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

A primary goal of artificial intelligence is to develop systems that exhibit intelligent behavior. During the 1980s, with the popularization of backpropagation (Rumelhart, G. E. Hinton, and Williams, 1986) and trainable Hopfield networks (Hopfield, 1982), the focus of the field shifted from expert systems and symbolic reasoning to *connectionist* approaches, such as artificial neural networks (ANNs). ANNs are networks of small computational units that can be trained to perform specific pattern recognition tasks. These networks are based loosely on the human brain.

As computing power and data storage capabilities increased exponentionally, and the rise of the internet provided abundant training data, ANNs have become a dominant field in artificial intelligence in the context of deep learning (DL). This has particularly been the case during the the 2010s, when GPUs were increasingly used to train deep neural networks. During the same period, convolutional neural networks (CNNs) and recurrent neural networks (RNNs) approached or exceeded human level performance in some areas (Schmidhuber, 2015). CNNs were also inspired by neuroscience; the connectivity pattern between units in a CNN resembles the organization of the primate visual cortex (Hubel and Wiesel, 1968).

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. IFLYTEK-CV, which is one of the best-performing systems in the LFW challenge for facial recognition, was trained using a dataset of 3.8 million face images of 85 thousant individuals.ResNethas been trained for 3 weeks on a 8-GPU server consuming about 1 GWh.For a more extreme example, training the 11-billion parameter version of Google's T5 model (Raffel et al., 2019) is estimated to cost more than $1.3 million per run(Sharir, Peleg, and Shoham, 2020). This contrasts strongly with the energy consumption of the human brain, which is made up of around 86 billion neurons (Azevedo et al., 2009) and 100–500 trillion synapses (Drachman, 2005), consumes approximately 20 W (Drubach, 2000; Sokoloff, 1960), and does not require as much data to learn patterns. This difference in power consumption is crucial for computations in mobile low-power or small-scale devices.

from vigneron2020, but cite original

cite

specify

from vigneron2020, but cite original

One reason why the human brain is more energy-efficient is that its computational function is realized in an analog and massively parallel physical substrate (A Pastur-Romay et al., 2017), in which neurons communicate through sparsely occurring spikes (Bear, Connors, and Paradiso, 2020). Connections in DL models, on the other hand, are represented by multiplications between the often large matrices of the

floating-point weight values of these connections and the activation values of the efferent units (LeCun, Bengio, and G. Hinton, 2015). Backpropagation, which has become the de-facto standard for training DL models, is a biologically implausible learning algorithm that trains models by propagating the error back into the network, further raising computational costs.

Spiking neural networks (SNNs) are another step towards biological plausibility of connectionist models. SNNs use neurons that do not relay continous activation values at every propagation cycle, but spike once when they reach a threshold value. The concept of SNNs dates back to the 1980s (Hopfield, 1982), but since spike-based activation is differentiable, gradient descent is not as effective as in ANNs to minimize the loss. Consequentially, the lack of suitable training methods has limited the popularity of SNNs.

Neuromorphic computing is an emerging technology that, like the human brain, peforms computation in a physical subtrate. This technology has the potential to offer more energy-efficient computation than the von Neumann architecture that is standard in training DL models. Due to the centralized nature of von Neumann computers, simulated SNNs do not enjoy the energy efficiency as networks of biological neurons. In theory, however, the massively parallel and decentralized neuromorphic computers can efficiently run SNNs. This requires a learning algorithm that is both spatially and temporally local (i. e., neurons and synapses can only change their state based on information that is available at the same timestep and immediately adjacent to that neuron or synapse).

This report examines functional modifications to *eligibility propagation* (e-prop), which is a spatially and temporally local learning algorithm for SNNs that suggests a promising approach to train SNNs in a neuromorphic architecture (Bellec et al., 2020).

- Brief historical overview of 3F-Hebbian (use my own literature trace) - Explain 3F-Hebbian (mention bioplausibility: online & local)

- E-prop approximates BPTT using RSNNs by using eligibility traces and learning signals. Also mathematical link (only intuition!)

- This paper examines the effects of enhancements that may improve the performance of e-prop. Some of these are used succesfully in DL. (Argue scientifically why these might improve performance) - Multilayer. Mention how MLPs were breakthrough on perceptrons. - Other neuron types - Regularization that's also observed in brain (e.g. synaptic scaling)

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 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

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

cite

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

- DEMO FIGURE

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:

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

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

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):

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

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:

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

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:

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

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

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

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:

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

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 3.6).

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 et al., 2020 stated that a constantly uniform weight matrix yields poor performance. why? 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 et al., 2020 as *random*, *symmetric*, and *adaptive* e-prop, respectively. In this paper, symmetric e-prop is used (i.e., $B_{jk} \overset{\text{def}}{=} W^{\text{out}}_{kj}$) unless explicitly stated otherwise.

Note that the term $\kappa^{t'-t}$ in Equation 2.33 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 $\bar{e}^t_{ji}$. Recall that the estimated loss gradient $\frac{dE}{dW_{ji}}$ is approximated by $\sum_t \frac{\partial E}{\partial z^t_j} e^t_{ji}$. Therefore, after inserting Equation 2.33 in Equation 2.29, the weight update is computed by

$$\Delta W_{ji} = -\eta \sum_{t'} \frac{\partial E}{\partial z^{t'}_j} e^{t'}_{ji} \tag{2.34}$$

$$= -\eta \sum_{t'} \sum_k B_{jk} \sum_{t' \geq t} \left(\pi^{t'}_k - \pi^{*,t}_k\right) \kappa^{t'-t} e^{t'}_{ji} \tag{2.35}$$

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

$$= -\eta \sum_t \underbrace{\sum_k B_{jk} \left(\pi^{t'}_k - \pi^{*,t}_k\right)}_{=L^t_j} \underbrace{\sum_{t' \leq t} \kappa^{t'-t} e^{t'}_{ji}}_{\overset{\text{def}}{=} \bar{e}^t_{ji}}, \tag{2.37}$$

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:

$$\Delta W^{\text{out}}_{kj} = -\eta \sum_t \left(\pi^t_k - \pi^{*,t}_k\right) \sum_{t' \leq t} \kappa^{t'-t} z^t_j \tag{2.38}$$

and

$$\Delta b_k = -\eta \sum_t \left(\pi^t_k - \pi^{*,t}_k\right) \tag{2.39}$$

- Full demo plot including eligibility vector, trace, and simulated learning signal

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 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 ??.

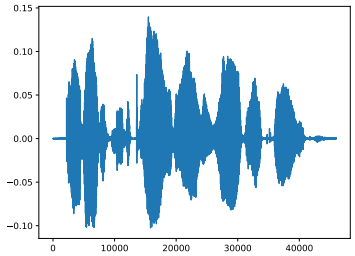

Figure 3.1: A raw waveform signal from the TIMIT dataset.

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 ??.

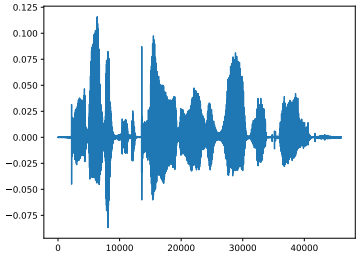

Figure 3.2: A signal after the pre-emphasis filter of Equation 3.1 was applied to it.

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

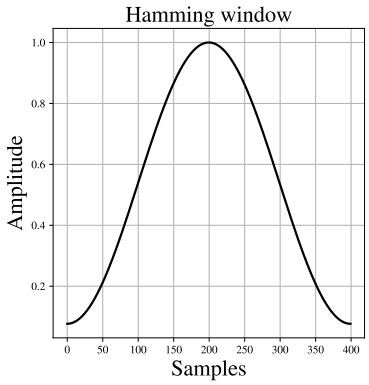

Figure 3.3: The magnitudes of the DFT of a frame.

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

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

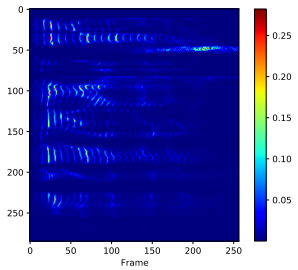

Figure 3.4: The magnitudes of the DFT of a frame.

an example of which is shown in Figure ??.

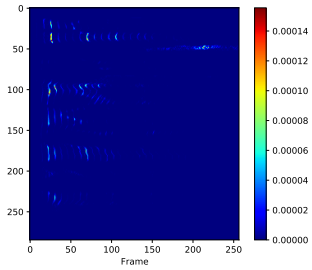

Figure 3.5: A power spectrum of a frame.

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 ??.

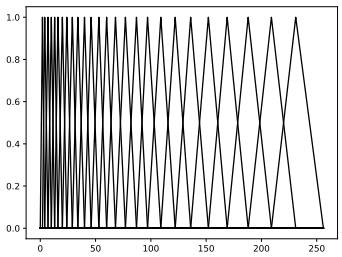

Figure 3.6: The Mel-spaced filterbanks.

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

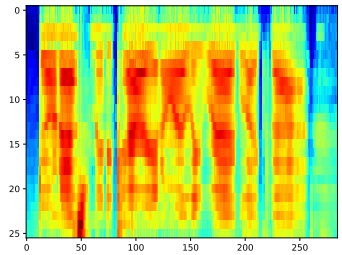

Figure 3.7: An example of a spectrogram.

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?

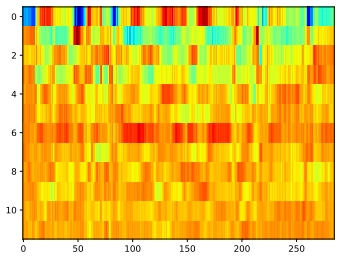

Figure 3.8: An example of Mel-frequency cepstral coefficients that are given as input to the system.

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

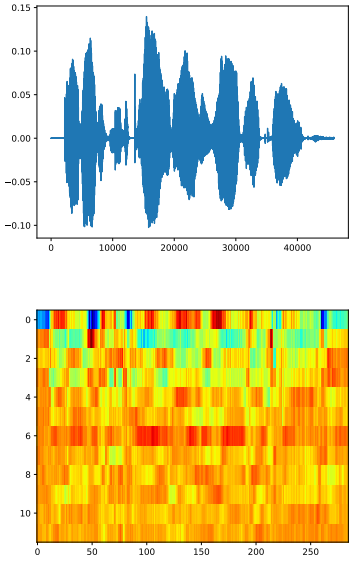

Figure 3.9: An alignment of a sample signal with its MFCCs and target phones.

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*

- Shoutout to Traub - Argue in favor of different models (refer to brain, simplicity vs. plausibility trade-off). E.g.: built-in refractory

3.2.2.1 *STDP-LIF*

- STDP-LIF (intuition, maths, graphs) - Also mention and motivate v-fix and psi-fix. - Mention Bellec's reset too.

3.2.2.2 *Izhikevich neuron*

- STDP-LIF (intuition, maths, graphs)

3.3 REGULARIZATION

preamble: explain why this is used (bioplausibility (natural constraints) and generalizability)

3.3.1 *Firing rate regularization*

- What, why? - Bioplausible?

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:

$$E_{\text{reg}} = \frac{1}{2} \sum_j \left(f^{\text{target}} - f_{rj}^{\text{av},t} \right)^2 , \tag{3.15}$$

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

$$f_{rj}^{\text{av},t} = \frac{1}{t} z_{rj}^{\text{total},t} \tag{3.16}$$

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):

$$\frac{\partial E_{\text{reg}}}{\partial z_{rj}^t} = \left(f^{\text{target}} - f_{rj}^{\text{av},t} \right) . \tag{3.17}$$

Note that this regularization loss differs from the firing rate regularization in Bellec 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 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

$$\Delta_{\text{reg}} W_{rji} = -\eta \; c_{\text{reg}} \sum_t \left(f^{\text{target}} - f^{\text{av},t}_{rj} \right). \tag{3.18}$$

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:

$$\Delta W_{rji} = -\eta \sum_t \left(c_{\text{reg}} \left(f^{\text{target}} - f^{\text{av},t}_{rj} \right) + L^t_{rj} \cdot \bar{e}^t_{rji} \right). \tag{3.19}$$

3.3.2 *Ridge regression*

- What, why, how? - Bioplausible?

3.3.3 *Weight decay*

- What, why, how? - Bioplausible?

3.4 BIDIRECTIONAL NETWORK

- I/A - What, why, how?

3.5 OPTIMIZER

- Show how Adam works, and how it replaces SGD

3.6 HYPERPARAMETER OPTIMIZATION

- What, why, how?

4

RESULTS

4.1 ALIF

4.2 STDP-ALIF

4.3 IZHIKEVICH

- RESULTS PLACEHOLDERS

5

DISCUSSION

- DISCUSSION PLACEHOLDER

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.

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.

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

Bellec, Guillaume et al. (2020). "A solution to the learning dilemma for recurrent networks of spiking neurons." In: *bioRxiv*, p. 738385.

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

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

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.

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.

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.

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.

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

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

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, and Geoffrey Hinton (2015). "Deep learning." In: *nature* 521.7553, pp. 436–444.

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

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

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

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

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.

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.