Philipps-Universität Marburg

Johannes Gille

Fachbereich 17
AG Allgemeine und Biologische Psychologie
AE Theoretische Kognitionswissenschaft

Learning in cortical microcircuits with multi-compartment pyramidal neurons

Supervisors:

Prof. Dr. Dominik Endres, Philipps-Universität Marburg

Dr. Johan Kwisthout, Radboud University

Contents

Chapter 1

Introduction

1.1 Motivation

The outstanding learning capabilities of the human brain have been found to be elusive and as of yet impossible to fully explain or replicate in silicio. While the power of classical machine learning solutions for some tasks has improved even beyond human capabilities in recent years, these approaches cannot serve as an adequate model of human cognition. Some of the reasons brains and machines appear irreconcilable relate to questions about network structure and neuron models. Yet more pressingly, almost all of the most powerful artificial neural networks are trained with the Backpropagation of errors algorithm, which is largely considered to be impossible for neurons to implement. Hence, Neuroscience has dismissed this algorithm in an almost dogmatic way for many years after its development, stating that the brain must employ a different mechanism to learn.

Yet in recent years, there has been a resurgence of attempts by neuroscientists towards reconciling biological and artificial neural networks despite these issues. This led to a number of experimental results indicating that brains might be capable of performing the impossible - using Backpropagation of errors to learn. Furthermore, despite rigorous efforts, no unifying alternative to this learning principle was found which performs well enough to account for the brain's vast capabilities.

Hence, there is a vibrant communitiy developing novel concepts for implementing this algorithm - or some approximation of it. These novel approaches are capable of replicating an increasing number of properties of biological brains. Nevertheless, many problems remain unsolved, and a lot of neuronal features remain unaccounted for in brain models that are capable of any kind of learning. It is this open problem, to which I want to dedicate my efforts in this thesis. After reviewing the existing literature, I have selected a promising model of learning in cortical circuits. This model uses multi-compartment neuron models and local plasticity rules to implement a variant of Backpropagation. In this project, I will investigate and attempt to further improve its concordance with data on the human neocortex. I will

use the approach of computationally modelling the model while increasingly adding biological features, attempting to retain learning performance in the process.

1.2 The Backpropagation of errors algorithm

The Backpropagation of errors algorithm (*Backprop*) (**?**) is the workhorse of modern machine learning and is able to outperform humans on a growing number of tasks (LeCun et al., 2015). Particularly for training deep neural networks it has remained popular and largely unchanged since its initial development. Its learning potential stems from its unique capability to attribute errors in the output of a network to activations of specific neurons and connections within its hidden layers. This property also forms the basis of the algorithm's name; After an initial forward pass to form a prediction about the nature of a given input, a separate backward pass propagates the arising error through all layers in reverse order. During this second network traversal, local error gradients dictate to what extent a given weight needs to be altered so that the next presentation of the same sample would elicit a lower error in the output layer. It has been argued that through this mechanism, Backprop solves the *credit assignment problem* - i.e. the question about to what degree a parameter contributes to an error signal - optimally (Lillicrap et al., 2020). With this critical information in hand, computing parameter changes that decrease error becomes almost trivial. Thus, it is naturally desirable find a solution to the credit assignment which likewise explains how the brain updates its parameters. This

1.3 Concerns over biological plausibility

While Backprop continues to prove exceptionally useful in conventional machine learning systems, it is viewed critically by many neuroscientists. For one, it relies on a slow adaptation of synaptic weights, and therefore requires a large amount of examples to learn rather simple input-output mappings. In this particular way, its performance is far inferior to the powerful one-shot learning exhibited by humans (Brea and Gerstner, 2016). Yet more importantly, no plausible mechanisms have yet been found by which biological neural networks could implement this algorithm. In fact, Backprop as an algorithm by which brains may learn has been dismissed entirely by much of the neuroscience community for decades (Grossberg, 1987; Crick, 1989; Mazzoni et al., 1991; O'Reilly, 1996). This dismissal is often focussed on three mechanisms that are instrumental for the algorithm (Whittington and Bogacz, 2019; Bengio et al., 2015; Liao et al., 2016):

1.3.1 Local error representation

Neuron-specific errors in Backprop are computed and propagated by a mechanism that is completely detached from the network itself, which requires access to the entirity of the network state. In order to compute the weight changes for a given layer, the algorithm takes as an

input the activation and synaptic weights of all downstream neurons. In contrast, plasticity in biological neurons is largely considered to be primarily dependent on factors that are local to the synapse (Abbott and Nelson, 2000; Magee and Grienberger, 2020; Urbanczik and Senn, 2014). While neuromodulators are known to influence synaptic plasticity, their dispersion is too wide to communicate neuron-specific errors. Thus, biologically plausible Backprop would require a method for encoding errors locally, i.e. close to the neurons to which they relate. This has been perhaps the strongest criticism of Backprop in the brain, as many qestions regarding biological mechanisms for both computing and storing these errors remain unanswered as of yet.

1.3.2 The weight transport problem

During the weight update stage of Backprop, errors are transmitted between layers with the same weights that are used in the forward pass. In other words, the magnitude of a neuron-specific error that is propagated through a given connection should be proportional to its impact on output loss during the forward pass. For this to work, a neuronal network implementing Backprop would require feedback connections that mirror both the connectivity and synaptic weights of the forward connections. Bidirectional connections that could theoretically back-propagate errors are common in the cortex, yet it is unclear by which mechanism pairs of synapses would be able to align. This issue becomes particularly apparent when considering long-range pyramidal projections, in which feedforward and feedback synapses would potentially be separated by a considerable distance.

1.3.3 Neuron models

Finally, the types of artificial neurons typically used in Backprop transmit a continuous scalar activation at all times, instead of discrete spikes. In theory, these activations correspond to the firing rate of a spiking neuron, giving this class of models the title *rate neurons*. Yet spike based communication requires more sophisticated neuron models. Additionally, plasticity rules for rate neurons do not necessarily have an easily derived counterpart for spiking neurons. A notable example for this issue is Backprop itself; The local error gradient of a neuron is not trivial to compute for Spiking neural networks (SNN), as a spiketrain has no natural derivative. Furthermore, a given neuron's activation in classical Backprop is computed from a simple weighted sum of all inputs. This fails to capture the complex nonlinearities of dendritic integration that are fundamental to cortical neurons (Gerstner and Naud, 2009; Sjostrom et al., 2008; Eyal et al., 2018). Finally, these abstract neurons - at least in classical Backprop - have no persistence through time. Thus, their activation is dictated strictly by the presentation of a single stimulus, in contrast to the leaky membrane dynamics exhibited by biological neurons.

Additional concerns regarding Backprop will be discussed in Section **TODO:** .

1.4 Overcoming biological implausibility

Despite these issues, Backprop has remained the gold standard against which most attempts at modelling learning in the brain eventually have to compare. Also, despite its apparent biological implausibility, it does share some notable parallels to learning in the brain. Artificial neural networks (ANN) trained with Backprop have been shown to develop similar representations to those found in brain areas responsible for comparable tasks (Yamins and DiCarlo, 2016; Whittington et al., 2018; Khaligh-Razavi and Kriegeskorte, 2014; Kubilius et al., 2016). Thus, numerous attempts have been made to define more biologically plausible learning rules which approximate Backprop to some degree. A full review of the available literature would be out of scope for this thesis, so only a few examples will be discussed in this section.

One approach to solve the issues around local error representations is, to drive synaptic plasticity through a global error signal **TODO: cite**. The appeal of this solution is that such signalling could be plausibly performed by neuromodulators like dopamine (Mazzoni et al., 1991; Seung, 2003; Izhikevich, 2007). These types solutions to not approximate Backprop, but instead lead to a kind of reinforcement learning. While some consider this the most plausible way for brains to learn, performance of global error/reward signalling stays far behind that of the exact credit assignment performed in Backprop. Additionally, this class of algorithms requires even more examples of a training dataset, and was shown to scale poorly with network size (Werfel et al., 2003). Two prominent classes of algorithms encode errors in either activation changes over time () **TODO: expand**

The weight transport problem was successfully adressed by a mechanism called *Feedback Alignment* (FA) (Lillicrap et al., 2014). This seminal paper shows that Backpropagation of errors can still learn successfully when feedback weights are random. In addition to learning to represent an input-output mapping in forward weights, Backpropagation is capable of training the network to extract information from randomly weighted instructive pathways. The authors call this process *learning to learn* and show that learning performance is even superior than classical Backprop for some tasks. This mechanism was further expanded to show that the principles of FA perform very well when biologically plausible plasticity rules are employed (Liao et al., 2016; Zenke and Ganguli, 2018). Another popular line of thought is - instead of computing local errors - to compute optimal activations for hidden layer neurons using autoencoders (Bengio, 2014; Lee et al., 2015; Ahmad et al., 2020). Approaches derived from this do not suffer from the weight transport problem, and by design does not require local error representations. While these solutions promise to solve the weight transport problem, on more complex benchmark datasets like *CIFAR* and *ImageNet* both of them fall far behind traditional Backprop (Bartunov et al., 2018).

Numerous approaches for implementing Backprop in more plausible neuron models exist, most of which employ variants of the *Leaky Integrate-and-fire* (LIF) neuron (Sporea and Grüning, 2013; Lee et al., 2016; Bengio et al., 2017; Lee et al., 2020). The aforementioned is-

sue of computing the derivative over spiketrains has been solved in several different ways, with the most prominent variant perhaps being *SuperSpike* (Zenke and Ganguli, 2018). One might therefore view this as the weakest criticism aimed at Backprop. Yet none of the employed neuron models come close to portraying the intricacies of biological neurons, and thus fail to provide explanations for their complexity.

All of these approaches successfully solve one or more concerns of biological plausibility, while still approximating Backprop to some degree. Yet none of them are able to solve all three concerns, and some of them even rely on novel mechanisms that are themselves biologically questionable. It further appears that in all but a few cases, an increase in biological plausibility leads to a decrease in performance. Thus, whether Backprop could be implemented or approximate by biological neurons remains an open question.

1.4.1 Dendrites as computational elements

The issue of oversimplified neuron models is by far the most frequent to be ommited from explanations of the biological implausibility of Backprop (See for example (Meulemans et al., 2020; Lillicrap et al., 2014)). This disregard might stem from the fact that rate-based point neurons are employed in many of the most powerful artificial neural networks. This observation might be taken as an argument that the simple summation of synaptic inputs is sufficient for powerful and generalized learning. Modelling neurons more closely to biology would by this view only increase mathematical complexity and computational cost without practical benefit. Another hypothesis states that the dominance of point neurons stems from a "somatocentric perspective" within neuroscience (Larkum et al., 2018), which stems from the technical challenges inherent to studying dendrites in vivo. The vastly different amount of available data regarding these two neuronal components might have induced a bias in how neurons are modelled computationally. Some researchers have even questioned whether dendrites should be seen as more of a 'bug' than a 'feature' (Häusser and Mel, 2003), i.e. a biological necessity which needs to be overcome and compensated for.

Yet in recent years, with novel mechanisms of dendritic computation being discovered, interest in researching and explicitly modelling dendrites has increased. Particularly the vast dendritic branches of pyramidal neurons found in the cerebral cortex, hyppocampus and amygdala, were shown to perform complex integrations of their synaptic inputs (Spruston, 2008). They recently have been shown to be capable of performing computations, which were previously assumed to require multi-layer networks (Schiess et al., 2016; Gidon et al., 2020). The size of dendritic trees is also known to discriminate regular spiking from burst firing pyramidal neurons (van Elburg and van Ooyen, 2010). **TODO: expand**

(See (Larkum, 2022) and (Poirazi and Papoutsi, 2020) for extensive reviews). These neuroscientific insights have also sparked hope that modelling dendritic compartments explicitly might aid machine learning in terms in both learning and efficiency (Chavlis and Poirazi, 2021;

Guerguiev et al., 2017; Richards and Lillicrap, 2019; Eyal et al., 2018). It appears then that, if not for computational gains, dendrites might be critical for any model that attempts to explain the power of human learning. While the network discussed here is concerned with rather simple multi-compartment models, the choice of model was strongly influenced by the recent excitement about dendrites.

1.5 Cortical microcircuits

Another feature of the brain which is often not considered in (biologically plausible) machine learning models is its intricate connectivity. This is quite understandable, as there is still some uncertainty about which parts of the brain are involved in generalized learning. It is also unclear, to what level of detail they need to be modeled. It has been shown that the connectivity patterns of cortical circuits are superior to amorphous networks in some cases (Haeusler and Maass, 2007), so there might be a computational gain from modeling network structure closer to biology. The question over network structure goes hand in hand with the choice of neuron models, as synaptic connections arrive at specific points of pyramidal neuron dendrites, depending on the origin of the connection (Felleman and Van Essen, 1991; Ishizuka et al., 1995; Larkum et al., 2018).

Several theories of cortical funcition focus more on reinforcement (Legenstein et al., 2008) or unsupervised learning (George and Hawkins, 2009; **?**). Without dismissing these theories, this thesis will adopt the viewpoint that human brains require a form of gradient descent to successfully adapt to their ever changing environments. Furthermore, we shall assume for now that this kind of learning occurs predominantly in the neocortex (Marblestone et al., 2016).

While many important exceptions have been published recently, the literature on the subject of learning historically appears to be somewhat split. On the one hand, the "machine-learning" point of view largely considers the utility of added network complexity first, with considerations of biology appearing as an afterthought**TODO: cite**. On the other hand, intricate models of cortical circuits exist, which can so far not be trained to perform tasks Potjans and Diesmann (2014); Schmidt et al. (2018); van Albada et al. (2022). Within this thesis, I hope to contribute to the body of literature between those extremes. For this, my approach will be to select a learning model that is already highly biologically plausible, and to attempt to improve its plausibility - without fully breaking the learning rule.

1.6 Model selection

The model selection progress was strongly influenced by a review article on biologically plausible approximations of Backprop (Whittington and Bogacz, 2019). The authors narrow the wide range of proposed solution down to four algorithms that are both highly performant and largely biologically plausible. The algorithms are united by requiring minimal external control,

and by the fact that they can all be described within a common framework of energy minimization (Scellier and Bengio, 2017). The first two models are Contrastive learning O'Reilly (1996), and its extension to time-continuous updates (Bengio et al., 2017). Both of these encode neuron-specific errors in the change of neural activity over time. One of their appeals is the fact that they rely on Hebbian and Anti-Hebbian plasticity. Yet in the plasticity rule also lies their greatest weakness, as synapses need to switch between the two once the target for a given stimulus is provided. This switch requires a global signal that communicates the change in state to all neurons in the network simultaneously.

The second class of models was more appealing to me, as both variants are based on the predictive coding account in Neuroscience (Rao and Ballard, 1999), which deserves an introduction.

1.6.1 Predictive coding

In this seminal model of processing in the visual cortex, each level of the visual hierarchy represents the outside world at some level of abstraction. Recurrent connections then serve communicate prediction errors and predictions up and down the hierarchy respectively, which the network attempts to reconcile. The authors showed that through rather simple computations, these prediction errors can be minimized to obtain useful representations at each level of the hierarchy. They further showed that a predictive coding network trained on natural images exhibits end-stopping properties previously found in mammalian visual cortex neurons. This work was instrumental to shaping the modern neuroscientific perspective of perception and action as a unified process. The extension of predictive coding principles from visual processing to the entire living system is promising to revolutionize neuroscience under the name of *Active inference* (Friston, 2008; Friston and Kiebel, 2009; Adams et al., 2015). By this view, the entire brain aims to minimize prediction errors with respect to an internal (generative) model of the world. A noteworthy property of this hypothesis is that it implies an agents action in the world as 'just another' way in which it can decrease discrepancies between its beliefs and sensory information. In a seminal paper, a model of the cortical microcircuit (Haeusler and Maass, 2007) was shown to have a plausible way for performing the computations required by predictive coding (Bastos et al., 2012).

While predictive coding was originally described for unsupervised learning, through a slight modification it is also capable of performing supervised learning (Whittington and Bogacz, 2017). This is the third model considered in the review paper, in which values (i.e. activations) and errors of a layer are encoded in separate, recurrently connected nodes. By employing only local Hebbian plasticity, this network is capable of approximating Backprop in multilayer perceptrons while conforming to the principles of predictive coding. The constraint on network topology was further relaxed by showing that the model is capable of approximating Backprop for arbitrary computation graphs (Millidge et al., 2022). The neuron-based predictive coding net was therefore an important contribution towards unifying the fields of Active inference and

machine learning research. As noted in a recent review article:

> Since predictive coding is largely biologically plausible, and has many potentially plausible process theories, this close link between the theories provides a potential route to the development of a biologically plausible alternative to backprop, which may be implemented in the brain. Additionally, since predictive coding can be derived as a variational inference algorithm, it also provides a close and fascinating link between backpropagation of error and variational inference. (Millidge et al., 2021)

With this in mind, we turn to the final model discussed in the review paper.

1.6.2 The Dendritic error model

The predictive coding network stores local prediction errors in nodes (i.e. neurons) close to the nodes to which these errors relate. That errors may be represented within the activation of individual neurons is a promising hypothesis with some advandages, as well as results backing it up (Hertäg and Clopath, 2022). Yet there is a competing view, by which errors elicited by individual neurons may be stored in their dendritic compartments (Guerguiev et al., 2017). The "Dendritic error model" (Sacramento et al., 2018) - as the name implies - follows this line of thought. It contains a highly recurrent network of both pyramidal- and interneurons, in which pyramidal neuron apical dendrites encode prediction errors. This view is supported by behavioral rodent experiments which show that stimulation to pyramidal neuron apical tufts in cortical layer 1 controls learning (Doron et al., 2020).

For the errors to be encoded successfully, the model requires a symmetry between feedforward and feedback sets of weights, which it has to learn prior to training. After that, apical compartments behave like the error nodes in a predictive coding network. They are silent during a feedforward network pass, and encode local prediction errors in their membrane potential when a target is applied to the output layer. Since they are a part of the pyramidal neuron, only local information is required to minimize these prediction errors through a plasticity rule for multi-compartment neurons (Urbanczik and Senn, 2014). A critical observation made in (Whittington and Bogacz, 2019) is that the dendritic error model is mathematically equivalent to their predictive coding network **TODO: expand if I have time, otherwise this will be a ref.**. All of these factors combined make the dendritic error model a promising model to help us further understand both energy minimization and deep learning in cortical circuits. While both the employed neuron and connectivity model are far behind some of the more rigorous cortical simulations, it can be considered an important step towards integrating deep learning and neuroscience (Marblestone et al., 2016).

Nevertheless, the model still suffers from some constraints with regard to its biological plausibility; Both the predictive coding network and the dendritic error network require strongly

constrained connectivity schemes, whithout which they cannot learn. This kind of specificity (in particular one-to-one relationships between pairs of neurons) are highly untypical for cortical connections (Thomson and Bannister, 2003). Hence, their exact network architectures are unlikely to be present in the cortex. The Dendritic error model additionally requires Pretraining to be capable of approximating Backprop. Both of these issues will be discussed in this thesis. Yet the most salient improvement to the network's biological plausibility is likely, to change neuron models from rate-based to spiking neurons. It has been shown that the Plasticity rule employed by the network is capable of performing simple learning tasks when adapted to spiking neurons (Stapmanns et al., 2021). Yet, (to the best of my knowledge) there are no studies investigating if this variant is capable of learning tasks on a network-level. A spiking implementation of the dendritic error network will therefore be the starting point for this thesis, upon which further analysis shall build.

Chapter 2

Methods

2.1 The dendritic error model

This section will go into detail about the dendritic error network (Sacramento et al., 2018). The model contains a somewhat complex and strongly recurrent connectivity, which poses one of the major criticisms aimed at it (Whittington and Bogacz, 2019). Much like traditional machine learning networks, it can be functionally separated into layers. Yet in this particular model, input- hidden- and output layers are quite distinct in both neuron populations and connectivity.

2.1.1 Network architecture

The basic connectivity scheme of the Model is shown in Fig. 2.1. Neurons at the input layer receive no feedback signals and serve primarily to apply a temporal low-pass filter to the stimulus which is injected directly into their membrane. Hidden layers consist of a pyramidal- and an interneuron population, which are fully connected to each other reciprocally. Both types of neurons are represented by multi-compartment neuron models with leaky membrane dynamics. Interneurons contain one somatic and one dendritic compartment, while pyramidal neurons are modeled with both a basal and an apical dendrite. Feedforward connections between layers are facilitated by all-to-all connections between their respective pyramidal neurons and innervate basal compartments. Feedback connections from superficial pyramidal neurons, as well as lateral interneuron connections arrive at the apical compartments of pyramidal neurons. Thus, a hidden layer pyramidal neuron forms two reciprocal loops, one with all interneurons in the same layer, and one with all pyramidal neurons in the next layer.

Interneurons receive feedback information from superficial pyramidal neurons in addition to their lateral connections. These feedback connections are unique in this model, as they connect one pyramidal neuron to exactly one interneuron. Instead of transmitting a neuronal

[1]Note that the input layer is displayed as having interneurons here. This appears to be a mistake in the Figure. Within the implementation, interneurons are only modelled in hidden layers.

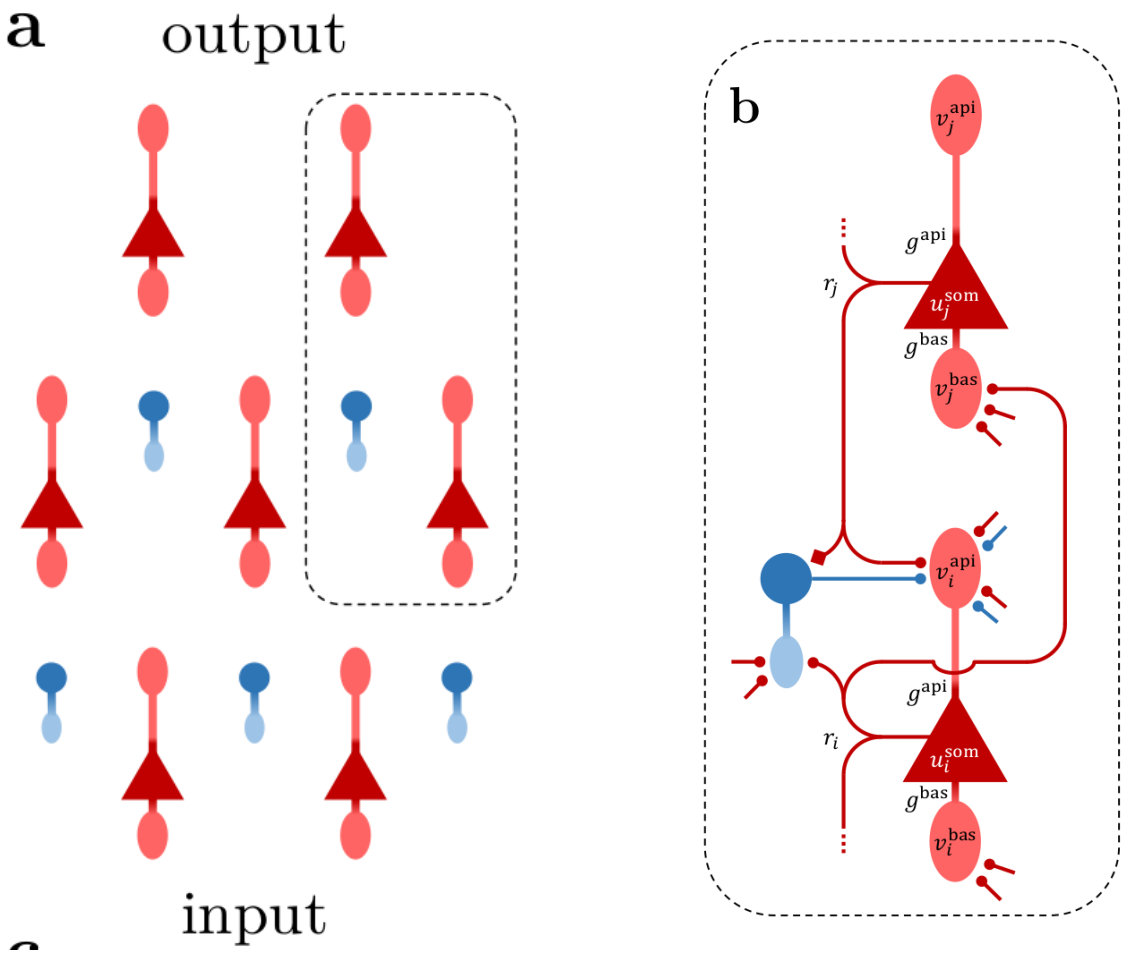

Fig. 2.1: Network structure, from Haider et al. (2021). **a:** pyramidal- (red) and interneurons (blue) in a network of three layers. Note the fact that the number interneurons in a layer is equal to the number of pyramidal neurons in the subsequent layer[1]. **b:** connectivity within the highlighted section. Feedback pyramidal-to-interneuron connections (displayed with rectangular synapse) transmit pyramidal somatic potential directly and connect to a single interneuron. This enables these interneurons to learn to match their corresponding next-layer pyramidal neurons. All other synapses (circles) transmit the neuron somatic activation $\phi(u^{som})$ and fully connect their origin and target populations.

activation as all other connections do, these connections relay somatic voltage directly. This one-to-one connectivity puts a strict constraint on the number of interneurons in a hidden layer, as it must be equal to the number of subsequent pyramidal neurons. These pairs of inter- and pyramidal neurons will henceforth be called *sister neurons*. The top-down signal serves to *nudge* interneuron somatic activation towards that of their pyramidal sisters. The purpose of an interneuron in this architecture is then, to predict the activity of its sister neuron. Any failure to do so results in layer-specific errors which in turn are the driving force of learning in this context, but more on this later.

Output layers have no interneurons, and are usually modeled as pyramidal neurons without an apical compartment. During learning, the target for the network's activation is injected into their somatic compartment. Through the feedback connections, it can propagate through the

entire network. To understand what purpose this rather complex connectivity scheme serves in our model, neuron models and plasticity rules require some elaboration.

2.1.2 Neuron models

The network contains two types of multi-compartment neurons; Pyramidal neurons with three compartments each, and interneurons with two compartments each. They integrate synaptic inputs into dendritic potentials, which in turn leak into the soma with specific conductances. Note that vector notation will be used throughout this section, and u_l^P and u_l^I denote the column vectors of pyramidal- and interneuron somatic voltages at layer l, respectively. Synaptic weights W are likewise assumed matrices of size $n \times m$, which are the number of output and input neurons of the connected populations respectively. The activation (or rate) r_l^P of pyramidal neurons is given by applying the neuronal transfer function ϕ to their somatic potentials u_l^P:

$$r_l^P = \phi(u_l^P) \tag{2.1}$$

$$\phi(x) = \begin{cases} 0 & \text{if } x < -\varepsilon \\ \gamma \, log(1 + e^{\beta(x-\theta)}) & \text{if } -\varepsilon \leq x < \varepsilon \\ \gamma \, x & \text{otherwise} \end{cases} \tag{2.2}$$

where ϕ acts component wise on u and can be interpreted as a smoothed variant of ReLu (sometimes called *Softplus*) with scaling factors $\gamma = 1$, $\beta = 1$, $\theta = 0$. Splitting the computation with the threshold parameter $\varepsilon = 15$ does not alter its output much, but instead serves to prevent overflow errors for large absolute values of x.

As mentioned before, pyramidal and interneurons are modeled as rate neurons with leaky membrane dynamics and multiple compartments. Where applicable, they will be differentiated with superscripts P and I respectively. The basal and apical dendrites of pyramidal neurons are denoted with superscripts *bas* and *api* respectively, while interneuron dendrites are simply denoted *dend*. The derivative somatic membrane potentials of layer l pyramidal neurons is given by:

$$C_m \dot{u}_l^P = -g_l u_l^P + g^{bas} v_l^{bas} + g^{api} v_l^{api} \tag{2.3}$$

where g_l is the somatic leakage conductance, and C_m is the somatic membrane capacitance which will be assumed to be 1 from here on out. v_l^{bas} and v_l^{api} are the membrane potentials of basal and apical dendrites respectively, and g^{bas} and g^{api} their corresponding coupling conductances. Dendritic compartments in this model have no persistence between simulation steps. Thus, they are defined at every timestep t through incoming weight matrices and presynaptic activities:

$$v_l^{bas}(t) = W_l^{up}\ \phi(u_{l-1}^P(t)) \tag{2.4}$$
$$v_l^{api}(t) = W_l^{pi}\ \phi(u_l^I(t))\ +\ W_l^{down}\ \phi(u_{l+1}^P(t)) \tag{2.5}$$

The nomenclature for weight matrices conforms to Haider et al. (2021) where they are indexed by the layer in which their target neurons lie, and belong to one of four populations: Feedforward and feedback pyramidal-to-pyramidal connections arriving at layer l are denoted W_l^{up} and W_l^{down} respectively. Lateral pyramidal-to-interneuron connections are denoted with W_l^{ip} and their corresponding feedback connections with W_l^{pi}.

Interneurons integrate synaptic information by largely the same principle, but instead of top-down signals from their sister neurons arriving at an apical compartment, it is injected directly into the soma.

$$C_m \dot{u}_l^I = -g_l u_l^I + g^{dend} v_l^{dend} + i^{nudge,I} \tag{2.6}$$
$$i^{nudge,I} = g^{nudge,I} u_{l+1}^P \tag{2.7}$$
$$v_l^{dend} = W_l^{ip}\ \phi(u_l^P) \tag{2.8}$$

where $g^{nudge,I}$ is the interneuron nudging conductance, and u_{l+1}^P is the somatic voltage of pyramidal neurons in the next layer. Pyramidal neurons in the output layer N effectively behave like interneurons, as they receive no input to their apical compartment. Instead, the target activation u^{tgt} is injected into their soma:

$$C_m \dot{u}_N^P = -g_l u_N^P + g^{bas} v_N^{bas} + i^{nudge,tgt} \tag{2.9}$$
$$i^{nudge,tgt} = g^{nudge,tgt} u^{tgt} \tag{2.10}$$

These neuron dynamics correspond closely to those by Urbanczik and Senn (2014), including the extension to more than two compartments which was proposed in the original paper. It should be noted however, that they are simplified in some ways. For example, dendritic couplings and nudging are not relative to the somatic or some reversal potential (i.e. $i^{nudge,tgt} = g^{nudge,tgt}(u^{tgt} - u_N^P)$), but are only dependent on absolute currents. Additionally, the strict separation of excitatory and inhibitory synaptic integration (cf. Section **TODO: write about this**), as well as additional nonlinearities in the plasticity are omitted. These simplifications do increase computational speed and allow for a much simpler approximation of network dynamics via a steady-state. Yet they do come at the cost of omitting neuroscientific insights from the model, which will be discussed later.

2.2 Urbanczik-Senn Plasticity

The synapses in the network are all modulated according to variations of the "Urbanczik-Senn" plasticity rule (Urbanczik and Senn, 2014), which will be discussed in this section. Note that as for the neuron model, the dendritic error model slightly simplifies some equations of the plasticity rule from its original implementation.

2.2.1 Derivation

The plasticity rule is defined for postsynaptic neurons which have one somatic and at least one dendritic compartment, to the latter of which synapses of this type can connect. Functionally, synaptic weights are changed in such a way, as to minimize discrepancies between the somatic activity and dendritic potential. This discrepancy is called the *dendritic prediction error*, and is computed from a hypothetical dendritic activation. The change in weight for a synapse from neuron j to the basal compartment of a pyramidal neuron i is given by:

$$\dot{w}_{ij} = \eta \ (\phi(u_i^{som}) - \phi(\hat{v}_i^{bas})) \ \phi(u_j^{som})^T \tag{2.11}$$

$$\hat{v}_i^{bas} = \alpha \ v_i^{bas} \tag{2.12}$$

with learning rate η, and u^T denoting the transposition of the vector u (which is by default assumed a column vector). The dendritic prediction $\hat{v}_i^{bas}$ is a scaled version of the dendritic potential by the constant factor α, which is calculated from coupling and leakage conductances. As an example, basal dendrites of pyramidal neurons in Sacramento et al. (2018) are attenuated by $\alpha = \frac{g^{bas}}{g_l + g^{bas} + g^{api}}$. A key property of this value for α is, that dendritic error is 0 when the only input to a neuron stems from the given dendrite. In other words, the dendrite predicts somatic activity perfectly, and no change in synaptic weights is required. Neuron- and layer-specific differences in α, as well as an analytical derivation are detailed in (Sacramento et al., 2018).

If a current is injected into the soma (or in this case, into a different dendrite), a dendritic error arises, and plasticity drives synaptic weights to minimize it. In addition to the learning rate η, the change in weight $\dot{w}_{ij}$ is proportional to presynaptic activity $\phi(u_j^{som})$. Therefore, a dendritic error arising without presynaptic contribution does not elicit a change in that particular synapse. This ensures that only synapses are modified which recently influenced the postsynaptic neuron, providing a form of credit assignment. Updates for the weight matrices in a hidden layer l of the dendritic error model are given by:

$$\dot{w}_l^{up} = \eta_l^{up} \ (\phi(u_l^P) - \phi(\hat{v}_l^{bas})) \ \phi(u_{l-1}^P)^T \tag{2.13}$$

$$\dot{w}_l^{ip} = \eta_l^{ip} \ (\phi(u_l^I) - \phi(\hat{v}_l^{dend})) \ \phi(u_l^P)^T \tag{2.14}$$

$$\dot{w}_l^{pi} = \eta_l^{pi} \ - v_l^{api} \ \phi(u_l^I)^T \tag{2.15}$$

$$\dot{w}_l^{down} = \eta_l^{down} \ (\phi(u_l^P) - \phi(w_l^{down} r_{l+1}^P)) \ \phi(u_{l+1}^P)^T \tag{2.16}$$

Each set of connections is updated with a specific learning rate η and a specific dendritic error term. The purpose of these particular dendritic errors will be explained in Section 2.3. Note that pyramidal-to-pyramidal feedback weights w_l^{down} are not plastic in the present simulations and are only listed for completeness, see Section 5.5.

2.3 The self-predicting network state

In the dendritic error model neuron dynamics, plasticity rules and network architecture form an elegant interplay, which will be explained in this section. Since each interneuron receives a somatic nudging signal from its corresponding sister neuron, incoming synapses from lateral pyramidal neurons adapt their weights to match feedforward pyramidal-to-pyramidal weights. In intuitive terms; Feedforward pyramidal-to-pyramidal weights elicit a certain activation in the subsequent layer, which is fed back into corresponding interneurons. Hence, in the absence of incoming connections, nudging from sister neurons causes interneurons to take on a proportional somatic potential. In order to minimize the dendritic error term in Equation 2.14, pyramidal-to-interneuron weight matrices at every layer must match these forward weights ($w_l^{ip} \approx \rho w_l^{up}$) up to some scaling factor ρ. The exact value for ρ is parameter-dependent and immaterial for now. As long as no feedback information arrives at the pyramidal neurons, plasticity drives synaptic weight to fulfill this constraint. Note, that this alignment of two separate sets of outgoing weights is achieved with only local information. Therefore, this mechanism could plausibly align the weights of biological synapses that are physically separated by long distances.

Next, consider the special case for interneuron-to-pyramidal weights in Equation 2.15, in which plasticity does not serve to reduce discrepancies between dendritic and somatic potential. The error term is instead defined solely by the apical compartment voltage[2]. Thus, plasticity in these synapses works towards silencing the apical compartment. The apical compartments also receive feedback from superficial pyramidal neurons, whose synapses will be considered non-plastic for now. As shown above, interneurons each learn to match their respective sister neuron activity. Thus, silencing of apical compartments can only be achieved by mirroring the

[2]In strict terms, it is defined by the deviation of the dendritic potential from its specific reversal potential. Since that potential is zero throughout, $-v_l^{api}$ remains as the error term.

pyramidal-to-pyramidal feedback weights ($w_l^{pi} \approx -w_l^{down}$).

When enabling plasticity in only these two synapse types, the network converges on the "**self-predicting state**" (Sacramento et al., 2018). This state is defined by a minimization of four error metrics at each hidden layer l:

- The symmetries between feedforward ($w_l^{ip} \rightarrow \rho w_l^{up}$) and feedback ($w_l^{pi} \rightarrow -w_l^{down}$) weights. *Mean squared error (MSE)* between these pairs of matrices will be called **Feedforward -** and **Feedback weight error** respectively.
- Silencing of pyramidal neuron apical compartments ($v_l^{api} \rightarrow 0$). Mean absolute apical compartment voltage within a layer is called the **Apical error**.
- Equal activations in interneurons and their respective sister neurons ($\phi(u_l^I) \rightarrow \phi(u_{l+1}^P)$). The mean squared error over these vectors is called the **Interneuron error**.

The network does not ever reach a state in which all of these error terms are zero. In the original implementation, these deviations are minute and can likely be explained with floating point conversions. Since it is impossible to replicate the timing of the original precisely within NEST, the NEST simulations deviate more strongly from this ideal. The key insight here is that this state is not clearly defined by absolute error thresholds, but is rather flexible. Thus, networks are able to learn successfully even when their weights are initialized imperfectly.

An analysis of the equations describing the network reveals that the idealized self-predicting state forms a stable point of minimal energy. When Interneuron error is zero, the nudging conductance is predicted perfectly, thus effectively disabling plasticity in incoming synapses. Likewise, a silenced apical compartment will disable plasticity in all incoming synapses from interneurons. Similarly, the apical compartment is also the driving factor for the dendritic error of feedforward synapses (Equation 2.13), since it leaks into the soma when active[3]. Thus, in the self-predicting state all plasticiy in the network is disabled, and the state is stable regardless of the kind of stimulus injected into the input layer. Next, notice how information flows backwards through the network; All feedback pathways between layers ultimately pass through the apical compartments of pyramidal neurons. Thus, successful silencing of all apical compartments implies that no information can travel backwards between layers. As a result, the network behaves strictly like a fully connected feedforward network consisting only of pyramidal neurons. The recurrence within this network is in balance, and completely cancels out its own effects. This holds true as long as the network only receives external stimulation at the input layer. One interpretation of this is, that the network has learned to predict its own top-down input. A failure by interneurons to fully explain (i.e. cancel out) top-down

[3]This feature is actually rather important when contemplating biological neurons using the Urbanczik-Senn plasticity. In the original paper, currents were injected directly into the soma to change the error term. The introduction of a second dendrite which performs that very task is much more useful, as originally described by the authors.

input thus results in a prediction error, encoded in deviation of apical dendrite potentials from their resting state. This prediction error in turn elicits a cascade of plasticity in several synapses, which drives the network towards a self-predicting state that is congruent with the novel top-down signal. Therefore, these neuron- specific prediction errors are the driving force of learning in these networks.

2.4 Training the network

Starting with a network in the self-predicting state, performing time-continuous supervised learning then requires the injection of a target activation into the network's output layer alongside with a stimulus at the input layer. Since output layer neurons feed back into both interneurons and pyramidal neurons of the previous layer, local prediction errors arise. Synapses activate and drive to minimize the prediction errors, which requires the network to replicate the target activation from activations and weights of the last hidden layer. Note that this mechanism is not exclusive to the last two layers. Any Apical errors cause a change in somatic activity, which previous layers will fail to predict. Thus, errors are propagated backwards through the entire network, causing error minimization at every layer. See the Supplementary analysis of Sacramento et al. (2018) for a rigorous proof that this type of network does indeed approximate the Backpropagation algorithm.

Classical Backprop relies on a strict separation of a forward pass of some stimulus, and subsequent a backwards pass dependent on the arising loss at the output layer. Since the present network is time-continuous, stimulus and target activation are injected into the network simultaneously. These injections are maintained for a given presentation time t_{pres}, in order to allow the network to calculate errors through its recurrent connections before slowly adapting weights. Particluarly for deep networks, signals travelling from both the input and output layer require some time to balance out and elicit the correct dendritic error terms. This property poses the most significant drawback of this type of time-continuous approximation of Backprop: The network tends to overshoot activations in some neurons, which in turn causes an imbalance between dendritic and somatic compartments. This effect causes the network to change synaptic weights away from the desired state during the first few milliseconds of a stimulus presentation. The solution Sacramento et al. found for this issue was to drastically reduce learning rates, while increasing stimulus presentation time. This solution is sufficient to prove that plasticity in this kind of network is able to perform error propagation, but still has some issues. Most notably, training is highly inefficient and computationally intensive. A closer investigation of the issue together with a different solution will be discussed in Section 2.8.

2.5 The NEST simulator

One of the key research questions motivating this thesis is whether the network would be able to learn successfully when employing spike-based communication instead of the rate neurons for which it was developed. As a framework for the spike-based implementation two options were considered: The first one was to use the existing implementation of the network which employs the Python frameworks `PyTorch` and `NumPy`, and expand it to employ spiking neurons. PyTorch does in principle support spiking communication between layers, but is streamlined for implementing less recurrent and less complex network and neuron models. Another concern is efficiency; PyTorch is very well optimized for computing matrix operations on dedicated hardware. This makes it a good choice for simulating large networks of rate neurons, which transmit all of their activations between layers at every simulation step. Spiking communication between leaky neurons is almost antithetical to this design philosophy and thus can be expected to perform comparatively poorly when using this backend.

The second option was to use the NEST simulator (nest-simulator.readthedocs.io, Gewaltig and Diesmann (2007)), which was developed with highly parallel simulations of large spiking neural networks in mind. It is written in C++ and uses the *Message Passing Interface* (MPI) to efficiently communicate events between both threads and compute nodes. One design pillar of the simulator, which is particularly relevant for this project, is the event-based communication scheme that underpins all simulated nodes. It ensures that communication bandwidth at every simulation step is only used by the subset of nodes which transmit signals at that time step, which is particularly efficient for spiking communication. Another important advantage of the NEST simulator is, that an event-based implementation of the Urbanczik-Senn plasticity alongside a corresponding neuron model had already been developed for it. Therefore, it was decided to implement the spiking neuron model in the NEST simulator.

The simulator has one particular limitation which needs to be considered. As communication between physically separate compute nodes takes time, Events[4] in NEST can not be handled in the same simulation step in which they were sent. Thus, NEST enforces a synaptic transmission delay of at least one simulation step for all connections. This property is integral to other parallel simulation backends (Hines and Carnevale, 1997) as well as neuromorphic hardware (Davies et al., 2018). It may not even considered a limitation by some, as synaptic transmission within biological neurons is never instantaneous either (Kandel et al., 2021). Yet particularly with regard to the relaxation period issue of this model (cf. Section 2.8), it can be expected to affect performance.

[4] An Event in NEST is an abstract C++ Class that is created by neurons, and transmitted across threads and compute nodes by the Simulator. A Multitude of Event types are provided (i.e. `SpikeEvent`, `CurrentEvent`, `RateEvent`), each able to carry specific types of payload and being processed differently by postsynaptic neurons.

2.6 Transitioning to spiking communication

The spiking neuron models rely heavily on the NEST implementation from Stapmanns and colleagues (Stapmanns et al., 2021), which was used show that spiking neurons are able to perform learning tasks that were designed for the rate neurons described in Urbanczik and Senn (2014). The existing model is an exact replication of the Urbanczik-Senn neuron in terms of membrane dynamics. The critical update of the NEST variant is that instead of transmitting their hypothetical rate $r = \phi(u)$ at every time step, these neurons emit spikes in a similar way to stochastic binary neurons (Ginzburg and Sompolinsky, 1994). The number of spikes to be generated during a simulation step n is determined by drawing from a Poisson distribution, which takes r as a parameter:

$$P\{n \text{ spikes during } \Delta t\} = e^{-r\Delta t}\frac{(r\ \Delta t)^n}{n!} \tag{2.17}$$

$$\langle n \rangle = r\ \Delta t \tag{2.18}$$

where Δt denotes the integration time step of the simulator, which will be assumed to be $0.1ms$ from here on out. $\langle n \rangle$ denotes the expected number of spikes to be emitted in a simulation step. Note that this mechanism makes the assumption that more than one spike can occur per simulation step. NEST was developed with this possibility in mind and provides a *multiplicity* parameter for SpikeEvents, which is processed at the postsynaptic neuron. As the high spike frequencies resulting from this could not occur in biological neurons, the model is also capable of simulating a refractory period. For this, the number of spikes per step is limited to 1, and the spiking probability is set to 0 for the duration of the refractory period t_{ref}. The probability of at least one spike occurring within the next simulation step is given the inverse probability of no spike occurring. Thus, when inserting $n = 0$ into Equation 2.17, the probability of eliciting at least one spike within the next simulation step can be derived as:

$$P\{n \geq 1\} = 1 - e^{-r\Delta t} \tag{2.19}$$

Drawing from this probability then determines whether a spike is sent during that step, henceforth denoted with the function $s(t)$, which outputs 1 if a spike is sent during the interval $[t, t + \Delta t]$, and 0 otherwise.

In order to implement the plasticity rule for spiking neurons, dendritic compartments need to be modeled with leaky dynamics. These dynamics are fundamentally the same as those described for the somatic compartment. Thus, the basal compartment of a pyramidal neuron j evolves according to:

$$C_m^{bas}\dot{v}_j^{bas} = -g_l^{bas}\ v_j^{bas} + \sum_{i\in I} W_{ji}s_i(t) \tag{2.20}$$

with presynaptic neurons I, and membrane capacitance C_m^{bas} and leakage conductance g_l^{bas} being specific to the basal dendrite. Note that these equations are calculated individually for each neuron and do not employ the matrix notation used for layers of rate neurons. Pyramidal apical and interneuron dendritic compartments evolve by the same principle and with largely the same parameters.

2.7 Event-based Urbanczik-Senn plasticity

One major challenge in implementing this architecture with spiking neurons is the Urbanczik-Senn plasticity introduced in Section 2.2. Since the plasticity rule is originally defined for rate neurons, computing the updates for spiking neurons requires some additional effort. Fortunately, this problem has already been solved in NEST for two-compartment neurons (Stapmanns et al., 2021). This Section will discuss its algorithm and its implementation.

Since NEST is an event-based simulator, most of the plasticity mechanisms developed for it compute weight changes at the location (i.e. thread and compute node) of the postsynaptic neuron whenever an Event is received. This has several advantages; It allows the thread that created the Event to continue processing neuron updates instead of having to synchronize with all threads that manage recipient neurons. More importantly, this feature mirrors the local properties of most biologically plausible synaptic plasticity models, as these are often considered to be primarily dependent on factors that are local to the synapse (Magee and Grienberger, 2020). For a spiking implementation of the Urbanczik-Senn plasticity, dendritic errors at every time step are required instead of just a scalar trace at the time of a spike, as would be the case for STDP. Thus, a mechanism for managing these errors was required, for which two basic possibilities were considered:

In a **Time-driven scheme**, dendritic errors are made available to synapses at every timestep, and weight changes are applied instantaneously. This approach is in principle an adaptation of the original computations for spiketrains. Its main drawback is that calls to the synaptic update function are as frequent as neuron updates - for all synapses. Particularly for large numbers of incoming synapses, as is common for simulations of cortical pyramidal neurons (Potjans and Diesmann, 2014; Vezoli et al., 2004), this requires numerous function calls per time step. Therefore, this approach proved costly in terms of computational resources.

An **Event-driven scheme** on the other hand, updates synaptic weights only when a spike is sent through the synapse. A history of the dendritic error is stored at the postsynaptic neuron, which is read by each synapse when a spike is transmitted in order to compute weight

changes. As the history of dendritic error applies equally to all incoming synapses, it only needs to be recorded once at the neuron. Alongside each entry in the history, a counter is stored and incremented whenever a synapse has read the history at that time step. Once all synapses have read out an entry, it is deleted. Thus, the history dynamically grows and shrinks during simulation and is only ever as long as the largest inter-spike interval (ISI) of all presynaptic neurons. This approach proves to be more efficient in terms of computation time, since fewer calls to the update function are required per synapse. It does come at the cost of memory consumption, as the history can grow particularly large for simulations with low in-degrees or large ISI[5]. During testing, the Event-based schemed proved substantially more efficient for many network types. This did however introduce the challenge of retroactively computing weight changes from the time of the last spike upon arrival of a new spike.

2.7.1 Integrating weight changes

Stapmanns et al. describe the Urbanczik-Senn plasticity rule based on the general equation for weight changes, while omitting obsolete parameters:

$$\dot{w}_{ij}(t) = F(s_j^*(t), V_i^*(t)) \tag{2.21}$$

where the change in weight $\dot{w}_{ij}$ of a synapse from neuron j to neuron i at time t is given by a function F that depends on the postsynaptic membrane potential V_i^* and the presynaptic spiketrain s_j^*. The $*$ operator denotes a causal function, indicating that a value $V_i^*(t)$ potentially depends on all previous values of $V_i(t' < t)$. One can formally integrate Equation 2.21 in order to obtain the weight change between two arbitrary time points t and T:

$$\Delta w_{ij}(t, T) = \int_t^T dt' F[s_j^*, V_i^*](t') \tag{2.22}$$

This integral forms the basis of computing the change in weight between two arriving spikes. Thus, at the implementational level, t is usually the time of the last spike that traversed the synapse, and T is the current `biological_time`[6]. For spiking neurons, it is necessary to

[5]It should also be noted that in this approach requires redundant integration of the history by every synapse. Stapmanns et al. propose a third solution, in which this integration is performed once whenever a spike is transmitted through any incoming connection, with the resulting weight change being applied to all synapses immediately. This approach proved to be even more efficient for some network configurations, but is incompatible with simulations where incoming synapses have heterogeneous synaptic delays due to the way that these delays are processed by the NEST simulator. See Section 3.1.3 in Stapmanns et al. (2021) for a detailed explanation.

[6]This term is adopted from the NEST convention, where it describes the time in *ms* which the simulator has computed. In other words, it is the number of simulation steps times Δt, not to be confused with a simulation's hardware-dependent runtime (sometimes also called *wall clock time* (Van Albada et al., 2018)).

approximate the presynaptic rate ($r_j = \phi(u_j)$). For this, a well established solution is to transform the spiketrain s_j into a decaying trace using an exponential filter kernel κ:

$$\kappa(t) = H(t)\frac{1}{t}e^{\frac{-t}{\tau_\kappa}} \tag{2.23}$$

$$H(t) = \begin{cases} 1 & \text{if } t > 0 \\ 0 & \text{if } t \leq 0 \end{cases} \tag{2.24}$$

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t')g(t-t')dt' \tag{2.25}$$

$$s_j^* = \kappa_s * s_j. \tag{2.26}$$

with filter time constant τ_κ. The trace is computed by convolving (Equation 2.25) the spiketrain with the exponential filter kernel κ. The filter uses the Heaviside step function $H(t)$, and is therefore only supported on positive values of t (also called a one-sided exponential decay kernel). This property is important, as integration limits of the convolution can be truncated when f and g are both only supported on $[0, \infty)$:

$$(f * g)(t) = \int_0^t f(t')g(t-t')dt' \tag{2.27}$$

Since spikes naturally only occur for $t > 0$, this simplified integral allows for a much more efficient computation of the convolution. The Function F on the right-hand side of Equation 2.21 can therefore be rewriten as:

$$F[s_j^*, V_i^*] = \eta\,\kappa * (V_i^* s_j^*) \tag{2.28}$$

$$V_i^* = (\phi(u_i^{som}) - \phi(\hat{v}_i^{dend})) \tag{2.29}$$

this notation seems slightly abusive, as neither side considers t, but it is taken precisely from Stapmanns et al. (2021) with learning rate η. V_i^* then is the dendritic error of the dendrite that the synapse between j and i is located at[7]. Writing out the convolutions in Equation 2.22 explicitly, we obtain

[7]The dendritic error here is defined as the difference between two hypothetical rates based on the arbitrary function ϕ. The original implementation uses the difference between the actual postsynaptic spiketrain and this dendritic prediction ($V_i^* = (s_i - \phi(\hat{v}_i^{dend}))$). Furthermore, Stapmanns et al. show that generating a spiketrain from the dendritic potential ($V_i^* = (s_i - s_i^{dend})$) also results in successful learning, although at the cost of additional training time. The rate-based variant was chosen in order to not hinder learning performance any more than necessary.

$$\Delta w_{ij}(t,T) = \int_t^T dt' F[s_j^*, V_i^*](t') \tag{2.30}$$

$$= \int_t^T dt' \ \eta \int_0^{t'} dt'' \ \kappa(t'-t'') V_i^*(t'') s_j^*(t'') \tag{2.31}$$

Computing this Equation directly is inefficient due to the nested integrals. Yet, it is possible to break up the integrals into two simpler computations and rewrite the weight change as:

$$\Delta W_{ij}(t,T) = \eta \left[I_1(t,T) - I_2(t,T) + I_2(0,t) \left(1 - e^{-\frac{T-t}{\tau_\kappa}}\right)\right] \tag{2.32}$$

$$I_1(a,b) = \int_a^b dt \ V_i^*(t) s_j^*(t) \tag{2.33}$$

$$I_2(a,b) = \int_a^b dt \ e^{-\frac{b-t}{\tau_\kappa}} V_i^*(t) s_j^*(t) \tag{2.34}$$

$$\tag{2.35}$$

See Section **TODO: ref** for a rigorous proof that this is in fact the desired integral. **That proof is currently part of the appendix, as the original proof (Stapmanns et al., 2021) has errors. Should I keep it or refer to the original paper?** The resulting equations allow for a rather efficient computation of weight changes compared to the complex integral described in Equation 2.31. This integration is performed whenever a spike traverses a synapse. It generalizes to all special cases in Equations 2.13-2.16, as long as the appropriate dendritic error is stored by the postsynaptic neuron.

2.8 Latent Equilibrium

The most significant drawback of the Sacramento model is the previously mentioned requirement for long stimulus presentation times and appropriately low learning rates. This makes the network prohibitively inefficient for the large networks required for complex learning tasks. Sacramento et al. developed a steady-state approximation of their network which models the state of the network after it has balanced out in response to a stimulus-target pair. It does not suffer from these issues and shows that their model can in principle solve more demanding learning tasks such as MNIST. Yet these types of approximation are much further detached from biological neurons than the original model and thus do not lend themselves well to an investigation of biological plausibility (Gerstner and Naud, 2009). Furthermore, the approximation is unsuitable for and investigation of spike-based communication, since the steady state of both network ideally would be the same. Thus, neither the fully modeled neuron dynamics nor the steady-state approximation are suited for complex learning tasks. A substantial

improvement to rate neurons which promises to solve this dilemma was developed by Haider et al. (2021), and will be discussed here.

The requirement for long stimulus presentation times of the dendritic error network is caused by the slow development of leaky neuron dynamics, and is therefore not unique to this model. When a stimulus-target pair is presented to the network, membrane potentials in all neurons slowly evolve until a steady state is reached. The time until a network of has reached this state after a change in input is called the *relaxation period* following Haider et al. (2021). Given a membrane time constant τ_m, a feedforward network with N layers of leaky neurons thus has a relaxation time constant of $N\tau_m$. Yet in our case, a target activation simultaneously injected into the output neurons slowly propagates backwards through the highly recurrent network. Neurons at early layers require all subsequent layers to be fully relaxed in order to correctly compute their dendritic error terms, effectively being dependent on two network passes. Haider et al. state that this kind of network therefore requires $2N\tau_m$ to relax in response to a given input-output pairing. This prediction proved to be slightly optimistic in experiments, as shown in Fig. 2.3.

This is a major issue, as it implies that plasticity during the first few miliseconds of a stimulus presentation is driven by faulty error terms. The network thus tends to 'overshoot', and needs to undo the synaptic weight changes made during the relaxation period in the later phase of a stimulus presentation, in order to make tangible progress on the learning task. Haider et al. call this issue the "relaxation problem" and suggest that it might be inherent to most established attempts at biologically plausible Backpropagation algorithms (Whittington and Bogacz, 2017; Guerguiev et al., 2017; Sacramento et al., 2018; Millidge et al., 2020).

The choice to simply increase presentation time to compensate for the relaxation period is therefore somewhat problematic. It implicitly tolerates adverse synaptic plasticity in all synapses, which are counteracted by enforcing the desired plasticity for a longer time. Physiological changes that are meant to immediately be undone are of course an inefficient use of a brain's resources, which can be considered highly untypical for a biological system. One possible solution to this is to decrease synaptic time constants and remove the temporal filtering of stimulus injections. Yet this does not solve the fundamental issue that during a substantial portion of stimulus presentations, the network is driven by erroneous plasticity. Removing temporal filtering does decrease the length of the relaxation period, but causes a drastic increase in dendritic error values during that period. Therefore, while improving response time, this change effectively impedes learning. Another possible solution is to disable plasticity for the first few milliseconds of stimulus presentation. After the network has relaxed, the plasticity rules produce useful weight changes and learning rates can consequently be safely increased. Yet a mechanism by which neurons could implement this style of phased plasticity is yet to be found, making this approach questionable in terms of biological plausibility. Furthermore, it introduces a requirement for external control to the network, a trait that is considered highly

undesirable for approximations of Backprop (Whittington and Bogacz, 2019). Ideally, the relaxation period would be skipped or shortened, in order to reduce the erroneous plasticity. This would allow for a loosening of the constraints put on presentation time and learning rates, thus increasing computational efficiency.

The approach proposed by Haider et al. is to change the parameter of the activation function ϕ, a mechanism called *Latent Equilibrium* (LE). Neurons in the original dendritic error network (henceforth called *Sacramento neurons*) transmit a function of their somatic potential u_i, which is updated through Euler integration at every simulation step (Equation 2.36). In contrast, neurons using Latent Equilibrium (henceforth called *LE neurons*) transmit a function of what the somatic potential is expected to be in the future. To calculate this expected future somatic potential $\breve{u}$, the integration is performed with a larger Euler step:

$$u_i(t + \Delta t) = u_i(t) + \dot{u}_i(t)\ \Delta t \tag{2.36}$$

$$\breve{u}_i(t + \Delta t) = u_i(t) + \dot{u}_i(t)\ \tau_{eff} \tag{2.37}$$

Instead of broadcasting their rate based on the current somatic potential ($r_i(t) = \phi(u_i(t))$), LE neurons send their predicted future activation, denoted as $\breve{r}_i(t) = \phi(\breve{u}_i(t))$. The degree to wich LE neurons look ahead is determined by the *effective membrane time constant* $\tau_{eff} = \frac{C_m}{g_l + g^{bas} + g^{api}}$. This time constant takes into account the conductance with which dendritic compartments leak into the soma, which is a key driving factor for the speed at which the network relaxes. Any computations that employ or relate to this prediction of future network states will henceforth be referred to as *prospective* and denoted with a breve ($\breve{x}$).

When employing the default parametrization given by Haider et al. (Table 5.1), τ_{eff} is slightly lower than reported pyramidal neuron time constants (McCormick et al., 1985) at approximately $5.26ms$. When presynaptic neurons employ prospective dynamics, postsynaptic neurons approach their steady state much more quickly, as depicted in Fig. 2.2. In intuitive terms, prospective activation is more strongly dependent on the derivative membrane potential compared to the instantaneous activation. This results in drastic changes in activation in response to changes in the somatic membrane potential. While this can lead to an overshoot of postsynaptic activity, under careful parametrization it strongly decreases response time.

When employing prospective dynamics in the dendritic error networks, local error terms of pyramidal- and interneurons relax much faster, as shown in Fig. 2.3. These results highlight the superiority of LE for learning in this network, as the relaxation period is almost instantaneous. In contrast, the error terms in the original dendritic error network drive random synaptic plasticity even when the network is fully trained on a given dataset and is able to make accurate predictions. Thus, both the issue of redundant weight changes, as well as the concern over response and learning speed can be solved by LE. The authors furthermore show, that

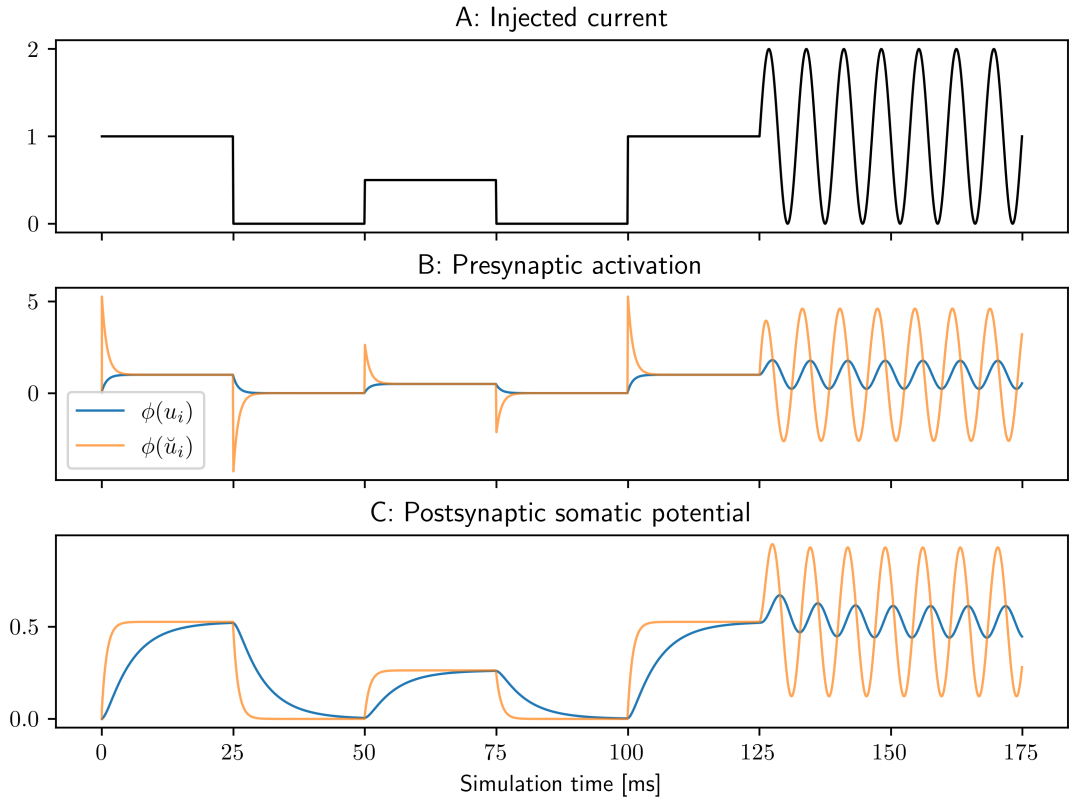

Fig. 2.2: Signal transmission between an input neuron i and a hidden layer pyramidal neuron j. Depicted are activations for the original Sacramento model and prospective activation using Latent Equilibrium. **A:** Current injected into the input neuron. Membrane potential slowly adapts to match this (not shown) **B:** Activation of the input neuron using instantaneous- $\phi(u_i)$ (blue), and prospective activation $\phi(\breve{u}_i)$ (orange). Note how strongly prospective activation reacts to changes in somatic voltage, leading to 'bursts' in neuron output. After the input neuron has reached its relaxed state ($\dot{u}_i = 0$), both types of neuron evoke the same activation. **C:** Somatic potential u_j of the pyramidal neuron responding to signals sent from the input neuron (color scheme as in B).

learning with this mechanism is indifferent to presentation times or effective time constant for rate neurons. In addition to using the prospective somatic potential for the neuronal transfer function, it is also used in the plasticity rule of LE neurons. The Urbanczik-Senn plasticity is therefore updated to compute dendritic error from prospective somatic activations and a non-prospective dendritic potential $\dot{w}_{ij} = \eta \ (\phi(\breve{u}_i^{som}) - \phi(\hat{v}_i^{bas})) \ \phi(\breve{u}_j^{som})^T$. Much like for the transfer function, this change serves to increase the responsiveness of the network to input changes.

2.9 Implementational details

Building on the neuron and plasticity model from Stapmanns et al. (2021), a replicate model of the pyramidal neuron which employs spiking communication was developed in NEST. The existing Urbanczik-Senn neuron was expanded to three compartments, and storage and readout

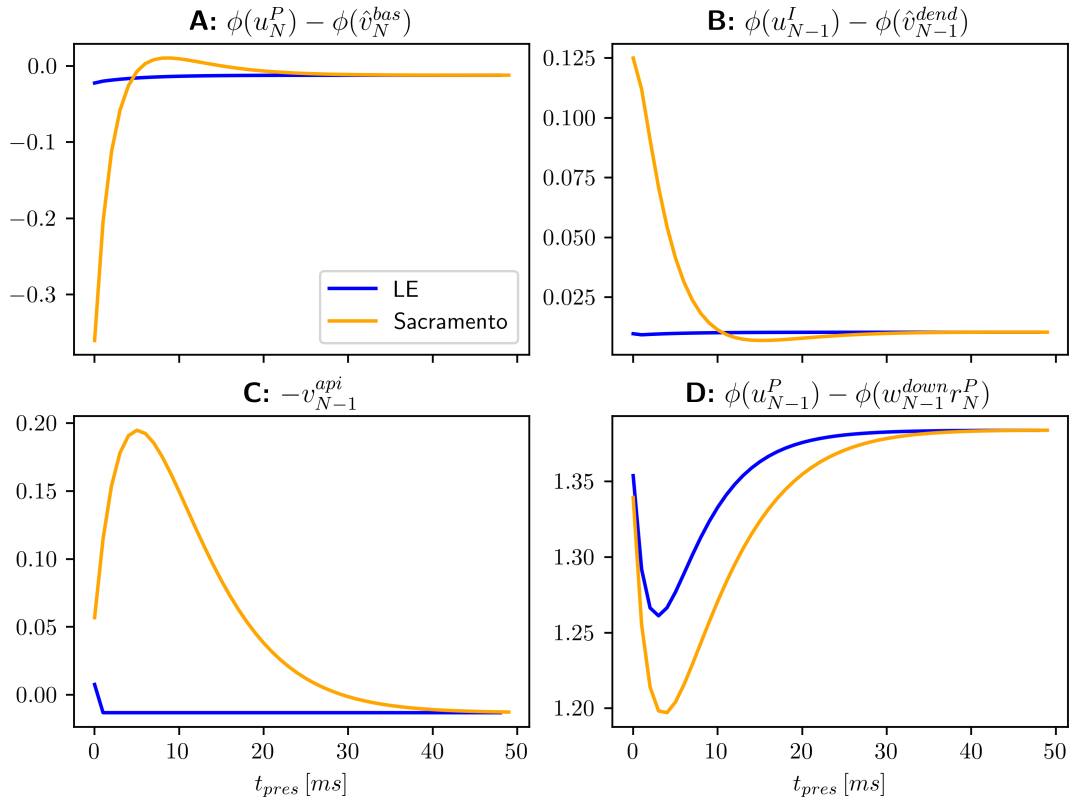

Fig. 2.3: Effects of Latent equilibrium dynamics on the dendritic error terms from Equations 2.13-2.16. Depicted are error terms for individual spiking neurons in a network with one hidden layer (N=3). The network was fully trained on the Bars dataset (cf. Section 3.2), so errors should ideally relax to zero. Note, that this does not happen here due to the fluctuations inherent to the spiking network variant which was employed. In the original dendritic error network (orange), dendritic errors exhibit longer and more intense deviations, while errors in an identical LE network (blue) relax much sooner. **A:** Basal dendritic error for a pyramidal neuron at the output layer. **B:** Dendritic error for a hidden layer Interneuron. **C:** Proximal apical error for a hidden layer Pyramidal neuron. **D:** Distal apical error for the same pyramidal neuron. Note that this error term does not converge close to zero for either network after the relaxation period, an issue that will be discussed in Section 5.5.

of dendritic errors were updated to allow for compartment-specific plasticity rules. Interneurons were chosen to be modeled as pyramidal neurons with slightly updated parameters and apical conductance $g^{api} = 0$. Since membrane dynamics of both neurons follow the same principles and additional compartments have minor impact on performance, this was deemed sufficient.

After facing some setbacks when attempting to train the first spiking variant of the network, the decision was made to also implement a rate-based variant of the neuron in NEST. While the additional effort required for another implementation might be questionable, this model turned out to be indispensible. It enabled the identification of both errors in the model, as well as training mechanisms and parameters that required changes to enable spike-compatible learning. The rate version in NEST additionally served to distinguish discrepancies that are due to the novel simulation backend from those that were introduced by the spike-based communication

scheme.

Following NEST convention, the spiking and rate-based neuron models were named `pp_cond_exp_mc_pyr`[8] and `rate_neuron_pyr` respectively. Furthermore, the `pyr_synapse` class was defined for spike events, and implements the event-based variant of the Urbanczik-Senn plasticity described in Section 2.7. The `pyr_synapse_rate` model on the other hand transmits rate events and updates its weight according to the original plasticity rule.

Simulations were managed using the python API `PyNEST` (Eppler et al., 2009), which is much more convenient than the SLI interface that lies at the core of NEST. An additional advantage of using this language is, that the LE network is also implemented in python. Thus, by including a slightly modified version of that code in my project, it was possible to unify all three variants in a single network class and accompanying interface. This allowed for exact alignment of network stimulation and readout and enabled in-depth comparative analyses. In some of the upcoming Results, three variants of the same network architecture will therefore be compared; The modified python implementation from (Haider et al., 2021) is termed `NumPy` based on the framework that is used to compute neuron dynamics and synaptic plasticity through matrix multiplication. The two NEST variants will be referred to as *NEST spiking* and *NEST rate*.

2.9.1 Neuron model Adaptations

The neuron model from Stapmanns et al. (2021) was modified in some ways in order to match the pyramidal neuron implementation more closely. Both the inclusion of nonzero reversal potentials and the flow of currents from the soma to the dendrites were omitted in my model. Furthermore, the present network requires synapses to be able change the sign of their weight at runtime, which is not permitted in the original synapse model. For this reason, the strict separation of excitatory and inhibitory synapses had to be removed from the synapse model. In order to compare the different implementations exactly, the ODE solver with variable stepsize was replaced with Euler integrations[9] with step size Δt. For the spiking neuron model, dendritic compartments are modeled with leaky membrane dynamics in contrast to the rate variant. The choice of dendritic leakage conductance $g_l^{dend} = \Delta t = 0.1$ is motivated in Section 5.4.

A major issue of the spiking network is the fact that under the default parametrization, spikes are too infrequent for the network to accurately compute the dendritic error terms. Initial experiments showed that the network is rather sensitive to changes in parametrization, which meant that it was desirable to change as few existing parameters as possible. Therefore, a novel parameter ψ was introduced. In a spiking neuron i, the probability of eliciting a spike is linearly increased by this factor ($r_i = \psi\phi(u_i)$). Likewise, all synaptic weights W in

[8] Despite being somewhat cryptic, the name does actually make sense, as it describes some key features of the model: It is a **point process** for **cond**uctance based synapses and has an **exp**onentially decaying membrane in **multiple compartments**.

[9] This change initially served debugging purposes, but turned out to have no negative effect on performance and was therefore kept

a spiking network are attenuated by the same factor ($W \leftarrow \frac{W}{\psi}$). These changes cancel each other out, as an increased value for ψ elicits no change in absolute compartment voltages of a network. Instead, it serves to stabilize these voltages over time, which drastically improves learning performance. One mechanism in which this parameter also needs to be considered is the plasticity rule. Weight changes are affected by ψ in three distinct ways: Since ψ is linearly scales the activation (spiking or hypothetical), it also increases dendritic error linearly, as it does the presynaptic activation. Additionally, since the frequency of weight changes is determined by the presynaptic spike rate, ψ increases the strength of plasticity three times. As these influences are multiplicative, learning rates are attenuated by $\eta \leftarrow \frac{\eta}{\psi^3}$. The exception to this are the weights from interneurons to pyramidal neurons, as these do not depend on dendritic predictions, but on absolute dendritic voltage. Hence, in this case $\eta^{pi} \leftarrow \frac{\eta^{pi}}{\psi^2}$. On close investigation of the spiking neuron model, one can observe that for $\psi \to \infty$, it approximates the rate-based implementation exactly at the steady state. Unsurprisingly therefore, increasing ψ caused the spiking network to learn successfully with fewer samples and to a lower test loss. Yet, the argument against increasing ψ is twofold: Initial experiments showed that with $\psi \in [0.1, 0.5]$, pyramidal and interneurons exhibit spike frequencies in biologically plausible range of less than $55Hz$ Kawaguchi (2001); Eyal et al. (2018) **TODO: reevaluate!**. Additionally, each transmitted `SpikeEvent` is computationally costly, which increases training time (cf. Fig. 3.6) and further makes high spike frequencies undesirable. As a middle ground, $\psi = 100$ proved useful during initial tests and will be assumed the default from here on out. Note that this parametrization was chosen primarily with efficiency in mind, and makes no claim towards biologically plausible spike frequency.

With these adaptations, the network was able to perform supervised learning with spiking neurons, as will be discussed in the upcoming sections.

2.10 Error metrics and nomenclature

In this thesis, the word 'error' is used frequently, which might understandably lead to confusion. While stylistically questionable, this choice was made deliberately to conform to the main underlying works (Urbanczik and Senn, 2014; Sacramento et al., 2018; Whittington and Bogacz, 2019; Haider et al., 2021). This section will provide a brief disambiguation. Firstly, there are four error metrics describing the network's deviation from the self-predicting state: *Feedforward weight error, Feedback weight error, Apical error and interneuron error*. These were introduced in Section 2.3. Close observation of the network reveals that pairs of them are closely related: Feedforward weight error drives interneuron error, and feedback weight error drives apical error (so long as interneuron error is minimal). An analytical upgrade to this model might include a way for unifying these pairs of metrics. Furthermore, three terms require elaboration:

Dendritic error: Any value which drives the Dendritic plasticity rules. Classically, this refers to a failure of a dendrite to predict somatic activity (Urbanczik and Senn, 2014). In this context, due to the changes to the plasticity rule, it may also refer to absolute voltage of pyramidal neuron apical compartments (i.e. Apical error).

Train error: Failure rate of a network to correctly classify inputs during testing. In the upcoming simulations, all targets are encoded with one-hot vectors. Thus, accuracy is defined as:

$$accuracy = \frac{1}{N}\sum_{i=1}^{N} \delta\left(argmax(y_i^{target}),\ argmax(y_i^{pred})\right)$$

for a test run over N samples, with δ being the Kronecker delta function. Train error is defined as inverse accuracy.

Loss: Unless specified otherwise, train- and test loss are are computed through MSE between predicted and target output:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(y_i^{target} - y_i^{pred}\right)^2$$

Due to the network's relaxation period, y^{pred} can not be accurately computed instantaneously[10]. Instead, the network needs to be presented with the stimulus for a given time t_{pres}. Particularly for the SNN, as well as networks injected with noise, output fluctuates. therefore, y^{pred} is an average over recorded somatic potentials for each output neuron. This recording typically starts after roughly 70% of t_{pres}.

[10]Sacramento et al. actually do exactly this. They compute y^{pred} without neuron dynamics, only from the input, activation function ϕ, and feedforward weights. This approach makes the assumption that the network is permanently in a perfect self-predicting state, in which feedback connections do not change the output. Particularly for the spiking variant, this assumption is likely erroneous, leading to artificially inflated performance. Hence, all tests are performed by fully simulating networks with disabled plasticity.

Chapter 3

Results

The following results are exploratory in nature, and After some poor initial results the focus was laid on proving that the network can perform at all, rather than fine-tuning hyperparameters towards optimal performance. This decision was in part motivated by a priorization of gaining neuroscientific insights over proving high performance. Yet it should be noted, that training the network is computationally quite costly (c.f. Section 3.8), which turned parameter studies into a time-consuming process. The network is furthermore fairly sensitive to changes in parametrization, hence many early experiments in this direction immediately caused a failure to learn. The search for default parameters itself took some effort, as a certain heterogeneity exists in the two existing implementations (Sacramento et al., 2018; Haider et al., 2021), both in hyperparameters as in the simulation environment. This model includes properties of both variants, while relying more strongly on the Latent Equilibrium implementation. Unless stated otherwise, neurons employ prospective activation functions in all simulations. So far, no drawbacks to this mechanism have presented themseleves, and learning speed can be increased drastically compared to the original implementation. The full default parametrization is shown in Table 5.1.

Since it was anticipated that the spiking implementation would perform worse than the rate-based variant, the first goal was to measure how big this difference in performance is. Furthermore, a relevant question was, to what degree the synaptic delays enforced by NEST would influence performance of the rate model. These questions will be answered in the part of the upcoming sections.

3.1 The self-predicting state

As a first comparison between the three implmementations, the pre-training towards a self-predicting state (cf. Sacramento et al. (2018)[Fig. S1]) was performed. For this experiment, no target signal is provided at the output layer, and the network is tasked with learning to self-predict top-down input. The network is initialized with fully random weights and stimulated

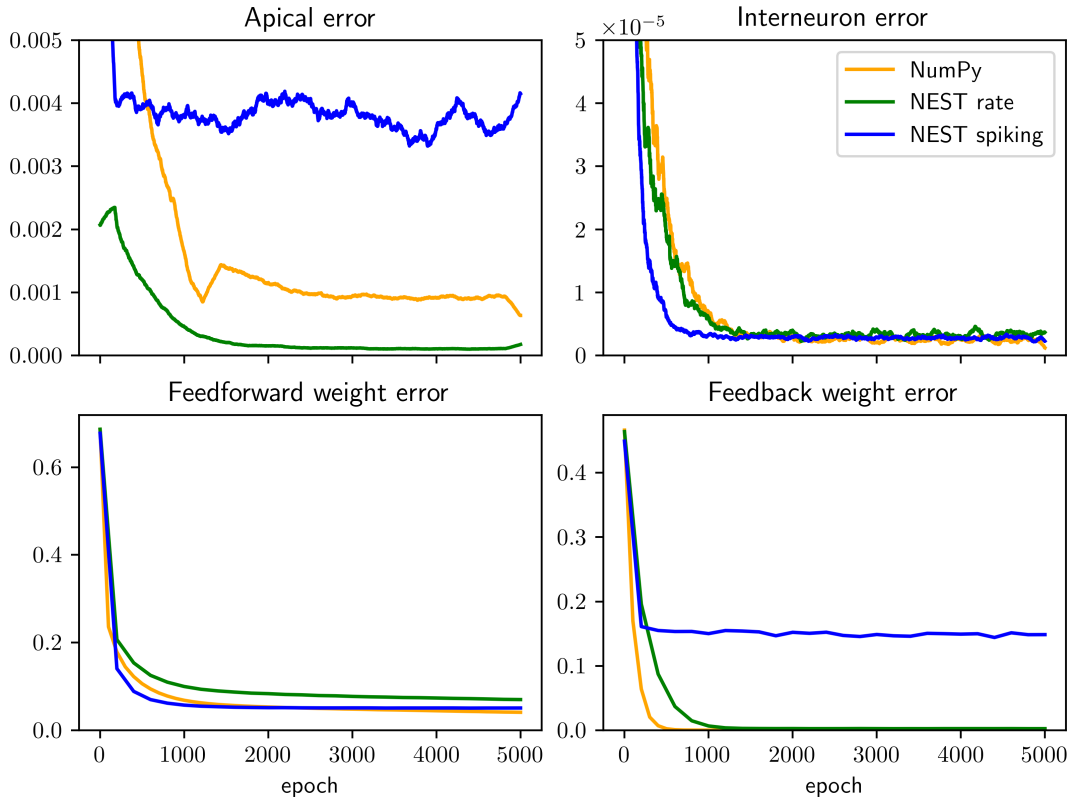

Fig. 3.1: Different network types learn to predict self-generated activity in superficial layers. All networks were initialized with the same random weights for dimensions $[5, 8, 3]$, and stimulated with 5000 samples of random input for 100*ms* each. As described in Sacramento et al. (2018), during this phase only Pyramidal-Interneuron and Interneuron-Pyramidal weights are plastic ($\eta^{pi} = 0.05, \eta^{ip} = 0.02375, \eta^{up} = \eta^{down} = 0$). For spiking neurons, stability was increased by setting $\psi = 150$.**TODO: redo figure with less smoothing and tighter yscale**

with random inputs from a uniform distribution between 0 and 1. A comparison of the four error metrics between networks is shown in Fig. 3.1.

Both rate neuron implementations were able to reach comparable values for all error metrics after roughly the same time. The exact values that errors converge on differs slightly between implementations, with no implementation being clearly superior. This is an important result for upcoming experiments, as it indicates that the simulation environment developed for the NEST version is an adequate replica of the original framework.

For the spiking variant, Interneuron- and its corresponding feedfworward weight error are comparable to the other implementations. In fact these metrics appear to converge slightly faster to comparable values. The primary limitation of this version are the apical error and the closely correlated feedback weight error. After appearing to converge very quickly, the two metrics stagnate at very high levels. This behaviour can at least in part be attributed to individual spikes causing fluctuations in apical potentials. This was confirmed by repeating the experiment with $\psi = 1500$, which alleviated the issue somewhat (results not shown). Yet, error

values were still inferior to the rate models, and this change came at the cost of substantially increased training time. Therefore, this approach was not pursued much further. A different possible solution is, to increase the membrane capacitance of the apical compartment in order to smoothe out the error induced by individual spikes. This will be discussed in Section 3.3.

In most simulations in the literature, the network is initialized to an ideal self-predicting state. Furthermore feedback weights are often non-plastic ($\eta^{pi} = \eta^{down} = 0$). Therefore, a failure to reduce these errors further might not be a substantial issue. For the time being, showing that the network approaches a self-predicting state was deemed a sufficient result.

3.2 Presentation times and latent equilibrium

In order to validate the performance of the NEST implementations on a learning task, the parameter study from (Haider et al., 2021)[Fig. 3] was replicated. In this experiment, the network is trained different stimulus presentation times $t_{pres} \in \{0.3,\ 500\}ms$. Performance of the original Dendritic error network is compared to the improved model which employs Latent equilibrium. Due to the costly computation of the network under such long t_{pres}, a simple artificial classification dataset was developed. The *Bars-dataset* is defined for 3×3 input- and 3 output neurons. it consists of three horizontal, three vertical, and two diagonal bars in the 3×3 grid, which are to be encoded in a 'one-hot-vector' at the output layer. In the experiment, networks of $9 - 30 - 3$ pyramidal neurons per layer were trained for 1000 Epochs of 24 samples each. Networks were initialized to the self-predicting state and only feedforward $Pyr \to Pyr$ and $Pyr \to Intn$ synapses were plastic. Learning rates scaled inversely with presentation times: $\eta_0^{ip} = \frac{0.2}{t_{pres}}, \eta_0^{up} = \frac{0.5}{t_{pres}}, \eta_1^{up} = \frac{0.1}{t_{pres}}$. The results for the NEST network using spiking neurons with default parameters **TODO: elaborate on this** are shown in Fig. 3.2, while the results for NumPy and Rate NEST variants are depicted in Supplementary Figures S1 and S2, respectively.

For the original dendritic error model, performance in all three implementations is close to being identical. This an important finding as it answers two open questions: Changes made for a NEST-compatible implementation were adequate and result in equal learning performance between rate-based implementations. Learning behaviour of the spiking model is the same, confirming the hypothesis that the spike-based dendritic plasticity model is capable of more complex credit assignment tasks than previously shown. In this regard, the implementation can be considered a success.

The results for the Latent equilibrium experiments are somewhat more interesting. For very long t_{pres}, both rate implementations behave the same. yet the NEST implementation requires considerably more epochs for training, as t_{pres} is reduced. In the limit, the NEST implementation was expected to break due to the enforced synaptic delay. The NumPy variant computes a full forward pass of the network during the first simulation step, as all layers are processed in sequence. Only feedback signals from pyramidal neurons are delayed by one

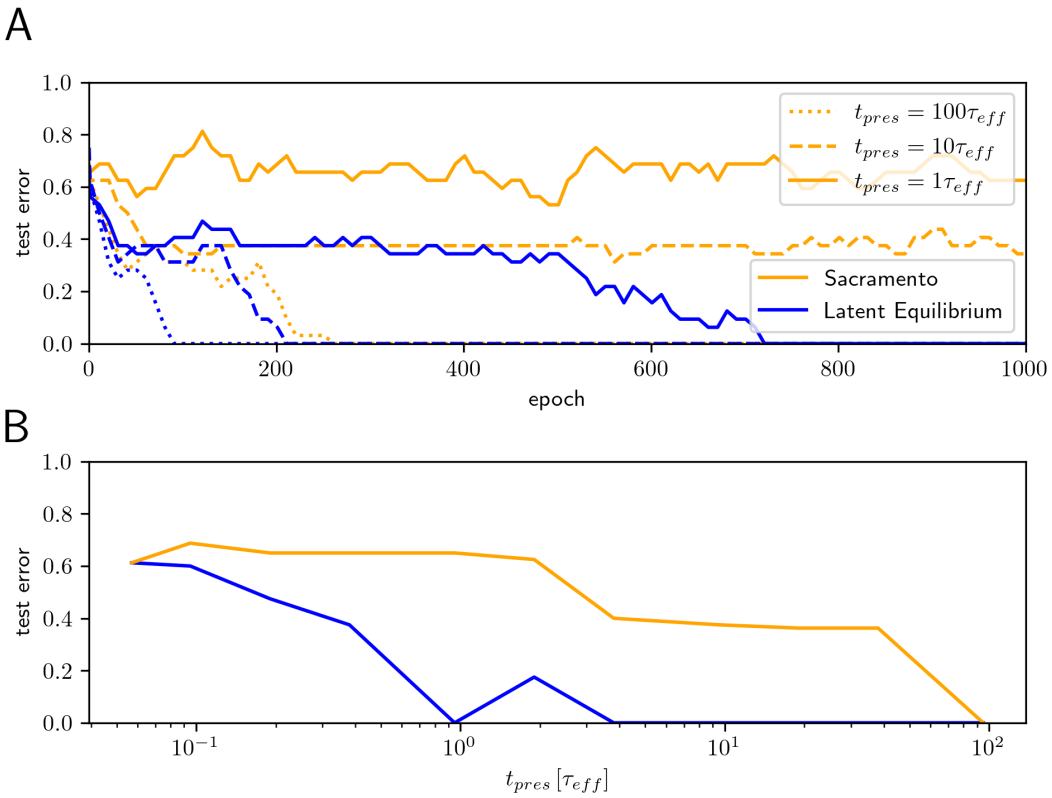

Fig. 3.2: Replication of Fig. 3 from Haider et al. (2021) using networks of spiking neurons in the NEST simulator. **A:** Comparison between Original pyramidal microcircuit network by Sacramento et al. (2018) and Latent equilibrium variant from Haider et al. (2021). Shown is the training of networks with 9-30-3 neurons on the Bars-dataset from with three different stimulus presentation times. **B:** Test performance after 1000 Epochs as a function of stimulus presentation time.

timestep, in this simulation backend. In NEST, all connections have a minimum synaptic delay of Δt. Therefore, for very short presentation times the NEST network can not be expected to perform well. It remains an open question whether this feature alone explains the gradual decrease in performance, or if there is an undiscovered error within the novel neuron models or simulation environment. Such short stimulus presentation times - while exceptionally beneficial to computational efficiency - are themselves questionable in terms of biological plausibility, as they are much lower than pyramidal neuron time constants (McCormick et al., 1985). Thus, no attempts were made to improve performance for very low t_{pres}.

The spiking variant proved similarly sensitive to presentation times as the other NEST variant. While obtaining similar performance, it required - at best - twice as many stimulus presentations as its direct competitor. This is result, while somewhat expected, makes the implementation **TODO:**

3.3 Apical compartment capacitance

Alternatively, increasing the membrane capacitance of the apical compartment also solves the issue as it smoothes out the effect of individual spikes. Yet this solution also increases the relaxation period of the entire network, requiring a highly undesirable increase in t_{pres} for successful learning. Since weight errors converge to similar values as the rate-based implementations, an increased absolute apical compartment voltage was deemed tolerable.

3.4 Separation of synaptic polarity

TODO: investigate dales law (Barranca et al., 2022)

A key limitation of the present network model is the requirement that all synapses must be able to assume both positive and negative polarities. When restricting any synaptic population in the network to just one polarity, the network is unable to reach the self-predicting state **TODO: expand?**. Thus, activity in any neuron must be able to have both excitatory and inhibitory postsynaptic effects facilitated by appropriate synaptic weights. This requirement is at odds with biology, which dictates a singular synaptic polarity for all outgoing connections of a neuron, determined by neuron type and its corresponding neurotransmitter **TODO: cite**.

To investigate to what degree the plasticity rule can deal with this constraint, an experiment was conducted: A population of pyramidal neurons A was connected to another population C with plastic synapses that were constrained to positive weights. In order to facilitate the required depression, A was also connected to a population of inhibitory interneurons B through excitatory synapses with random and non-plastic weights. The interneurons in turn were connected to C through plastic, inhibitory connections. All incoming synapses at C targeted the same dendritic compartment. When inducing a dendritic error in that compartment, all plastic synapses in the network collaborated in order to minimize that error. When injecting a positive basal error for example, the inhibitory weights ($C \to B$) decayed, while excitatory synaptic weights ($A \to B$) increased. Flipping the sign of that error injection had the opposite effect on weights, and likewise cancelled the artificial error. This shows that a separation of synaptic polarity does not interfere with the principles of the Urbanczik-Senn plasticity when depression is facilitated by interneurons.

Yet, as criticised previously (Whittington and Bogacz, 2019), the one-to-one connections between A and B are untypical for biological neural networks **TODO: cite**. Hence, a second experiment was performed, in which A and B were fully connected through static synapses with random positive weights. This decrease in specificity of the connections did not hinder the error-correcting learning, as shown in Fig. 3.3.

These results are useful, as they enable a biologically plausible way for excitatory long-range pyramidal projections to connect to pyramidal neurons in another layer of the network (i.e. in a different part of the cortex). The steps required to facilitate this type of network are rather

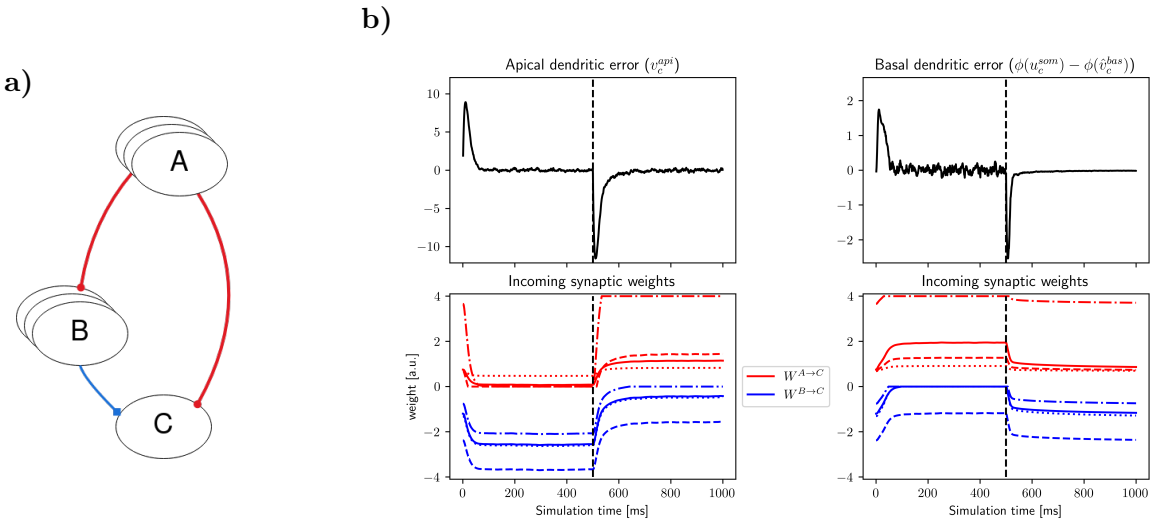

Fig. 3.3: Dendritic error minimization under biological constraints on synaptic polarity and network connectivity. **a)** Network architecture. An excitatory population A connects to a dendrite of Neuron C both directly and through inhibitory interneuron population B. Only synapses $A \to C$ and $B \to C$ are plastic through dendritic error rules. Populations A and B are fully connected with random weights. **b)** *Left:* All plastic synapses arrive at apical dendrites and evolve according to Equation 2.15. *Right:* Identical network setup, plasticity for synapses at basal dendrites (Equations 2.13, 2.14). *Top:* Dendritic error of a single target neuron. Errors of opposite signs are induced at 0 and 500ms (vertical dashed line). *Bottom:* Synaptic weights of incoming connections. All initial synaptic weights and input neuron activations were drawn from uniform distributions.

simple; A pyramidal neuron projection could enter a distant cortical area and spread its axonal tree **phrasing**within a layer that contains both pyramidal neuron dendrites and interneurons. If these interneurons themselves connect to the local pyramidal population, Dendritic errors with arbitrary signs and magnitudes could be effectively minimized.

While error minimization is a funcamental feature of this network, it does not necessarily imply that synaptic credit assignment is successful aswell. Numerous weight configurations are concievable which could silence dendritic errors, but likely only a small subset of them is capable of transmitting useful information. To prove that this nonspecific connectivity is compatible with learning of complex tasks, it was introduced into the dendritic error network. The connection between Interneurons and Pyramidal neuron apical dendrites was chosen for the first test, as the employed plasticity rule had proven most resilient to parameter imperfections previously. A network of rate neurons was initialized with self-predicting weights as in Section 3.2. The Weights w^{pi} were redrawn and restricted to postive values, and a secondary inhibitory interneuron population was created and fully connected to both populations as described in Fig. 3.3.

3.5 In search of plausible spike frequencies

TODO: expand

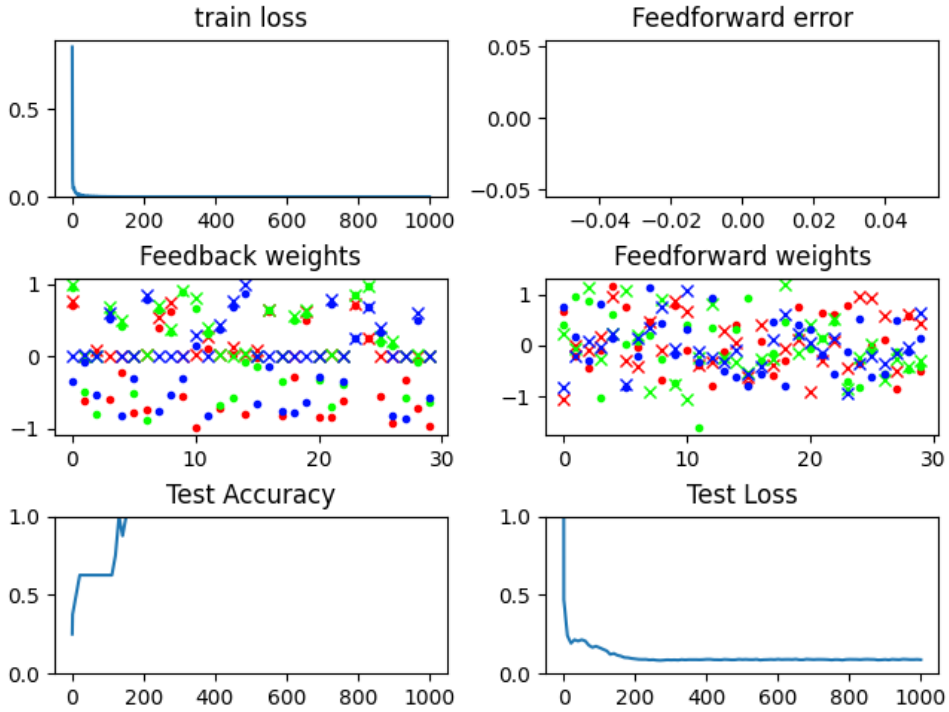

Fig. 3.4: Training progress of a network of rate neurons in NEST, in which hidden layer connections between interneurons and pyramidal neurons are unipolar and nonspecific.

3.6 Resilience to imperfect connectivity

3.7 direct feedback connections to interneurons

Vaughn and Haas (2022); Mancilla et al. (2007)

3.8 Performance of the different implementations

As stated in Haider et al. (2021), simulating the present network with many neurons or more than one hidden layer quickly becomes unfeasible when simulating the full leaky dynamics. To investigate how network size affects simulation time, all three implementations created for this project were trained on the bars dataset for a single epoch with different network sizes for a single epoch, in order to assess efficiency.

The result of this comparison is shown in Fig. 3.6. The NumPy network is slow at baseline, which is likely explained by the fact that it is the only variant which is running on a single thread. This is due to a limitation of NumPy, and could likely be improved greatly by using batched matrix multiplications, as are provided for example by `PyTorch`[1].

[1]It is also possible, that the network code surrounding the NumPy computations is less efficient than the one for the NEST network. As this implementation was needed primarily to prove that neuron dynamics and synaptic updates were ported correctly to NEST, efficiency was a minor concern here and this was not investigated further.

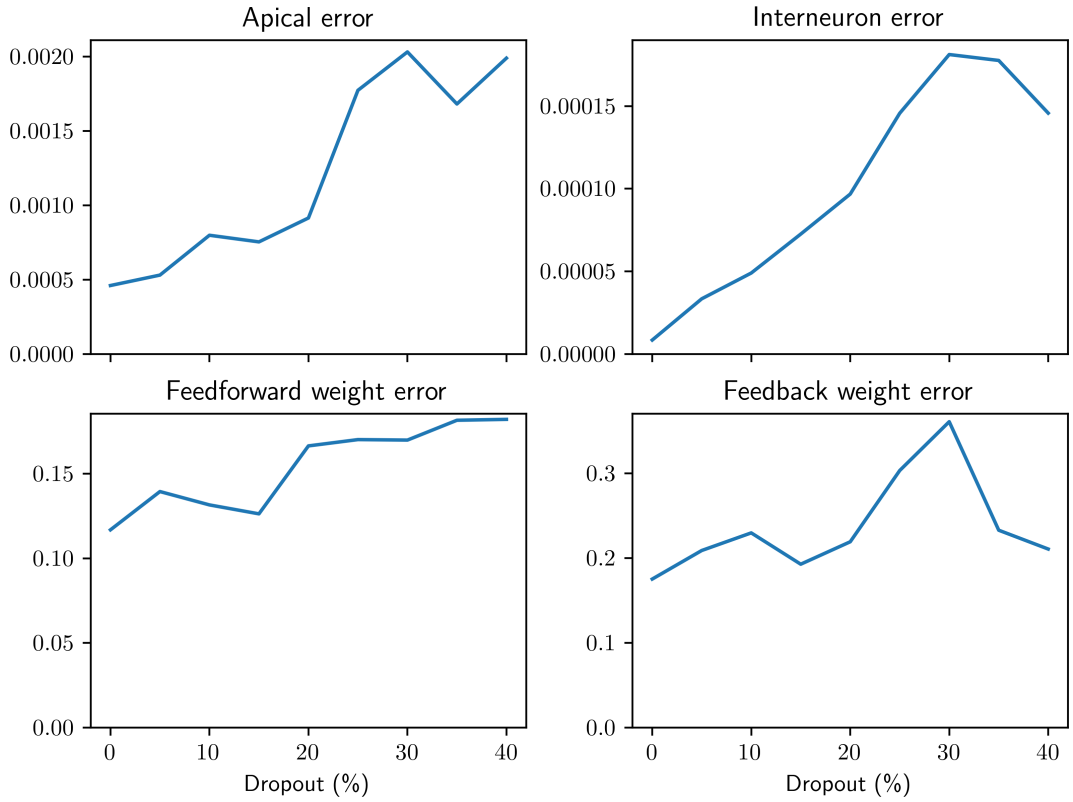

Fig. 3.5: Error terms after training a network with dimensions [8, 8, 8] towards the self predicting states, with different percentages of synaptic connections randomly removed. To avoid completely separated layers, synapse deletion was performed per synaptic population, and the percentage is therefore only an approximation. Even with only 60% of synaptic connections present, the network architecture still achieves competetive values for all four error metrics. For the weight errors, which is calculated with MSE over two matrices, missing synapses were set to 0. This choice was made under the assumption that a missing connection in an ideal self-predicting network would be matched by a zero-weight - or likewise absent - synapse. Experiments were performed with the rate-based network in NEST, each network was trained fro 2000 epochs of 50ms each.

Notably, this variant exhibits very little slowdown in response to an increase in network size. My assumption is, that the vectorization of synaptic updates on a single thread scales up better than the communication between threads that is required by most events in the NEST simulations. The NEST implementation using rate neurons performed best in terms of speed across the board. This result was slightly surprising, as the demand on the communication interface between threads is very high, since all neurons transmit an event to each of their postsynaptic targets at every time step.

Finally, the novel spiking variant of this model performed substantially worse than anticipated. Particularly in comparison to the rate implementation, I initially expected substantial performance improvements. The Difference between the two was even greater when simulating on an office-grade processor (Benchmark was also run on an *Intel Core i5-9300H* at 2.40GHz, results not shown). Three hints about the comparatively poor performance can be deduced:

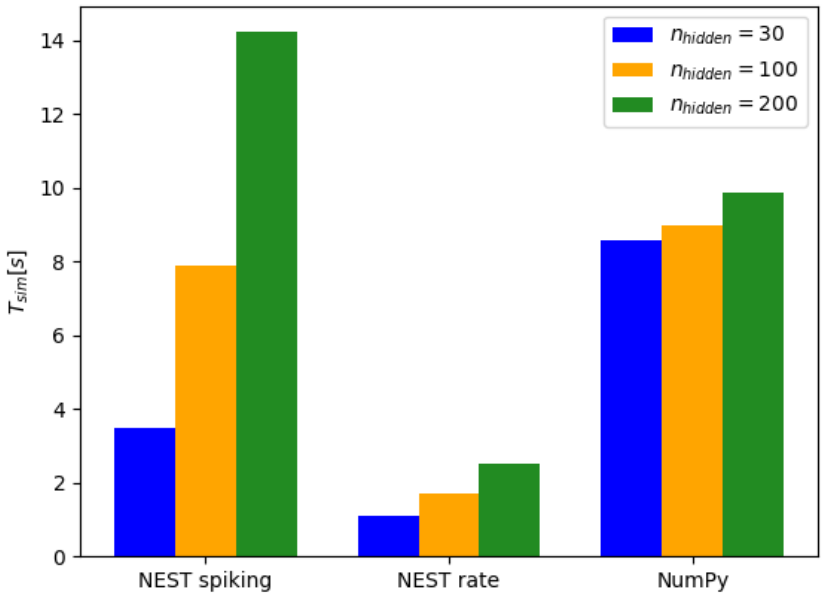

Fig. 3.6: Benchmark of the three different implementations using a network of $[9, n_{hidden}, 3]$ neurons per layer. $n_{hidden} = 30$ was chosen as a baseline, as it is the default throughout all simulations on the Bars dataset. Networks were instantiated with the same synaptic weights and trained for a single epoch of 5 stimulus presentations of $100ms$ each. Simulations were performed on an *AMD Ryzen Threadripper 2990WX* using 8 cores for the NEST simulations at up to $3.0GHz$.

For one, both the rate and the spiking neuron model employ almost identical neuron models, with minor changes to parametrization and output generation. Thus, updates to the neuron state are unlikely to be responsible for the worse performance. Secondly, the number of Events transmitted between neurons is much lower for the SNN compared to

the *relative* performance decrease when increasing the number neurons by the same amout is much greater for the spiking network. Thus, the most likely cause of slowdown are the updates at the synapses. This is supported by the fact, that the number of synapses increases much faster for this kind of network than the number of neurons. For the given network of $n_x = 9$ input neurons, $n_y = 3$ output neurons and n_h neurons in the hidden layer l, the number of total synapses in the network is given by

$$n_{synapses} = |w_l^{up}| + |w_l^{pi}| + |w_l^{ip}| + |w_l^{down}| + |w_y^{up}| \tag{3.1}$$
$$= n_h n_x + n_h n_y + n_y n_h + n_h n_y + n_y n_h \tag{3.2}$$
$$= n_h(n_x + n_y^4) \tag{3.3}$$

with $|w|$ of a weight matrix w in this case denoting the total number of elements in that

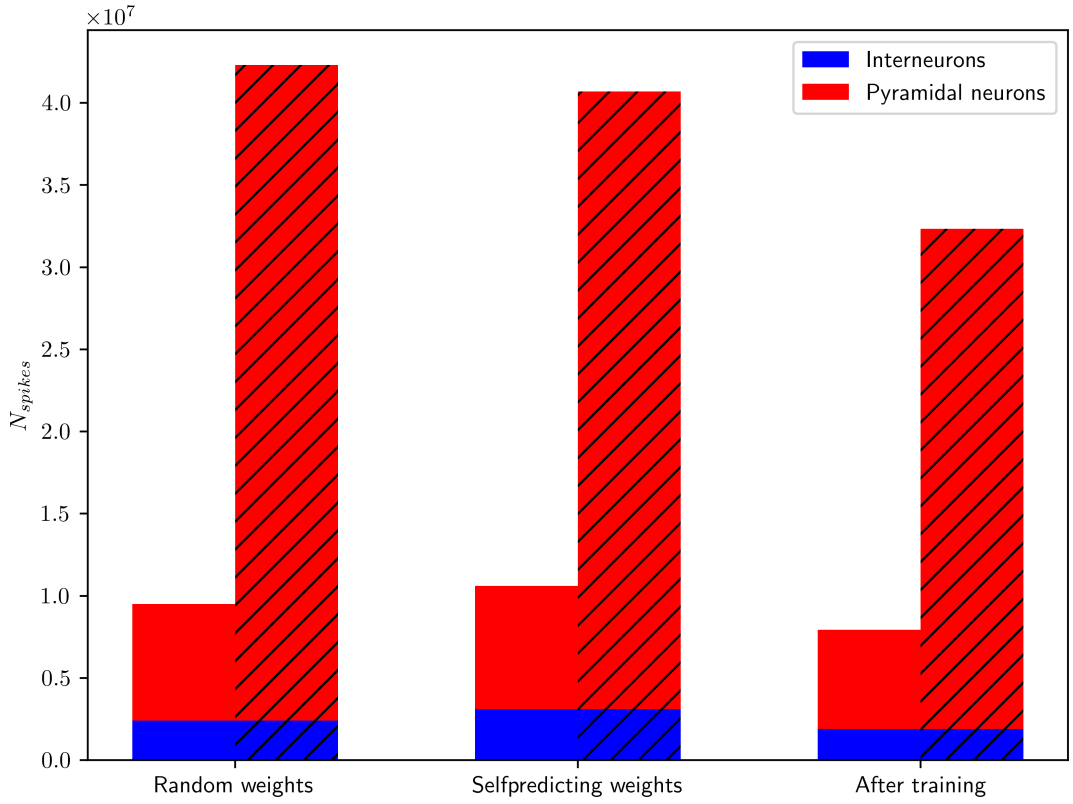

Fig. 3.7: Comparison of network response to stimuli from the Bars dataset.

matrix **Is there a more conventional notation?** . Thus, the number of synapses in a network grows much faster than the number of total neurons when increasing the size of the hidden layer.

availis given by the stark increase in

3.9 Response to unpredicted stimuli

The activity of many cortical neurons increases when a brain is presented with unpredicted stimuly that regard these neurons **TODO: check out Whittington and Bogacz (2019) refs 50-54**. This property is prominently replicated by predictive coding networks, since activation of error nodes is a function of local prediction errors. Prediction errors in the present model on the other hand are encoded in both positive and negative potentials of apical dendrites. Hence, the

Chapter 4

Discussion

4.1 Contribution

4.1.1 criteria for biological plausibility

In which we discuss, to what extent the network conforms to the criteria for biologically plausible learning rules introduced by Whittington and Bogacz (2017):

1. Local computation. A neuron performs computation only on the basis of the inputs it receives from other neurons weighted by the strengths of its synaptic connections.
2. Local plasticity. The amount of synaptic weight modification is dependent on only the activity of the two neurons the synapse connects (and possibly a neuromodulator).
3. Minimal external control. The neurons perform the computation autonomously with as little external control routing information in different ways at different times as possible.
4. Plausible architecture. The connectivity patterns in the model should be consistent with basic constraints of connectivity in neocortex.

4.2 Limitations

Network needs to be reset between stimuli original does not do that, in NEST it's kindof a big deal.

Most experiments require repeats to confirm results across with multiple independent runs.

Efficiency

exposure time and training set still quite large

Neuron model

non-resetting, non-refractory

Network

Oversized networks have a higher tendency of failing on some tasks (results not shown). This issue is not consistent, but plagued some of my experiments with no satisfactory explanation yet found.

When injecting large currents (i.e. in func approx experiments), the network kinda shits itself. Somewhat expected, somewhat annoying

4.3 Future directions

4.3.1 Additional Backprop concerns

from (Marblestone et al., 2016) on one-shot learning

Additionally, the nervous system may have a way of quickly storing and replaying sequences of events. This would allow the brain to move an item from episodic memory into a long- term memory stored in the weights of a cortical network (Ji and Wilson, 2007), by replaying the memory over and over. This solution effectively uses many iterations of weight updating to fully learn a single item, even if one has only been exposed to it once. Alternatively, the brain could rapidly store an episodic memory and then retrieve it later without the need to perform slow gradient updates, which has proven to be useful for fast reinforcement learning in scenarios with limited available data (Blundell et al., 2016)

Where do targets come from? Bengio et al. (2015)

4.4 Should it be considered pre-training?

TODO: someone said this network needs pre-training and that made me sad :(

4.5 Relation to energy minimization

4.6 Correspondence of the final model to cortical circuitry

Cortical neurons depend on neuromodulation too Roelfsema and Holtmaat (2018)

Completely unclear whether cortical circuits perform supervised learning (Magee and Grienberger, 2020)

This would probs be super efficient on neuromorphics!

I am not going to try plasticity with spike-spike or spike-rate dendritic errors

Training on ImageNet

Making this shit faster

BPTT/time-continuous inputs

The dendritic error rule at the core of Urbanczik-Senn looks absurdly similar to superspike (Zenke and Ganguli, 2018). Someone should look into that!

Different configurations for which synapses are plastic should be elaborated on.

4.6.1 improvements to the neuron models

TODO: talk about ALIF neurons (Gerstner and Naud, 2009) also reffed in e-prop(Bellec et al., 2019, 2020). look for sources

pyr and intn are wayy to similar

LIF, membrane reset and t_{ref}

reward-modulated urbanczik-senn plasticity?

wtf is shunting inhibition?

Consider neurogenesis deeper. Any network that is plausible must be able to develop in a plausible fashion. Investigating how the cortex develops might hold insights into plausible connectivity schemes. This does not necessarily compete or conflict with looking at connectivity of developed brains

improve prospective activity with regard to spike creation

le does a lot, particularly at hidden layers. Yet response time under spiking paradigms is still lackluster. In particular, prospective activation does next to nothing for these very low input time constants. SNN performs best when τ_x is greater than Δ_t by roughly x10.

4.7 Conclusion

Chapter 5

Appendix

5.1 Somato-dendritic coupling

Urbanczik and Senn (2014) discuss a possible extension to their neuron- and plasticity model, in which the dendro-somatic coupling transmits voltages in both directions. They show that the plasticity rule requires only minor adaptations for successful learning under this paradigm. Yet, as described by passive cable theory, the flow between neuronal compartments is dictated by their respective membrane capacitances. These are calculated from their membrane areas, which vastly differ in the case of pyramidal neurons. **TODO: find a nice citation for this**

15,006 458

will not be considered here. The motivation is, that dendritic membrane area is

5.2 Default parameters

5.3 Integration of the spike-based Urbanczik-Senn plasticity

Starting with the complete Integral from $t = 0$.

Name	Description	Default
Simulation		
n_epochs	Number of training iterations	1000
delta_t	Euler integration step in [ms]	0.1
t_pres	Stimulus presentation time during training [ms]	50
out_lag	Lag before recording output during testing [ms]	35
dims	Network dimensions, i.e. pyramidal neurons per layer	[9, 30, 3]
threads	Number of threads for parallel processing	8
test_interval	Test the network every N epochs	10
record_interval	Interval for storing membrane potentials [ms]	1
init_self_pred	Flag to initialize weights to self-predicting state	True
noise	Flag to apply noise to membrane potentials	False
sigma	Standard deviation for membrane potential noise	0.3
mode	Which dataset to train on. Choice between (bars, mnist, self-pred, teacher)	bars
store_errors	Flag to compute and store apical and interneuron errors during training	False
network_type	Choice between (numpy, snest, rnest)	snest
tau_x	Network input filtering time constant [ms]	0.1
reset	Reset method between simulations (0:no reset, 1=soft reset, 2=hard reset)	2
Neurons		
latent_equilibrium	Flag for whether to use prospective transfer functions	True
g_l	Somatic leakage conductance [nS]	0.03
g_a	Apical compartment coupling conductance [nS]	0.06
g_d	Basal compartment coupling conductance [nS]	0.1
g_som	Output neuron nudging conductance [nS]	0.06
g_l_eff	Effective leakage conductance [nS]	g_l+g_d+g_a
g_lk_dnd	Dendritic leakage [nS]	delta_t
t_ref	Refractory period [ms]	0.0
C_m_som	Somatic compartment membrane capacitance [pF]	1.0
C_m_bas	Basal compartment membrane capacitance [pF]	1.0
C_m_api	Apical compartment membrane capacitance [pF]	1.0
gamma	Linearly scales activation function ϕ	1.0
beta	Exponentially scales activation function ϕ	1.0
theta	Shifts activation function ϕ	0.0
Synapses		
wmin_init	Min. initial synaptic weight	-1.0
wmax_init	Max. initial synaptic weight	1.0
Wmin	Min. allowed synaptic weight	-4.0
Wmax	Max. allowed synaptic weight	4.0
tau_delta	Weight change filter time constant (NEST only) [ms]	1.0
p_conn	Connection probability between populations	1.0
eta_ip	Learning rate for *pyr* $\rightarrow$ *intn* synapses	0.004
eta_pi	Learning rate for *intn* $\rightarrow$ *pyr* synapses	0.01
eta_up	Learning rates for feedforwarsd *pyr* $\rightarrow$ *pyr* synapses	[0.01, 0.003]
eta_down	Learning rate for feedback *pyr* $\rightarrow$ *pyr* synapses	0.0

Table 5.1: Default parameters used in all simulations.

$$\dot{W}_{ij}(t) = \eta(\phi(u_i) - \phi(\alpha v_i^{basal}(t)))\phi(u_j) \tag{5.1}$$

$$\Delta W_{ij}(t,T) = \int_t^T dt' \ \eta \ (\phi(u_i^{t'}) - \phi(\widehat{v_i^{t'}})) \ \phi(u_j^{t'}) \tag{5.2}$$

$$\Delta W_{ij}(t,T) = \eta \int_t^T dt' \ (\phi(u_i^{t'}) - \phi(\widehat{v_i^{t'}})) \ \phi(u_j^{t'}) \tag{5.3}$$

$$V_i^* = \phi(u_i^{t'}) - \phi(\widehat{v_i^{t'}}) \tag{5.4}$$

$$s_j^* = \kappa_s * s_j \tag{5.5}$$

$$\Delta W_{ij}(0,t) = \eta \int_0^t dt' \ \int_0^{t'} dt'' \ \kappa(t'-t'')V_i^*(t'')s_j^*(t'') \tag{5.6}$$

$$= \eta \int_0^t dt'' \ \int_{t''}^t dt' \ \kappa(t'-t'')V_i^*(t'')s_j^*(t'') \tag{5.7}$$

$$= \eta \int_0^t dt'' \ \left[\tilde{\kappa}(t-t'') - \tilde{\kappa}(0)\right] V_i^*(t'')s_j^*(t'') \tag{5.8}$$

$$\tag{5.9}$$

With $\tilde{\kappa}$ being the antiderivative of κ:

$$\kappa(t) = \frac{\delta}{\delta t}\tilde{\kappa}(t) \tag{5.10}$$

$$\tilde{\kappa}(t) = -e^{-\frac{t}{\iota_\kappa}} \tag{5.11}$$

$$\tag{5.12}$$

The above can be split up into two separate integrals:
Which implies the identities

$$I_1(t_1, t_2 + \Delta t) = I_1(t_1, t_2) + I_1(t_2, t_2 + \Delta t) \tag{5.13}$$

$$I_2(t_1, t_2 + \Delta t) = e^{-\frac{t_2 - t_1}{\tau_\kappa}} I_2(t_1, t_2) + I_2(t_2, t_2 + \Delta t) \tag{5.14}$$

$$I_2(t_1, t_2 + \Delta t) = -\int_{t_1}^{t_2+\Delta t} dt' \ \tilde{\kappa}(t_2 + \Delta t - t') V_i^*(t') s_j^*(t') \tag{5.15}$$

$$= -\int_{t_1}^{t_2} dt' \ \left[-e^{-\frac{t_2+\Delta t-t'}{\tau_\kappa}}\right] V_i^*(t') s_j^*(t') - \int_{t_2}^{t_2+\Delta t} dt' \ \left[-e^{-\frac{t_2+\Delta t-t'}{\tau_\kappa}}\right] V_i^*(t') s_j^*(t') \tag{5.16}$$

$$= -e^{-\frac{\Delta t}{\tau_\kappa}} \int_{t_1}^{t_2} dt' \ \left[-e^{-\frac{t_2-t'}{\tau_\kappa}}\right] V_i^*(t') s_j^*(t') - \int_{t_2}^{t_2+\Delta t} dt' \ \left[-e^{-\frac{t_2+\Delta t-t'}{\tau_\kappa}}\right] V_i^*(t') s_j^*(t') \tag{5.17}$$

Using this we can rewrite the weight change from t to T as:

$$\Delta W_{ij}(t, T) = \Delta W_{ij}(0, T) - \Delta W_{ij}(0, t) \tag{5.18}$$

$$= \eta[-I_2(0, T) + I_1(0, T) + I_2(0, t) - I_1(0, t)] \tag{5.19}$$

$$= \eta[I_1(t, T) - I_2(t, T) + I_2(0, t)\left(1 - e^{-\frac{T-t}{\tau_\kappa}}\right)] \tag{5.20}$$

The simplified Sacramento et al. (2018) case would be:

$$\frac{dW_{ij}}{dt} = \eta(\phi(u_i) - \phi(\hat{v}_i))\phi(u_j) \tag{5.21}$$

$$\Delta W_{ij}(t, T) = \int_t^T dt' \ \eta \ (\phi(u_i^{t'}) - \phi(\widehat{v_i^{t'}})) \ \phi(u_j^{t'}) \tag{5.22}$$

$$\Delta W_{ij}(t, T) = \eta \int_t^T dt' \ (\phi(u_i^{t'}) - \phi(\widehat{v_i^{t'}})) \ \phi(u_j^{t'}) \tag{5.23}$$

$$V_i^* = \phi(u_i^{t'}) - \phi(\widehat{v_i^{t'}}) \tag{5.24}$$

$$s_j^* = \kappa_s * s_j \tag{5.25}$$

Where s_i is the postsynaptic spiketrain and V_i^* is the error between dendritic prediction and somatic rate and $h(u)$. The additional nonlinearity $h(u) = \frac{d}{du} ln \ \phi(u)$ is ommited in our model **TODO: should it though?**.

Antiderivatives:

$$\int_{-\infty}^{x} H(t)dt = tH(t) = max(0, t) \tag{5.26}$$

$$\tau_l = \frac{C_m}{g_L} = 10 \tag{5.27}$$

$$\tau_s = 3 \tag{5.28}$$

Writing membrane potential to history (happens at every update step of the postsynaptic neuron):

5.4 Dendritic leakage conductance

In order to match the dendritic potential of rate neurons in the spiking neuron model, a suitable leakage conductance for dendritic compartments was required. As described in Equation 2.20, a dendritic compartment evolves according to:

$$C_m^{dend} \dot{v}_j^{dend} = -g_l^{dend}\ v_j^{dend} + \sum_i W_{ji}\ \langle n_i \rangle \tag{5.29}$$

Under the assumption that the activation of all presynaptic neurons i remains static over time, we can replace the spontaneous activation $s_i(t)$ with the expected number of spikes per simulation step $\langle n_i \rangle = r_i\ \Delta t$ (cf. Equation 2.18). Note that these values do not employ matrix notation, but refer to individual neurons. Next, in order to find the convergence point of the ODE, we set the left side of the equation to 0 and to solve it:

$$0 = -g_l^{dend}\ v_j^{dend} + \sum_i W_{ji}\ r_i\ \Delta t \tag{5.30}$$

$$g_l^{dend}\ v_j^{dend} = W_{ji}\ r_i\ \Delta t \tag{5.31}$$

The instantaneous dendritic potential of rate neurons is given by $v_j^{dend} = \sum_i W_{ji}\ r_i$. Since we are searching for a parametrization which fulfils this equality in the steady state, the terms drop out from the above equation. Thus, the correct parametrization for the dendritic leakage conductance remains:

$$g_l^{dend} = \Delta t \tag{5.32}$$

It was shown experimentally that for high values of ψ, this parameterization leads to an exact match of dendritic potentials between the neuron models. It will therefore be assumed as the default throughout all experiments where spiking neurons are used.

In order to keep the two NEST models as similar as possible, rate neurons evolve according to the same dynamics. Like in the original implementation, dendrites of rate neurons ought

to be fully defined by their inputs at time t. This behaviour is achieved by setting the leakage conductance to 1 for all dendritic compartments. During network initialization, dendritic leakage conductances are set to either one of these values depending on the type of neuron model employed.

5.5 Plasticity in feedback connections

TODO: move to results

Within the present model, Pyramidal-to-pyramidal feedback weights evolve according to:

$$\dot{w}_l^{down} = \eta_l^{down} \left(\phi(u_l^P) - \phi(w_l^{down} r_{l+1}^P)\right) \phi(u_{l+1}^P)^T \tag{5.33}$$

The error term in this case differs slightly from the others, but could arguably still be implemented by biological neurons. An intuitive way to interpret the error term is as the difference between somatic activity and the activity of a distant apical compartment that is innervated only by superficial pyramidal neurons. Within the NEST implementation, this distal compartment leaks into the proximal apical compartment (v^{api}) with a conductance of $g^{api,dist} = 1$. The separation of pyramidal neuron apical dendrites into a proximal and a distal tree is well documented **TODO: cite**. A difference between plasticity mechanisms for synapses arriving at these two integration zones is plausible, although I was unable to find prior research supporting this type of plasticity **TODO: cite**. A more sophisticated model of the apical tree which resembles pyramidal neurons more closely could be a desirable extension to the model.

While the plasticity was successfully implemented in all variants of the model, it did not prove useful for training the networks during my tests. A strong indicator to the reason behind this is the fact, that the dendritic error for this rule is nonzero, even in the self-predicting state (cf. Fig. 2.3). Making these connections non-plastic led to the best learning performance, and is therefore assumed as the default for all training simulations. This matches the previous implementations of this network too, which typically set learning rates of these connections to 0 with the exception of a few experiments employing steady-state approximations. Note that feedback information is transmitted through fixed weights in this case. Feedforward weights in turn learn to match these, meaning that the network effectively implements a type of Feedback alignment Lillicrap et al. (2014).

5.6 Supplementary Figures

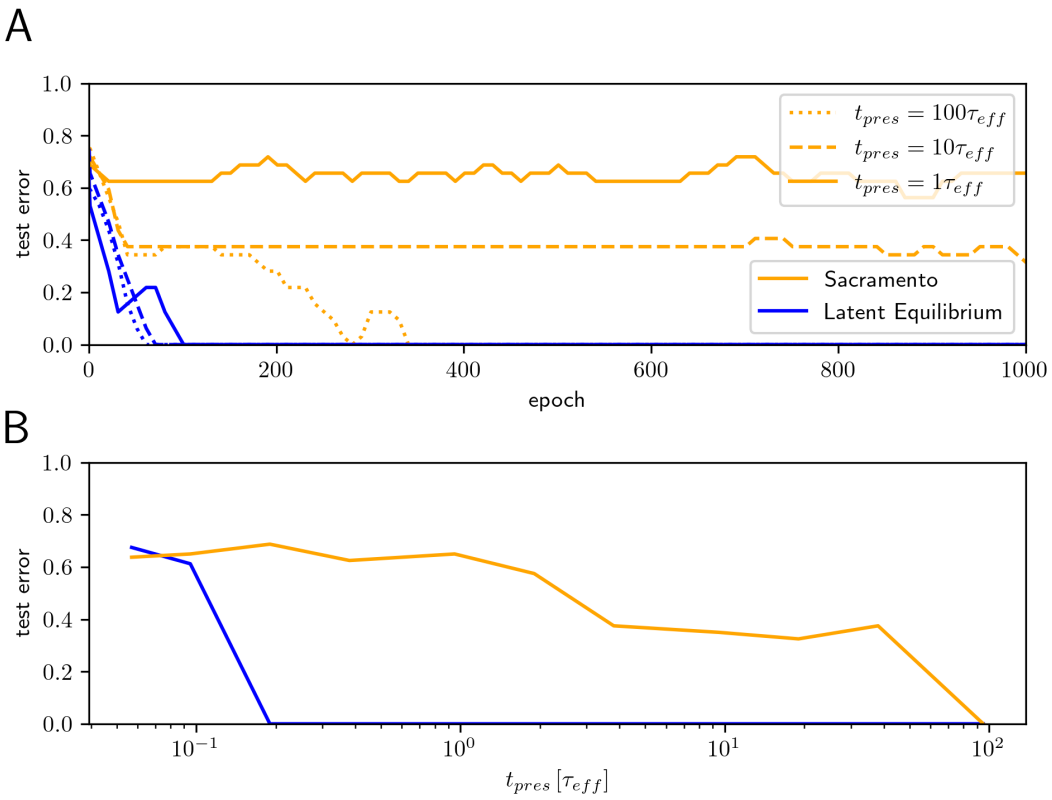

Fig. S1: Replication of Fig. 3.2 using a slightly modified version of the python code from Haider et al. (2021). Resulting performance matches the original results closely, showing that this version can serve as a baseline for comparing performance of the NEST implementation to the original results. Note, how in this implementation, presentation time has hardly any effect on the LE network because all updates are instantaneous. At the lower end presentation time is only limited by simulation timestep Δt.

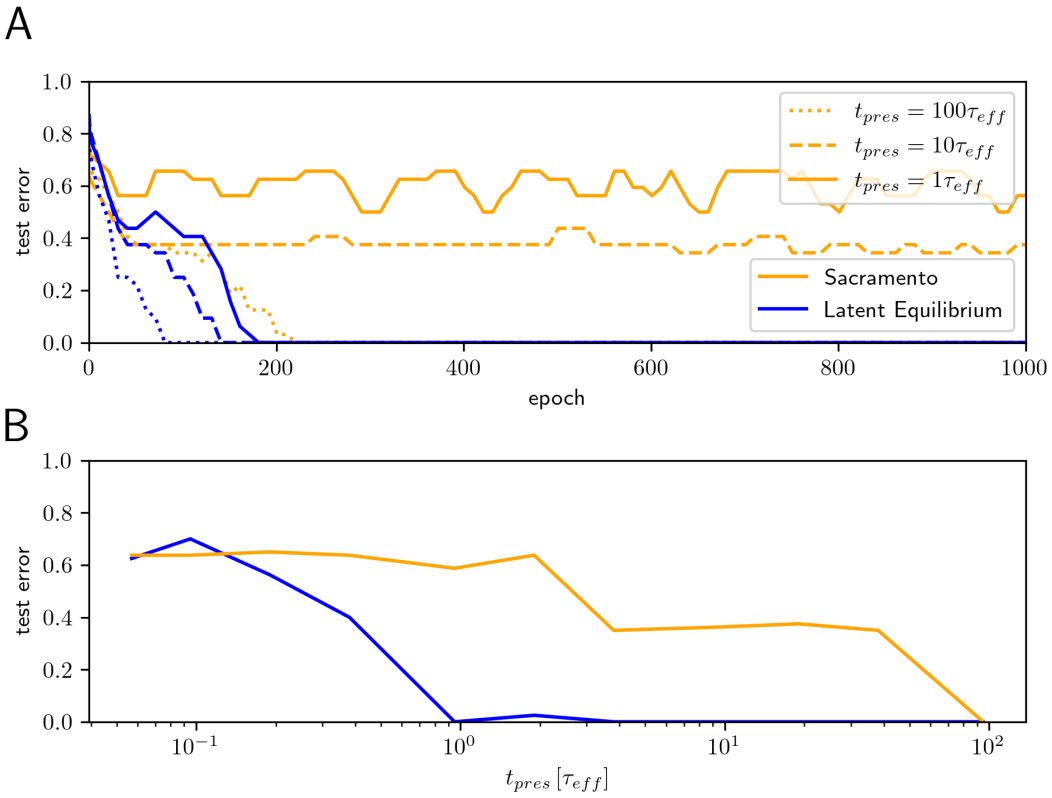

Fig. S2: Replication of Fig. 3.2 using networks of rate neurons in the NEST simulator. A notable difference to the python implementation in Fig. S1 is, that this version does not handle very low presentation times as well. This can likely be traced back to the synaptic delay enforced by NEST, which imposes an upper bound on network relaxation time. Besides that, performance of the two variants is very similar.

Chapter 6

additional notes and questions

6.1 Questions

- how interested are you in the code? reference it, explain it, or shut up?
- Code beigefügt als CD? GitHub link? egal?
- Citations in my motivation or not?
- No defense? is that correct?
- Mathematical notation for "variable is changed to" ($W \leftarrow \frac{W}{\psi}$)
- Schlechte ergebnisse oder gar keine ergebnisse reporten?
- Abgabe postalisch möglich/sinnvoll?
- 4 seiten probekapitel ok?
- Darf ich auch ein Probekapitel an Johan schicken?
- Reconsider title?
- Begutachtungsbogen: 2.1 Operationalizations?

6.2 TODOS

- I can "cheat" the apical voltage constraint for self prediction by increasing apical leakage conductance. How does this influence my model?
- Does the network still learn when neurons have a refractory period?
- talk about feedback plasticity
- redo the spike rate experiment, but investigate whether novel stimuli cause bursts.

- replace cite with citep
- investigate exc-inh split biologically
- Urbanczik-senn has little empirical background. shits for the outlook
- Spiking network as of yet can only process positive-valued input
- test time can be reduced by introducing separate t_pres

squared error / variance of training output = explained variance

6.2.1 Interneurons and their jobs

reciprocal inhibition of SST+ neurons. In agreement, recent experiments show that feedback input can gate plasticity of the feedforward synapses through VIP+ neuron mediated disinhibition 176 and that pyrami dal neurons indeed recruit inhibitory populations to produce a predictive error177 (Poirazi and Papoutsi, 2020)

6.2.2 tpres

$t_{pres} 10 - 50\tau$

Bibliography

Abbott, L. F. and Nelson, S. B. (2000). Synaptic plasticity: taming the beast. *Nature neuroscience*, 3(11):1178–1183.

Adams, R. A., Friston, K. J., and Bastos, A. M. (2015). Active inference, predictive coding and cortical architecture. *Recent Advances on the Modular Organization of the Cortex*, pages 97–121.

Ahmad, N., van Gerven, M. A., and Ambrogioni, L. (2020). Gait-prop: A biologically plausible learning rule derived from backpropagation of error. *Advances in Neural Information Processing Systems*, 33:10913–10923.

Barranca, V. J., Bhuiyan, A., Sundgren, M., and Xing, F. (2022). Functional implications of dale's law in balanced neuronal network dynamics and decision making. *Frontiers in Neuroscience*, 16.

Bartunov, S., Santoro, A., Richards, B., Marris, L., Hinton, G. E., and Lillicrap, T. (2018). Assessing the scalability of biologically-motivated deep learning algorithms and architectures. *Advances in neural information processing systems*, 31.

Bastos, A. M., Usrey, W. M., Adams, R. A., Mangun, G. R., Fries, P., and Friston, K. J. (2012). Canonical microcircuits for predictive coding. *Neuron*, 76(4):695–711.

Bellec, G., Scherr, F., Hajek, E., Salaj, D., Legenstein, R., and Maass, W. (2019). Biologically inspired alternatives to backpropagation through time for learning in recurrent neural nets. *arXiv preprint arXiv:1901.09049*.

Bellec, G., Scherr, F., Subramoney, A., Hajek, E., Salaj, D., Legenstein, R., and Maass, W. (2020). A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature communications*, 11(1):1–15.

Bengio, Y. (2014). How auto-encoders could provide credit assignment in deep networks via target propagation. *arXiv preprint arXiv:1407.7906*.

Bengio, Y., Lee, D.-H., Bornschein, J., Mesnard, T., and Lin, Z. (2015). Towards biologically plausible deep learning. *arXiv preprint arXiv:1502.04156*.

Bengio, Y., Mesnard, T., Fischer, A., Zhang, S., and Wu, Y. (2017). Stdp-compatible approximation of backpropagation in an energy-based model. *Neural computation*, 29(3):555–577.

Brea, J. and Gerstner, W. (2016). Does computational neuroscience need new synaptic learning paradigms? *Current opinion in behavioral sciences*, 11:61–66.

Chavlis, S. and Poirazi, P. (2021). Drawing inspiration from biological dendrites to empower artificial neural networks. *Current opinion in neurobiology*, 70:1–10.

Crick, F. (1989). The recent excitement about neural networks. *Nature*, 337(6203):129–132.

Davies, M., Srinivasa, N., Lin, T.-H., Chinya, G., Cao, Y., Choday, S. H., Dimou, G., Joshi, P., Imam, N., Jain, S., et al. (2018). Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1):82–99.

Doron, G., Shin, J. N., Takahashi, N., Drüke, M., Bocklisch, C., Skenderi, S., de Mont, L., Toumazou, M., Ledderose, J., Brecht, M., et al. (2020). Perirhinal input to neocortical layer 1 controls learning. *Science*, 370(6523):eaaz3136.

Eppler, J. M., Helias, M., Muller, E., Diesmann, M., and Gewaltig, M.-O. (2009). Pynest: a convenient interface to the nest simulator. *Frontiers in neuroinformatics*, 2:12.

Eyal, G., Verhoog, M. B., Testa-Silva, G., Deitcher, Y., Benavides-Piccione, R., DeFelipe, J., De Kock, C. P., Mansvelder, H. D., and Segev, I. (2018). Human cortical pyramidal neurons: from spines to spikes via models. *Frontiers in cellular neuroscience*, 12:181.

Felleman, D. J. and Van Essen, D. C. (1991). Distributed hierarchical processing in the primate cerebral cortex. *Cerebral cortex (New York, NY: 1991)*, 1(1):1–47.

Friston, K. (2008). Hierarchical models in the brain. *PLoS computational biology*, 4(11):e1000211.

Friston, K. and Kiebel, S. (2009). Predictive coding under the free-energy principle. *Philosophical transactions of the Royal Society B: Biological sciences*, 364(1521):1211–1221.

George, D. and Hawkins, J. (2009). Towards a mathematical theory of cortical micro-circuits. *PLoS computational biology*, 5(10):e1000532.

Gerstner, W. and Naud, R. (2009). How good are neuron models? *Science*, 326(5951):379–380.

Gewaltig, M.-O. and Diesmann, M. (2007). Nest (neural simulation tool). *Scholarpedia*, 2(4):1430.

Gidon, A., Zolnik, T. A., Fidzinski, P., Bolduan, F., Papoutsi, A., Poirazi, P., Holtkamp, M., Vida, I., and Larkum, M. E. (2020). Dendritic action potentials and computation in human layer 2/3 cortical neurons. *Science*, 367(6473):83–87.

Ginzburg, I. and Sompolinsky, H. (1994). Theory of correlations in stochastic neural networks. *Physical review E*, 50(4):3171.

Grossberg, S. (1987). Competitive learning: From interactive activation to adaptive resonance. *Cognitive science*, 11(1):23–63.

Guerguiev, J., Lillicrap, T. P., and Richards, B. A. (2017). Towards deep learning with segregated dendrites. *ELife*, 6:e22901.

Haeusler, S. and Maass, W. (2007). A statistical analysis of information-processing properties of lamina-specific cortical microcircuit models. *Cerebral cortex*, 17(1):149–162.

Haider, P., Ellenberger, B., Kriener, L., Jordan, J., Senn, W., and Petrovici, M. A. (2021). Latent equilibrium: A unified learning theory for arbitrarily fast computation with arbitrarily slow neurons. *Advances in Neural Information Processing Systems*, 34:17839–17851.

Häusser, M. and Mel, B. (2003). Dendrites: bug or feature? *Current opinion in neurobiology*, 13(3):372–383.

Hertäg, L. and Clopath, C. (2022). Prediction-error neurons in circuits with multiple neuron types: Formation, refinement, and functional implications. *Proceedings of the National Academy of Sciences*, 119(13):e2115699119.

Hines, M. L. and Carnevale, N. T. (1997). The neuron simulation environment. *Neural computation*, 9(6):1179–1209.

Ishizuka, N., Cowan, W. M., and Amaral, D. G. (1995). A quantitative analysis of the dendritic organization of pyramidal cells in the rat hippocampus. *Journal of Comparative Neurology*, 362(1):17–45.

Izhikevich, E. M. (2007). Solving the distal reward problem through linkage of stdp and dopamine signaling. *Cerebral cortex*, 17(10):2443–2452.

Kandel, E., Koester, J., Mack, S., and Siegelbaum, S. (2021). *Principles of Neural Science, Sixth Edition*. McGraw-Hill Education.

Kawaguchi, Y. (2001). Distinct firing patterns of neuronal subtypes in cortical synchronized activities. *Journal of Neuroscience*, 21(18):7261–7272.

Khaligh-Razavi, S.-M. and Kriegeskorte, N. (2014). Deep supervised, but not unsupervised, models may explain it cortical representation. *PLoS computational biology*, 10(11):e1003915.

Kubilius, J., Bracci, S., and Op de Beeck, H. P. (2016). Deep neural networks as a computational model for human shape sensitivity. *PLoS computational biology*, 12(4):e1004896.

Larkum, M. E. (2022). Are dendrites conceptually useful? *Neuroscience*, 489:4–14.

Larkum, M. E., Petro, L. S., Sachdev, R. N., and Muckli, L. (2018). A perspective on cortical layering and layer-spanning neuronal elements. *Frontiers in neuroanatomy*, page 56.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

Lee, C., Sarwar, S. S., Panda, P., Srinivasan, G., and Roy, K. (2020). Enabling spike-based backpropagation for training deep neural network architectures. *Frontiers in neuroscience*, page 119.

Lee, D.-H., Zhang, S., Fischer, A., and Bengio, Y. (2015). Difference target propagation. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part I 15*, pages 498–515. Springer.

Lee, J. H., Delbruck, T., and Pfeiffer, M. (2016). Training deep spiking neural networks using backpropagation. *Frontiers in neuroscience*, 10:508.

Legenstein, R., Pecevski, D., and Maass, W. (2008). A learning theory for reward-modulated spike-timing-dependent plasticity with application to biofeedback. *PLoS computational biology*, 4(10):e1000180.

Liao, Q., Leibo, J., and Poggio, T. (2016). How important is weight symmetry in backpropagation? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Lillicrap, T. P., Cownden, D., Tweed, D. B., and Akerman, C. J. (2014). Random feedback weights support learning in deep neural networks. *arXiv preprint arXiv:1411.0247*.

Lillicrap, T. P., Santoro, A., Marris, L., Akerman, C. J., and Hinton, G. (2020). Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346.

Magee, J. C. and Grienberger, C. (2020). Synaptic plasticity forms and functions. *Annual review of neuroscience*, 43:95–117.

Mancilla, J. G., Lewis, T. J., Pinto, D. J., Rinzel, J., and Connors, B. W. (2007). Synchronization of electrically coupled pairs of inhibitory interneurons in neocortex. *Journal of Neuroscience*, 27(8):2058–2073.

Marblestone, A. H., Wayne, G., and Kording, K. P. (2016). Toward an integration of deep learning and neuroscience. *Frontiers in computational neuroscience*, page 94.

Mazzoni, P., Andersen, R. A., and Jordan, M. I. (1991). A more biologically plausible learning rule for neural networks. *Proceedings of the National Academy of Sciences*, 88(10):4433–4437.

McCormick, D. A., Connors, B. W., Lighthall, J. W., and Prince, D. A. (1985). Comparative electrophysiology of pyramidal and sparsely spiny stellate neurons of the neocortex. *Journal of neurophysiology*, 54(4):782–806.

Meulemans, A., Carzaniga, F., Suykens, J., Sacramento, J., and Grewe, B. F. (2020). A theoretical framework for target propagation. *Advances in Neural Information Processing Systems*, 33:20024–20036.

Millidge, B., Seth, A., and Buckley, C. L. (2021). Predictive coding: a theoretical and experimental review. *arXiv preprint arXiv:2107.12979*.

Millidge, B., Tschantz, A., and Buckley, C. L. (2022). Predictive coding approximates backprop along arbitrary computation graphs. *Neural Computation*, 34(6):1329–1368.

Millidge, B., Tschantz, A., Seth, A. K., and Buckley, C. L. (2020). Activation relaxation: A local dynamical approximation to backpropagation in the brain. *arXiv preprint arXiv:2009.05359*.

O'Reilly, R. C. (1996). Biologically plausible error-driven learning using local activation differences: The generalized recirculation algorithm. *Neural computation*, 8(5):895–938.

Poirazi, P. and Papoutsi, A. (2020). Illuminating dendritic function with computational models. *Nature Reviews Neuroscience*, 21(6):303–321.

Potjans, T. C. and Diesmann, M. (2014). The cell-type specific cortical microcircuit: relating structure and activity in a full-scale spiking network model. *Cerebral cortex*, 24(3):785–806.

Rao, R. P. and Ballard, D. H. (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1):79–87.

Richards, B. A. and Lillicrap, T. P. (2019). Dendritic solutions to the credit assignment problem. *Current opinion in neurobiology*, 54:28–36.

Roelfsema, P. R. and Holtmaat, A. (2018). Control of synaptic plasticity in deep cortical networks. *Nature Reviews Neuroscience*, 19(3):166–180.

Sacramento, J., Ponte Costa, R., Bengio, Y., and Senn, W. (2018). Dendritic cortical microcircuits approximate the backpropagation algorithm. *Advances in neural information processing systems*, 31.

Scellier, B. and Bengio, Y. (2017). Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11:24.

Schiess, M., Urbanczik, R., and Senn, W. (2016). Somato-dendritic synaptic plasticity and error-backpropagation in active dendrites. *PLoS computational biology*, 12(2):e1004638.

Schmidt, M., Bakker, R., Hilgetag, C. C., Diesmann, M., and van Albada, S. J. (2018). Multi-scale account of the network structure of macaque visual cortex. *Brain Structure and Function*, 223(3):1409–1435.

Seung, H. S. (2003). Learning in spiking neural networks by reinforcement of stochastic synaptic transmission. *Neuron*, 40(6):1063–1073.

Sjostrom, P. J., Rancz, E. A., Roth, A., and Hausser, M. (2008). Dendritic excitability and synaptic plasticity. *Physiological reviews*, 88(2):769–840.

Sporea, I. and Grüning, A. (2013). Supervised learning in multilayer spiking neural networks. *Neural computation*, 25(2):473–509.

Spruston, N. (2008). Pyramidal neurons: dendritic structure and synaptic integration. *Nature Reviews Neuroscience*, 9(3):206–221.

Stapmanns, J., Hahne, J., Helias, M., Bolten, M., Diesmann, M., and Dahmen, D. (2021). Event-based update of synapses in voltage-based learning rules. *Frontiers in neuroinformatics*, page 15.

Thomson, A. M. and Bannister, A. P. (2003). Interlaminar connections in the neocortex. *Cerebral cortex*, 13(1):5–14.

Urbanczik, R. and Senn, W. (2014). Learning by the dendritic prediction of somatic spiking. *Neuron*, 81(3):521–528.

van Albada, S., Morales-Gregorio, A., Dickscheid, T., Goulas, A., Bakker, R., Bludau, S., Palm, G., Hilgetag, C.-C., and Diesmann, M. (2022). Bringing anatomical information into neuronal network models. In *Computational Modelling of the Brain*, pages 201–234. Springer.

Van Albada, S. J., Rowley, A. G., Senk, J., Hopkins, M., Schmidt, M., Stokes, A. B., Lester, D. R., Diesmann, M., and Furber, S. B. (2018). Performance comparison of the digital neuromorphic hardware spinnaker and the neural network simulation software nest for a full-scale cortical microcircuit model. *Frontiers in neuroscience*, 12:291.

van Elburg, R. A. and van Ooyen, A. (2010). Impact of dendritic size and dendritic topology on burst firing in pyramidal cells. *PLoS computational biology*, 6(5):e1000781.

Vaughn, M. J. and Haas, J. S. (2022). On the diverse functions of electrical synapses. *Frontiers in Cellular Neuroscience*, 16.

Vezoli, J., Falchier, A., Jouve, B., Knoblauch, K., Young, M., and Kennedy, H. (2004). Quantitative analysis of connectivity in the visual cortex: extracting function from structure. *The Neuroscientist*, 10(5):476–482.

Werfel, J., Xie, X., and Seung, H. (2003). Learning curves for stochastic gradient descent in linear feedforward networks. *Advances in neural information processing systems*, 16.

Whittington, J., Muller, T., Mark, S., Barry, C., and Behrens, T. (2018). Generalisation of structural knowledge in the hippocampal-entorhinal system. *Advances in neural information processing systems*, 31.

Whittington, J. C. and Bogacz, R. (2017). An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural computation*, 29(5):1229–1262.

Whittington, J. C. and Bogacz, R. (2019). Theories of error back-propagation in the brain. *Trends in cognitive sciences*, 23(3):235–250.

Yamins, D. L. and DiCarlo, J. J. (2016). Using goal-driven deep learning models to understand sensory cortex. *Nature neuroscience*, 19(3):356–365.

Zenke, F. and Ganguli, S. (2018). Superspike: Supervised learning in multilayer spiking neural networks. *Neural computation*, 30(6):1514–1541.