}} z_j^t + b_k, \tag{2.30}$$

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, smoothening 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:

$$\pi_k^t = \sigma_k\left(y_1^t, \ldots, y_K^t\right) = \frac{\exp\left(y_k^t\right)}{\sum_{k'} \exp\left(y_{k'}^t\right)}. \tag{2.31}$$

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

$$E = -\sum_{t,k} \pi_k^{*,t} \log \pi_k^t, \tag{2.32}$$

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 neurons k, we can derive

$$L_j^t = \frac{\partial E}{\partial z_j^t} = \sum_k B_{jk} \sum_{t' \geq t} \left(\pi_k^{t'} - \pi_k^{*,t}\right) \kappa^{t'-t}, \tag{2.33}$$

where B_{jk} is a feedback weight from neuron k back to neuron j. There are multiple strategies for choosing feedback weights. Bellec, Scherr, Subramoney, et al., 2020 stated that a constantly uniform weight matrix yields poor performance. However, when the feedback weight matrix is initialized from a zero-centered normal distribution, it can remain constant, $(W^{\text{out}})^\top$, or update according to $(\Delta W^{\text{out}})^\top$. These 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., $B_{jk} \overset{\text{def}}{=} W_{kj}^{\text{out}}$) unless explicitly stated otherwise.

Note that the term $\kappa^{t'-t}$ in Equation 2.33 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 $\bar{e}_{ji}^t$. Recall that the estimated loss gradient $\frac{dE}{dW_{ji}}$ is approximated by $\sum_t \frac{\partial E}{\partial z_j^t} e_{ji}^t$.

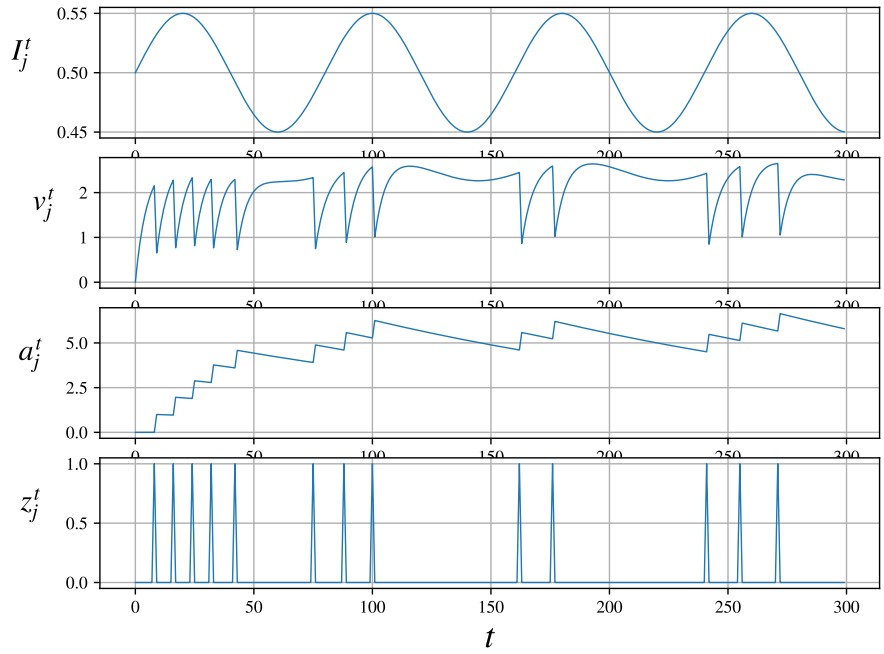

Figure 2.1: .

Therefore, after inserting Equation 2.33 in Equation 2.29, the weight update is computed by

$$\Delta W_{ji} = -\eta \sum_{t'} \frac{\partial E}{\partial z_j^{t'}} e_{ji}^{t'} \tag{2.34}$$

$$= -\eta \sum_{t'} \sum_{k} B_{jk} \sum_{t' \geq t} \left(\pi_k^{t'} - \pi_k^{*,t} \right) \kappa^{t'-t} e_{ji}^{t'} \tag{2.35}$$

$$= -\eta \sum_{k,t'} B_{jk} \sum_{t' \geq t} \left(\pi_k^{t'} - \pi_k^{*,t} \right) \kappa^{t'-t} e_{ji}^{t'} \tag{2.36}$$

$$= -\eta \sum_{t} \underbrace{\sum_{k} B_{jk} \left(\pi_k^{t'} - \pi_k^{*,t} \right)}_{=L_j^t} \underbrace{\sum_{t' \leq t} \kappa^{t'-t} e_{ji}^{t'}}_{\stackrel{\text{def}}{=} \bar{e}_{ji}^t}, \tag{2.37}$$

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 2.37 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:

$$\Delta W_{kj}^{\text{out}} = -\eta \sum_{t} \left(\pi_k^t - \pi_k^{*,t} \right) \sum_{t' \leq t} \kappa^{t'-t} z_j^t \tag{2.38}$$

and

$$\Delta b_k = -\eta \sum_{t} \left(\pi_k^t - \pi_k^{*,t} \right) \tag{2.39}$$

why?

2.3 SYNAPTIC SCALING

- Synaptic Scaling (in brain, if applicable)

2.4 NETWORK TOPOLOGY

- Network topology (e.g. multilayer) (but keep relevant) - Mainly focus on how brain does it. Cite often! Don't hypothesize on effects, just describe with sources. - Also denote a subsection on effects of topologies in related ANNs (preferably (R)SNNs).

3

METHOD

3.1 DATA PREPROCESSING

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 all speakers lived in 8 different geographical regions in the United States (as categorized in Labov, Ash, and Boberg, 2008) 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.

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.

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.

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, and to anneal the learning rate (see Section **??**). 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 portions.
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 phonemes 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 phoneme. 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.

3.1.3 *Engineering features*

In this subsection, we describe the preprocessing pipeline as 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). We align these MFCCs with the phones found in the TIMIT dataset. An example of a waveform signal is given in Figure ??.

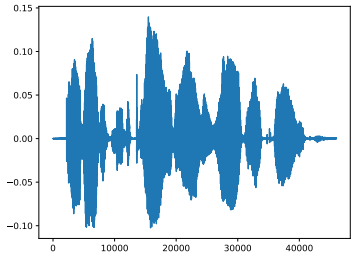

Figure 3.1: A raw waveform signal from the TIMIT dataset.

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, we apply a pre-emphasis filter $y(t)$ on the waveform signal $x(t)$ defined in Equation 3.1.

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

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

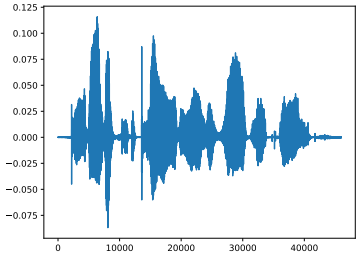

Figure 3.2: A signal after the pre-emphasis filter of Equation 3.1 was applied to it.

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 samples. Furthermore, the samples are not very informative, because they represent the sound wave of the uttered 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. We assume the frequency components to be stationary over short intervals—in contrast to the full sentence, which carries its meaning because it is non-stationary. Therefore, we transform the waveform signals into series of frequency coefficients of short-term power spectra. To obtain multiple short-term power spectra over the duration of the waveform, we slice it up into brief overlapping frames.

Every 160 samples (equivalent to 10 ms) of a pre-emphasized signal we take an interval frame of 400 samples (equivalent to 25 ms). This 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, we obtain signal frames $x_i(n)$, where n ranges over 1–400, and i ranges over the number of frames in the waveform.

Then, we apply a Hamming window with the form

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

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 ??. This

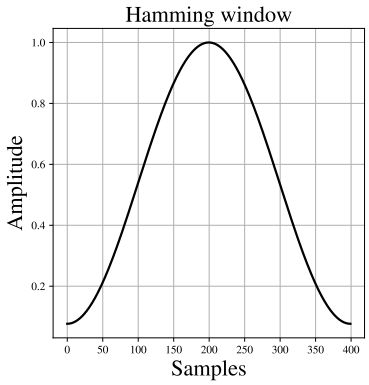

Figure 3.3: The magnitudes of the DFT of a frame.

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 (Smith, accessed <date>)..

plot for illustration

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

don't bother with eqn, just call mathbb(F)

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

where $K = 512$. This yields the magnitudes of the DCT of the frames (an example is illustrated in Figure ??).

We obtain the power spectrum using the equation

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

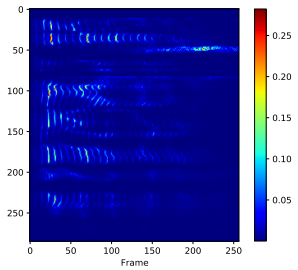

Figure 3.4: The magnitudes of the DFT of a frame.

an example of which is shown in Figure ??.

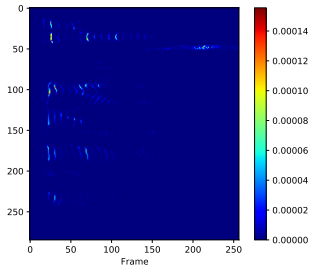

Figure 3.5: A power spectrum of a frame.

MEL FILTERBANK We then transform the short-term power spectra 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 better distinguish lower frequencies better 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 we choose lower and upper band edges of 0 Hz and $f_s/2 = 8\,\text{kHz}$, respectively, 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. We obtain a lower band edge of 0 Mels and an upper band edge of approximately 2835 Mels.

We begin obtaining the 40 filterbanks by spacing 42 points $\mathbf{m}$ linearly between these bounds (inclusive). Hence, we obtain 42 points spaced exclusively between the bounds.

Then, we convert each point m back to Hz using

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

We round each resulting Mel-spaced frequency f to their nearest Fourier transform bin b using

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

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

not sure

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

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

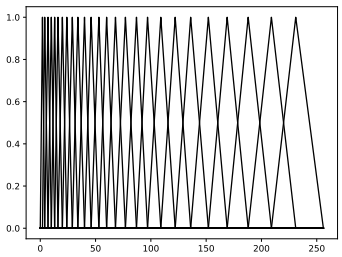

Figure 3.6: The Mel-spaced filterbanks.

We obtain a spectrogram S of the frame (see e. g.Figure ??) after applying the filterbank to the short-term power spectrum.

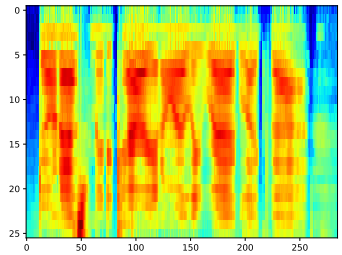

Figure 3.7: An example of a spectrogram.

MEL-FREQUENCY CEPSTRAL COEFFICIENTS We observe that the coefficients in the spectrograms are strongly correlated, which would negatively impact the learning performance of the model .

why?

Therefore, we apply the DCT again to decorrelate the coefficients and obtain the power cepstrum C of the speech frame:

do we take absolute?

$$C(n) = \left| \sum_{n=0}^{N-1} S(n) \cdot e^{-\frac{i2\pi}{N} kn} \right|. \tag{3.9}$$

We discard the first coefficient in C, because it is the average power of the input signal and therefore carries little meaning. We also discard coefficients higher than 13, because they represent only fast changes in the spectrogram and increase the complexity of the input signal while adding increasingly less meaning to it. An example of the remaining MFCC components is shown in Figure ??.

source?

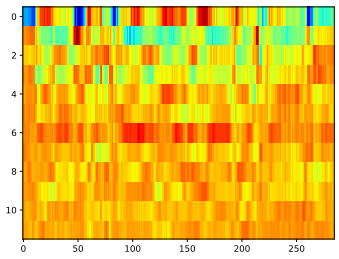

Figure 3.8: An example of Mel-frequency cepstral coefficients that are given as input to the system.

Then, we balance the final MFCCs by centering each frame around the value 0. Next, the trailing frames that are labeled as 'silent' are trimmed from the end of the input and target sequences. Finally, to reflectthe speed of the original waveform signal, the input sequences are stretched by a factor of 5, interpolating linearly between frames. The target sequences are also stretched by this factor, but proximally interpolated to retain its one-hot encoding. An example of the final MFCCs is given in ??.

better wording: re-approximate?

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 ?? illustrates the waveform data and its frame-wise aligned MFCCs and target output.

side-by-side with original text and phonemes, label as fig:source_mfcc_target

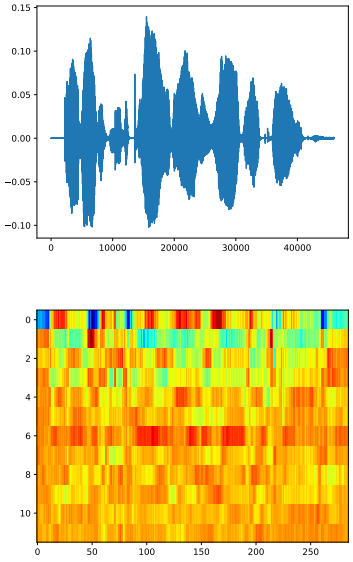

Figure 3.9: An alignment of a sample signal with its MFCCs and target phones.

3.2 ENHANCING E-PROP

preamble

3.2.1 *Multi-layer architecture*

- English, visual and formal descriptions.

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

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

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 $r \in [1 \,..\, R]$:

$$\mathbf{h}_{rj}^t = \begin{cases} M\left(\mathbf{h}_{rj}^{t-1}, \mathbf{z}_r^{t-1}, \mathbf{x}^t, \mathbf{W}_{rj}\right) & \text{if } r = 1 \\ M\left(\mathbf{h}_{rj}^{t-1}, \mathbf{z}_r^{t-1}, \mathbf{z}_{r-1}^t, \mathbf{W}_{rj}\right) & \text{otherwise,} \end{cases} \tag{3.10}$$

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

$$z_{rj}^t = f\left(\mathbf{h}_{rj}^t\right) \tag{3.11}$$

and the network output by

$$y_k^t = \kappa y_k^{t-1} + \sum_{j,r} W_{rkj}^{\text{out}} z_{rj}^t + b_k, \tag{3.12}$$

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 2.2.2). The only difference, apart from the layer indexing, is its activity update. For a multi-layer ALIF neuron, the activity value is given by

$$v_{rj}^{t+1} = \alpha v_{rj}^t + \sum_{i \neq j} W_{rji}^{\text{rec}} z_i^t + \sum_i W_{rji}^{\text{in}} I - z_{rj}^t v_{\text{th}}, \tag{3.13}$$

where

$$I = \begin{cases} x_i^{t+1} & \text{if } r = 1 \\ z_{r-1,i}^{t+1} & \text{otherwise.} \end{cases} \tag{3.14}$$

3.2.2 *Other neuron types*

- Argue in favor of different models (refer to brain, simplicity vs. plausibility trade-off). E.g.: built-in refractory
- Explain that ALIF displays no STDP, which is believed to be crucial for biological learning and forming associations and memory.

In this section, the multi-layer 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, as opposed to the (STDP-)ALIF model that requires an explicit timer variable in a neuron's hidden state.

The Izhikevich 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-LIF*

- Also mention and motivate v-fix and psi-fix.
- Mention Bellec's reset too.

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 uses a soft reset by including a term $-z_{rj}^t v_{\text{thr}}$ in its activation update equation (Equation 3.13).

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

$$v_{rj}^{t+1} = \alpha v_{rj}^t + \sum_{i \neq j} W_{rji}^{\text{rec}} z_i^t + \sum_i W_{rji}^{\text{in}} I - z_{rj}^t \alpha v_{rj}^t - z_{rj}^{t-\delta t_{\text{ref}}} \alpha v_{rj}^t, \quad (3.15)$$

where, again, I is the network input if $r = 1$, otherwise it is the observable state of the preceding layer (see Equation 3.14).

The hidden state derivative changes accordingly:

$$\frac{\partial v_{rj}^{t+1}}{\partial v_{rj}^t} = \alpha - z_{rj}^t \alpha - \alpha z_{rj}^{t-\delta t_{\text{ref}}} \quad (3.16)$$

$$= \alpha \left(1 - z_{rj}^t - z_{rj}^{t-\delta t_{\text{ref}}}\right). \quad (3.17)$$

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:

$$\epsilon_{rji}^{t+1} = \frac{\partial \mathbf{h}_{rj}^{t+1}}{\partial \mathbf{h}_{rj}^t} \cdot \epsilon_{rji}^t + \frac{\partial \mathbf{h}_{rj}^{t+1}}{\partial W_{rji}} \quad (3.18)$$

$$= \begin{pmatrix} [1.5]\alpha \left(1 - z_{rj}^t - z_{rj}^{t-\delta t_{\text{ref}}}\right) \epsilon_{rji,v}^t + z_{ri}^{t-1} \\ \psi_{rj}^t \epsilon_{rji,v}^t + \left(\rho - \psi_{rj}^t \beta\right) \epsilon_{rji,a}^t \end{pmatrix} \quad (3.19)$$

Note that the observable state of the afferent neuron z_{ri}^{t-1} in Equation 3.19 changes to $z_{r-1,i}^t$ if ϵ_{rji}^{t+1} corresponds to a weight between layer $r-1$ and r and $r > 1$. If the weight is instead between the network input and the first layer, then it changes to x_i^t. Note also that this corrects an error 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 3.10.

is this true? Critically review how Traub handles pseudo-derivatives in z, and check if dzdh remains the

The eligibility trace remains

$$e^t_{rji} = \frac{\partial z^t_{rj}}{\partial \mathbf{h}^t_{rj}} \cdot \epsilon^t_{rji} \tag{3.20}$$

$$= \psi^t_{rj} \left(\epsilon^t_{rji,v} - \beta \epsilon^t_{rji,a} \right). \tag{3.21}$$

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:

$$\psi^t_{rj} = \begin{cases} -\gamma & \text{if } t - t_{z_{rj}} < \delta t_{\text{ref}} \\ \gamma \max\left(0, 1 - \left|\frac{v^t_{rj} - v_{\text{th}}}{v_{\text{th}}}\right|\right) & \text{otherwise} \end{cases} \tag{3.22}$$

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

caption and reference to fig

3.2.2.2 *Izhikevich neuron*

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

$$v' = 0.04v^2 + 5v + 140 - u + I \tag{3.23}$$

$$a' = 0.004v - 0.02a, \tag{3.24}$$

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:

$$\tilde{v}^t_{rj} = v^t_{rj} - \left(v^t_{rj} + 65\right) z^t_{rj} \tag{3.25}$$

$$\tilde{a}^t_{rj} = a^t_{rj} + 2z^t_{rj}, \tag{3.26}$$

such that when a spike event occurs (i. e., $z^t_{rj} = 1$), 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:

$$v^{t+1}_{rj} = \tilde{v}^t_{rj} + 0.04 \left(\tilde{v}^t_{rj}\right)^2 + 5\tilde{v}^t_{rj} + 140 - \tilde{u}^t_{rj} + I^t_{rj} \tag{3.27}$$

$$u^{t+1}_{rj} = \tilde{u}^t_{rj} + 0.004\tilde{v}^t_{rj} - 0.02\tilde{u}^t_{rj}. \tag{3.28}$$

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

$$\frac{\partial v^{t+1}_{rj}}{\partial v^t_{rj}} = \left(1 - z^t_{rj}\right)\left(6 + 0.08 v^t_{rj}\right) \tag{3.29}$$

$$\frac{\partial a^{t+1}_{rj}}{\partial v^t_{rj}} = -1 \tag{3.30}$$

$$\frac{\partial v^{t+1}_{rj}}{\partial a^t_{rj}} = 0.004\left(1 - z^t_{rj}\right) \tag{3.31}$$

$$\frac{\partial a^{t+1}_{rj}}{\partial a^t_{rj}} = 0.98. \tag{3.32}$$

The eligibility vector is then computed as Using these values, we compute the new eligibility trace:

$$\epsilon_{rji}^{t+1} = \frac{\partial \mathbf{h}_{rj}^{t+1}}{\partial \mathbf{h}_{rj}^{t}} \cdot \epsilon_{rji}^{t} + \frac{\partial \mathbf{h}_{rj}^{t+1}}{\partial W_{rji}} \tag{3.33}$$

$$= \begin{pmatrix} [1.5]\left(1 - z_{rj}^{t}\right)\left(6 + 0.08 v_{rj}^{t}\right)\epsilon_{rji,v}^{t} - \epsilon_{rji,a}^{t} + z_{ri}^{t-1} \\ 0.004\left(1 - z_{rj}^{t}\right)\epsilon_{rji,v}^{t} + 0.98\epsilon_{rji,a}^{t} \end{pmatrix} \tag{3.34}$$

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

$$\psi_{rj}^{t} = \gamma \ \exp\left(\frac{\min\left(v_{rj}^{t}, 30\right) - 30}{30}\right), \tag{3.35}$$

such that

$$\begin{pmatrix} [1.5]\frac{\partial z_{rj}^{t}}{\partial v_{rj}^{t}} \\ \frac{\partial z_{rj}^{t}}{\partial u_{rj}^{t}} \end{pmatrix} = \begin{pmatrix} [1.5]\psi_{rj}^{t} \\ 0 \end{pmatrix} \tag{3.36}$$

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

$$e_{rji}^{t} = \begin{pmatrix} \psi_{rj}^{t} & 0 \end{pmatrix} \begin{pmatrix} [1.5]\epsilon_{rji,v}^{t} \\ \epsilon_{rji,a}^{t} \end{pmatrix} \tag{3.37}$$

$$= \psi_{rj}^{t}\epsilon_{rji,v}^{t}. \tag{3.38}$$

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*

- What, why?
- Bioplausible? Quote "Activity-dependent regulation of vesicular glutamate and GABA transporters: A means to scale quantal size"

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

$$E_{\text{reg}} = \frac{1}{2}\sum_{j}\left(f^{\text{target}} - f_{rj}^{\text{av},t}\right)^{2}, \tag{3.39}$$

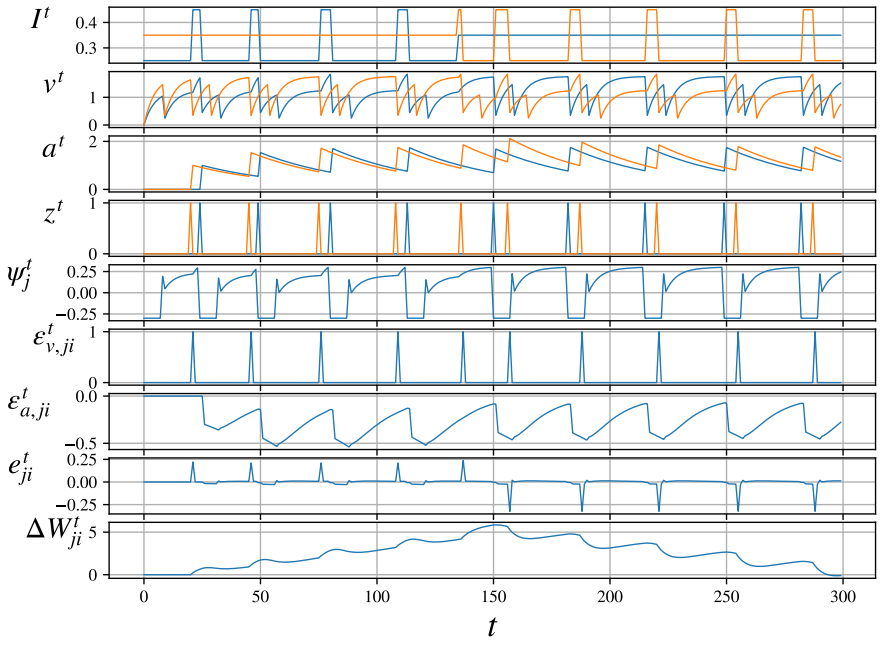

Figure 3.10: .

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

$$f_{rj}^{\text{av},t} = \frac{1}{t} z_{rj}^{\text{total},t} \tag{3.40}$$

is the running average 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 every new training sample. By implementing this sum as a hidden variable, e-prop remains an online training algorithm when firing rate regularization is implemented.

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 (Equation 2.29):

$$\frac{\partial E_{\text{reg}}}{\partial z_{rj}^t} = \left(f^{\text{target}} - f_{rj}^{\text{av},t} \right). \tag{3.41}$$

Note that this regularization loss differs from the firing rate regularization 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 2.29 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 nullifies the regularization gradient, and thereby results in a better empirical learning performance;
2. It is more efficient in emulations on von Neumann machines, because the element-wise multiplication of $\frac{\partial E_{\text{reg}}}{\partial z^t_{rj}}$ 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

$$\Delta_{\text{reg}} W_{rji} = -\eta \; c_{\text{reg}} \sum_t \left(f^{\text{target}} - f^{\text{av},t}_{rj} \right). \tag{3.42}$$

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:

$$\Delta W_{rji} = -\eta \sum_t \left(c_{\text{reg}} \left(f^{\text{target}} - f^{\text{av},t}_{rj} \right) + L^t_{rj} \cdot \bar{e}^t_{rji} \right). \tag{3.43}$$

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:

$$\Delta_{\text{L2}} W_{rji} = -\eta \; c_{\text{L2}} \cdot W_{rji}, \tag{3.44}$$

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

$$\Delta W_{rji} = -\eta \left(c_{\text{L2}} \cdot W_{rji} + \sum_t \left(c_{\text{reg}} \left(f^{\text{target}} - f^{\text{av},t}_{rj} \right) + L^t_{rj} \cdot \bar{e}^t_{rji} \right) \right). \tag{3.45}$$

caption and reference to fig

The statistical motivation for this extra 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 limite (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:

$$\Delta W_{rji} = -\eta \sum_t L^t_{rj} \cdot \bar{e}^t_{rji}. \tag{3.46}$$

However, the results described in Chapter 4 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}). The Adam weight update in the context of multi-layer e-prop is given by:

$$M^{(i+1)}_{rji} = \beta_1 M^{(i)}_{rji} + (1-\beta_1)\, G^{(i)}_{rji} \tag{3.47}$$

$$V^{(i+1)}_{rji} = \beta_2 V^{(i)}_{rji} + (1-\beta_2) \left(G^{(i)}_{rji}\right)^2 \tag{3.48}$$

$$\widehat{M}_{rji} = \frac{M^{(i+1)}_{rji}}{1-\beta_1^{i+1}} \tag{3.49}$$

$$\widehat{V}_{rji} = \frac{V^{(i+1)}_{rji}}{1-\beta_2^{i+1}} \tag{3.50}$$

$$\Delta W^{(i+1)}_{rji} = -\eta \frac{\widehat{M}_{rji}}{\sqrt{\widehat{V}_{rji}} + 10^{-5}}, \tag{3.51}$$

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

3.5 HYPERPARAMETER OPTIMIZATION

- What, why, how?

4 RESULTS

4.1 NEURON MODELS

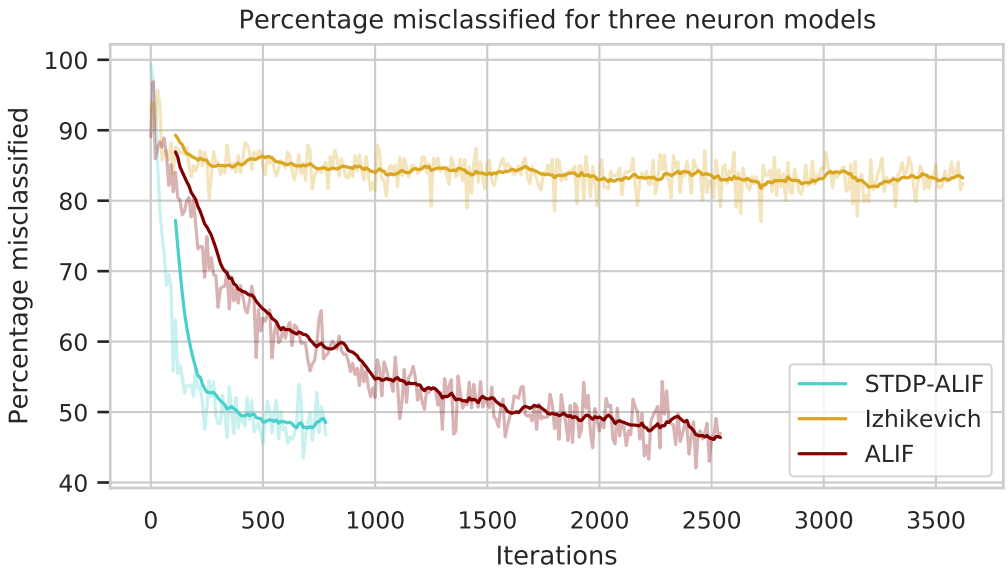

Figure 4.1

ACCURACY

CROSS-ENTROPY

FIRING RATE

4.2 NETWORK DEPTH

1 LAYER

2 LAYERS

3 LAYERS

- RESULTS PLACEHOLDERS

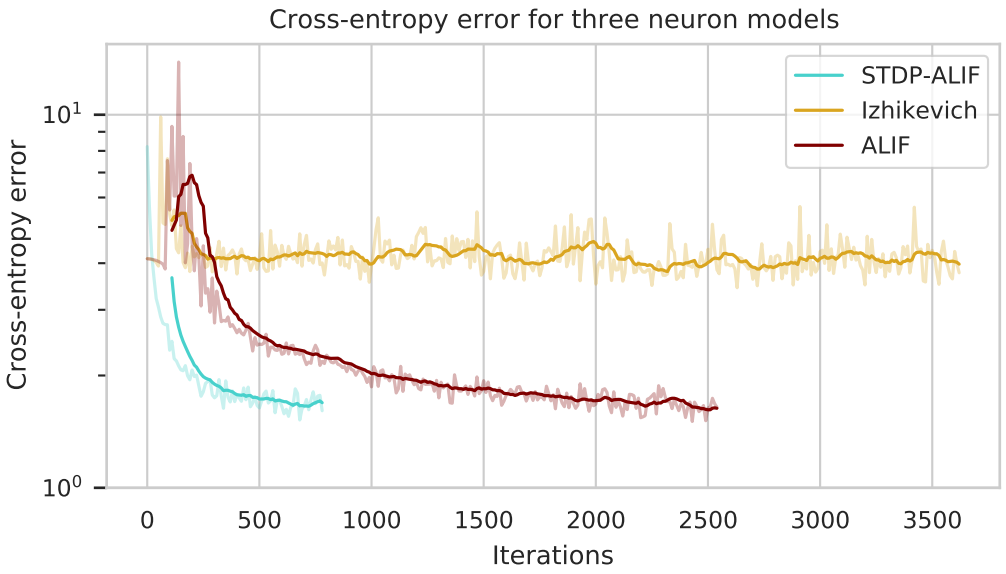

Figure 4.2

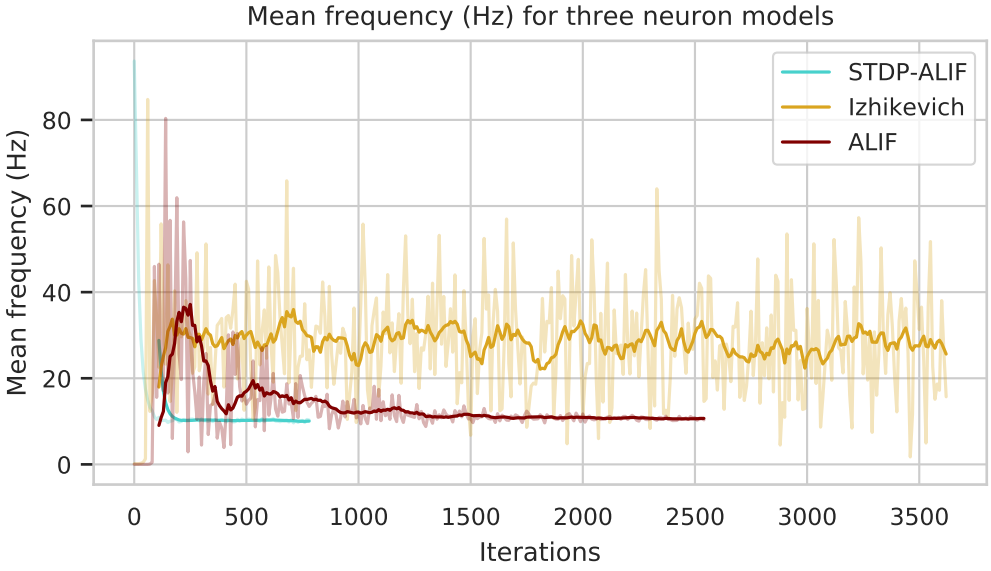

Figure 4.3

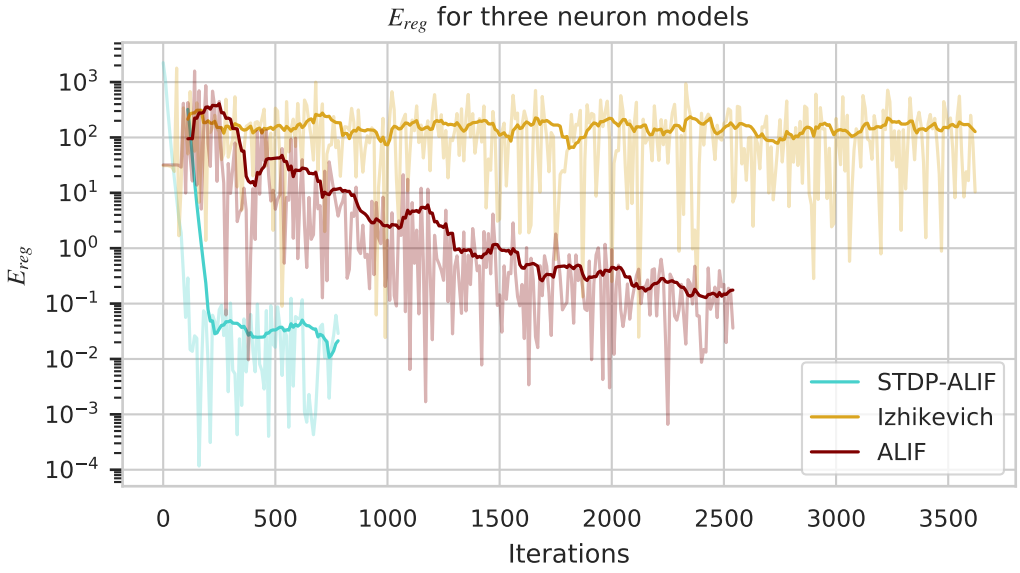
E_{reg} for three neuron models
E_{reg}
10^3
10^2
10^1
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
500
1000
1500
2000
2500
3000
3500
Iterations
STDP-ALIF
Izhikevich
ALIF

Figure 4.4

5

DISCUSSION

- DISCUSSION PLACEHOLDER
- STDP is not important for good performance in e-prop. - A multi-layer architecture slows has a theoretical advantage (src?) but raises sparsity issues

Future research: - Beta, rho, alpha, synaptic delay not constant but spread

6 CONCLUSION

- CONCLUSION PLACEHOLDERS

APPENDIX A

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.

BIBLIOGRAPHY

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

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

Alemi, Alireza et al. (2018). "Learning nonlinear dynamics in efficient, balanced spiking networks using local plasticity rules." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1.

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.

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: *bioRxiv*, p. 738385.

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

Bing, Zhenshan, Claus Meschede, Guang Chen, et al. (2020). "Indirect and direct training of spiking neural networks for end-to-end control of a lane-keeping vehicle." In: *Neural Networks* 121, pp. 21–36.

Bing, Zhenshan, Claus Meschede, Kai Huang, et al. (2018). "End to end learning of spiking neural network based on r-stdp for a lane keeping vehicle." In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 4725–4732.

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.

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.

Crevier, Daniel (1993). *AI: the tumultuous history of the search for artificial intelligence*. Basic Books, Inc., pp. 100–144.

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?*

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

Elbez, Hammouda et al. (2020). "Progressive Compression and Weight Reinforcement for Spiking Neural Networks." In:

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

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.

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.

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.

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.

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.

Gilra, Aditya and Wulfram Gerstner (2017). "Predicting non-linear dynamics by stable local learning in a recurrent spiking neural network." In: *Elife* 6, e28295.

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.

Haugeland, John (1985). *Artificial Intelligence: The Very Idea, Cambridge, MA: Bradford*.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

Minsky, Marvin and Seymour Papert (1969). "An introduction to computational geometry." In: *Cambridge tiass., HIT*.

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 (2014). *Neuromorphic computing gets ready for the (really) big time*.

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.

Nicola, Wilten and Claudia Clopath (2017). "Supervised learning in spiking neural networks with FORCE training." In: *Nature communications* 8.1, pp. 1–15.

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.

Raffel, Colin et al. (2019). "Exploring the limits of transfer learning with a unified text-to-text transformer." In: *arXiv preprint arXiv:1910.10683*.

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.

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.

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.

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*.

Sharir, Or, Barak Peleg, and Yoav Shoham (2020). "The Cost of Training NLP Models: A Concise Overview." In: *arXiv preprint arXiv:2004.08900*.

Shulz, Daniel E et al. (2000). "A neuronal analogue of state-dependent learning." In: *Nature* 403.6769, pp. 549–553.

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

Smith, Julius O. (accessed <date>). *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.

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 multi-layer spiking neural network extracts discriminative features from speech signals." In: *International conference on neural information processing*. Springer, pp. 899–908.

Thalmeier, Dominik et al. (2016). "Learning universal computations with spikes." In: *PLoS computational biology* 12.6, e1004895.

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 M (2009). "Computing machinery and intelligence." In: *Parsing the turing test*. Springer, pp. 23–65.

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.

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.

Yu, Shimeng et al. (2013). "A low energy oxide-based electronic synaptic device for neuromorphic visual systems with tolerance to device variation." In: *Advanced Materials* 25.12, pp. 1774–1779.

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.