Philipps-Universität Marburg

Fachbereich 17
AG Allgemeine und Biologische Psychologie
AE Theoretische Kognitionswissenschaft

Learning in cortical microcircuits with multi-compartment pyramidal neurons

Masterarbeit
zur Erlangung des Master of Science (M.Sc.)
im interdisziplinären Master-Studiengang
„Kognitive und Integrative Systemneurowissenschaften“

Betreut durch:
Prof. Dr. Dominik Endres, Philipps-Universität Marburg
Prof. Dr. Johan Kwisthout, Radboud University Nijmegen

Vorgelegt von Johannes Gille
im Mai 2023

Eidesstattliche Erklärung
Ich versichere hiermit, dass ich die vorliegende Arbeit mit dem Titel

„Learning in cortical microcircuits with multi-compartment pyramidal neurons“

selbständig verfasst und keine anderen als die im Text angegebenen Hilfsmittel verwendet habe.

Sämtliche Textstellen, die im Wortlaut oder dem Sinn nach anderen Werken entnommen wurden, sind mit einer Quellenangabe kenntlich gemacht. Die Masterarbeit wurde in der jetzigen oder in ähnlicher Form noch bei keiner anderen Hochschule eingereicht und hat noch keinen sonstigen Prüfungszwecken gedient.

Experimentelle Daten wurden ordnungsgemäß und nachvollziehbar gesichert und dem Betreuer/der Betreuerin übergeben.

Marburg, den 22. Mai 2023 Johannes Gille

Contents

List of Figures

List of Tables

Abstract

The Backpropagation of errors algorithm is exceptionally capable of optimizing artificial neural networks and thus forms the backbone of modern machine learning. Despite its usefulness, it has long been regarded as impossible for brains to implement, as it relies on multiple biologically implausible mechanisms. Several of these have recently been overcome in models which could be replicated by biological neurons. After an extensive review of the recent literature on such models, one of them was selected for further investigation. The model - called the "Dendritic error network" - shows how a recurrently connected network of cortical neurons could approximate the Backpropagation algorithm. What sets it apart from competing models is the inclusion of multi-compartment models of pyramidal neurons and its close functional correspondence to predictive coding. The aim of this thesis was, to investigate if this model is capable of learning under yet more constraints on its biological plausibility. These include a transition to spike-based communication, as well as a strict segregation of excitatory and inhibitory neuron populations - as demanded by Dale's law. Further experiments investigated if the network can handle network topologies informed by the human neocortex which it professes to model. Results indicate that learning in the dendritic error network is largely unobstructed by such constraints. Thus, the model is uniquely capable of performing error-minimizing learning while conforming to many neuron- and network-properties informed by neuroscientific insights. This study comes to the conclusion that the dendritic error network is one of the most promising computational models for supervised learning in the neocortex and should therefore be investigated further.

Chapter 1

Introduction

1.1 Motivation

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

In recent years, there has been a resurgence of research by neuroscientists towards reconciling biological and artificial neural networks in spite of these concerns. This led to a number of experimental results indicating that brains might be capable of performing something very similar to Backpropagation after all. Furthermore, despite rigorous efforts, no unifying alternative to this learning principle was found which performs well enough to account for the brain's unmatched capabilities.

Hence, there now exists a vibrant community developing alternative ways to implement this algorithm - or some approximation of it. These novel approaches are capable of replicating an increasing number of properties of biological brains. Nevertheless, many issues 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 progressively adding biological features, attempting to retain learning performance in the process.

1.2 The Backpropagation of errors algorithm

The Backpropagation of errors algorithm (*Backprop*) (Schmidhuber, 2014) 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 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. As biological neural networks are likewise subject to the credit assignment problem, finding a general solution to it promises to be invaluable to neuroscience. For a long time Backprop was believed to be unsuitable for networks of biological neurons for several reasons.

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 the algorithm. In fact, Backprop as a way 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 entirety 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 questions regarding mechanisms for both computing and storing these errors remain unanswered as 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 back-propagated through a given connection should be proportional to its impact on output loss during the forward pass. To replicate this, a neuronal network implementing Backprop would require feedback connections that mirror both the precise 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 these, the feedforward and feedback synapses which need to be aligned 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 handling spike based communication requires more sophisticated neuron models than are typically employed in Backprop networks. 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 (cf. Section 1.4.1). Finally, these abstract neurons - at least in classical Backprop - have no persistence through time. Thus, their activation is dictated strictly by

instantaneous presynaptic activity, in contrast to the leaky membrane dynamics exhibited by biological neurons.

1.4 Overcoming biological implausibility

Backprop has remained the gold standard against which most attempts at modelling learning in the brain eventually are compared. 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 (Potjans et al., 2011; Mozafari et al., 2018; Sutton and Barto, 2018). 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 (Sutton and Barto, 2018), performance of global error/reward signalling stays far behind that of the credit assignment performed by 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 Backprop approximations have been developed, which are capable of locally representing errors. These algorithms encode errors in either activation changes over time or local membrane potentials. They will be discussed further in Section 1.6.

The weight transport problem was successfully addressed by a mechanism called *Feedback Alignment* (FA) (Lillicrap et al., 2014). This seminal paper shows that Backprop can still learn successfully when feedback weights are random. In addition to learning to represent an input-output mapping in forward weights, the network is trained to extract useful information from randomly weighted instructive pathways. The authors call this process *learning to learn*, and show that performance is even superior to 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 (summized as *Target propagation* algorithms) by

design do not require local error representations. While they therefore are not affected by the weight transport problem, they fall far behind traditional Backprop on more complex benchmark datasets like *CIFAR* and *ImageNet* (Bartunov et al., 2018).

Numerous approaches for implementing Backprop with 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 issue of computing the derivative over spiketrains has been solved in several 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. One aspect of this will be discussed in the upcoming section.

All of these studies 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 simultaneously, and some of them introduce 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 omitted 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 fact 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, hippocampus and amygdala, were shown to perform complex integrations of their synaptic inputs (Spruston, 2008). These

dendritic trees are capable of performing coincidence- (Larkum et al., 1999) and sequence detection (Branco et al., 2010) within their synaptic inputs. The size of dendritic trees is also known to discriminate regular spiking from burst firing pyramidal neurons (van Elburg and van Ooyen, 2010). Furthermore, pyramidal neuron dendrites are capable of performing computations, which were previously assumed to require multi-layer neural networks (Schiess et al., 2016; Gidon et al., 2020). See (Larkum, 2022) and (Poirazi and Papoutsi, 2020) for extensive reviews.
These neuroscientific insights have sparked hope that modelling dendritic compartments explicitly might aid machine learning in terms in both learning performance and energy efficiency (Chavlis and Poirazi, 2021; Guerguiev et al., 2017; Richards and Lillicrap, 2019; Eyal et al., 2018). It appears that, if not for computational gains, dendrites should be considered in any model attempting to explain the power of human learning. While the network discussed in this thesis simulates dendrites with very simple dynamics, the choice of model was strongly influenced by the fact that segregated dendrites were considered at all.

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 brain areas would be involved in Backprop-like learning. It is also unclear, to what level of detail these areas would 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 function focus more on reinforcement (Legenstein et al., 2008) or unsupervised learning (George and Hawkins, 2009; Häusler and Maass, 2017). 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 share the hypothesis that this kind of learning occurs predominantly in the neocortex (Marblestone et al., 2016).
The literature on the subject of learning historically appears to be somewhat split (although important exceptions have been published recently). On the one hand, the “machine-learning“ point of view largely considers the utility of network changes 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 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 solutions down to four algorithms that are both highly performant and largely biologically plausible. Due to impact of the paper on this thesis, their model comparison is depicted in Supplementary Table S1. The algorithms were in part selected for requiring minimal external control during training, as well as 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, which are highly regarded in the neuroscience literature (Magee and Grienberger, 2020; Brea and Gerstner, 2016). Yet in the plasticity rule also lies their greatest weakness, as synapses need to switch between the two opposing mechanisms 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 its own 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 show that through rather simple computations, these prediction errors can be minimized to obtain useful representations at each level of the hierarchy. They further show 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 being largely driven by cortico-cortical feedback connections in addition to the feedforward processes. 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 as a mechanism for unsupervised learning, through a slight modification it is also capable of performing Backprop-like supervised learning (Whittington and Bogacz, 2017). This is the third model considered in the review paper, in which values (i.e. predictions) and errors of a layer are encoded in separate, recurrently connected neuron populations. 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 later relaxed by showing that the model is capable of approximating Backprop for arbitrary computation graphs (Millidge et al., 2022). The neuron-based predictive coding network 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 perspective 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 advantages, 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 represented by membrane potentials of 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 com-

partments 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 predictive coding 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 is regarded in the literature as an important step towards integrating deep learning and neuroscience.

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, without 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 Pre-training 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 more complex 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 activation

[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.

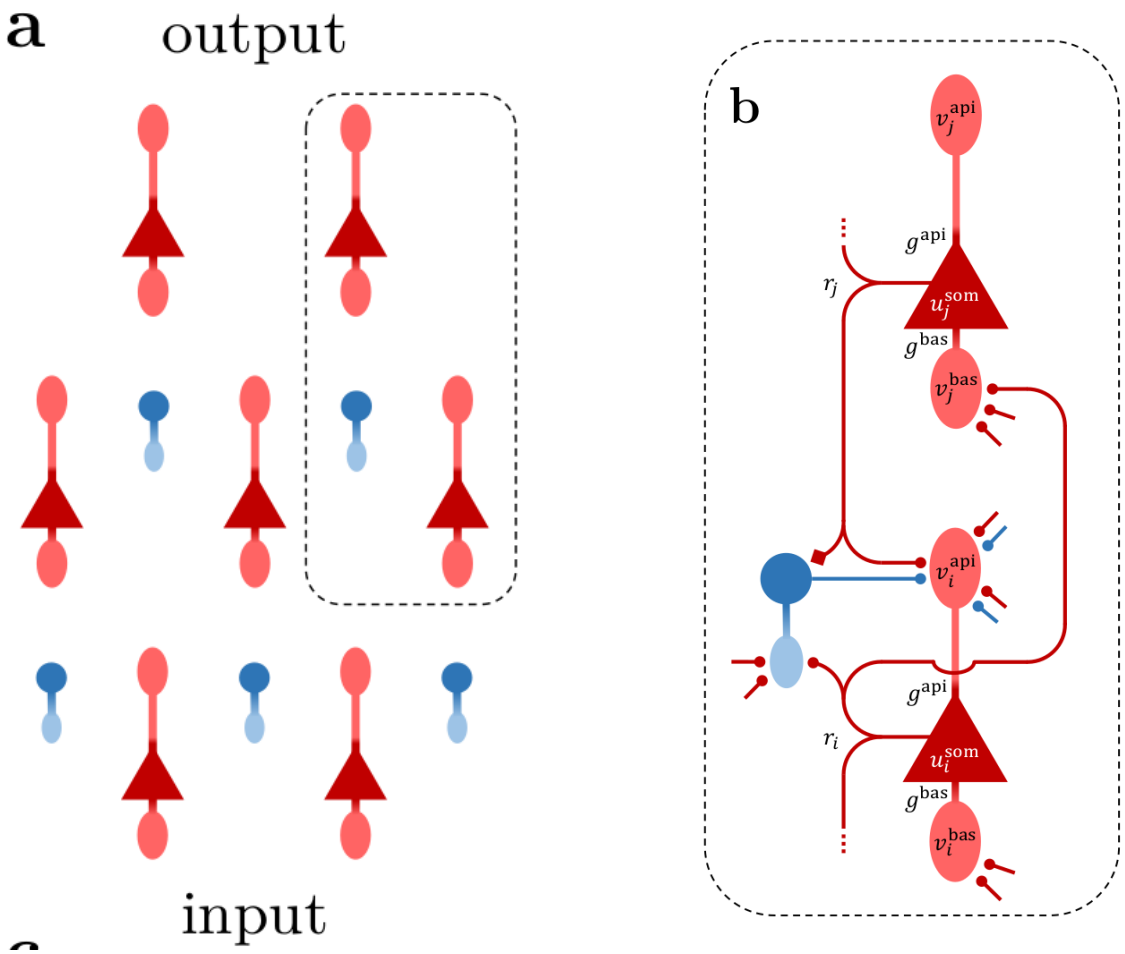

Fig. 2.1: Structure of the dendritic error network, from (Haider et al., 2021). **a:** pyramidal- (red) and interneurons (blue) in a network of three layers. Note the fact that the number interneurons in a layer is equal to the number of pyramidal neurons in the subsequent layer[1]. **b:** connectivity 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's somatic activation $\phi(u^{som})$ and fully connect their origin and target populations.

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 described in (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. These simplifications enabled the authors to prove analytically that this model approximates Backprop. 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 5.2 for more details on this case.

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 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 ($w_l^{ip} \approx \rho w_l^{up}$) and feedback ($w_l^{pi} \approx -w_l^{down}$) weights. *Mean squared error (MSE)* between these pairs of matrices will be called **Feedforward -** and **Feedback weight error** respectively.

[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.

- Silencing of pyramidal neuron apical compartments ($v_l^{api} \approx 0$). Mean absolute apical compartment voltage within a layer is called the **Apical error**.
- Equal activations in interneurons and their respective sister neurons ($\phi(u_l^I) \approx \phi(u_{l+1}^P)$). 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 exactly 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 signal from sister neurons is predicted perfectly, thus disabling plasticity in incoming synapses. Likewise, a silenced apical compartment will disable plasticity in all incoming synapses from interneurons. Furthermore, the apical compartment is also the driving factor for the dendritic error of feedforward synapses (Equation 2.13), since any nonzero potential leaks into the soma[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 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. The authors show analytically that this intricate mechanism can be recast as a gradient descent optimization.

2.4 Training the network

Training the network then requires only the injection of a target activation into the network's output layer alongside with a stimulus at the input layer. Since output layer neurons have

[3]This property might actually be considered the purpose of the Urbanczik-Senn plasticity. In the original paper, currents were injected directly into the soma to change the error term. In biological neurons, introducing a second dendrite which performs that very task makes far more sense.

strong feedback connections, a prediction error arises in the previous layer. Synapses then drive to minimize this error by creating a new self-predicting state in which interneurons mirror the novel behaviour of their sisters. Note that this interaction is not exclusive to the last two layers. Any Apical errors elicit a change in somatic activity, which preceding interneurons will fail to predict. Thus, errors are propagated backwards through arbitrarily deep networks, 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.

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:

[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.

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

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

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

[5]It should also be noted that in this approach requires redundant integration of the history by every synapse.

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 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:

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)).

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

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

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

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

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:

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

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

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

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

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

Computing this Equation directly is somewhat inefficient due to the nested integrals. Yet, the

[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 true 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.

authors show that 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 5.1 in (Stapmanns et al., 2021) for a rigorous proof that this is in fact the desired integral. 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 S2), 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 simulations 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 concerns over response time and learning speed can be solved by LE. The authors furthermore show, that 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.

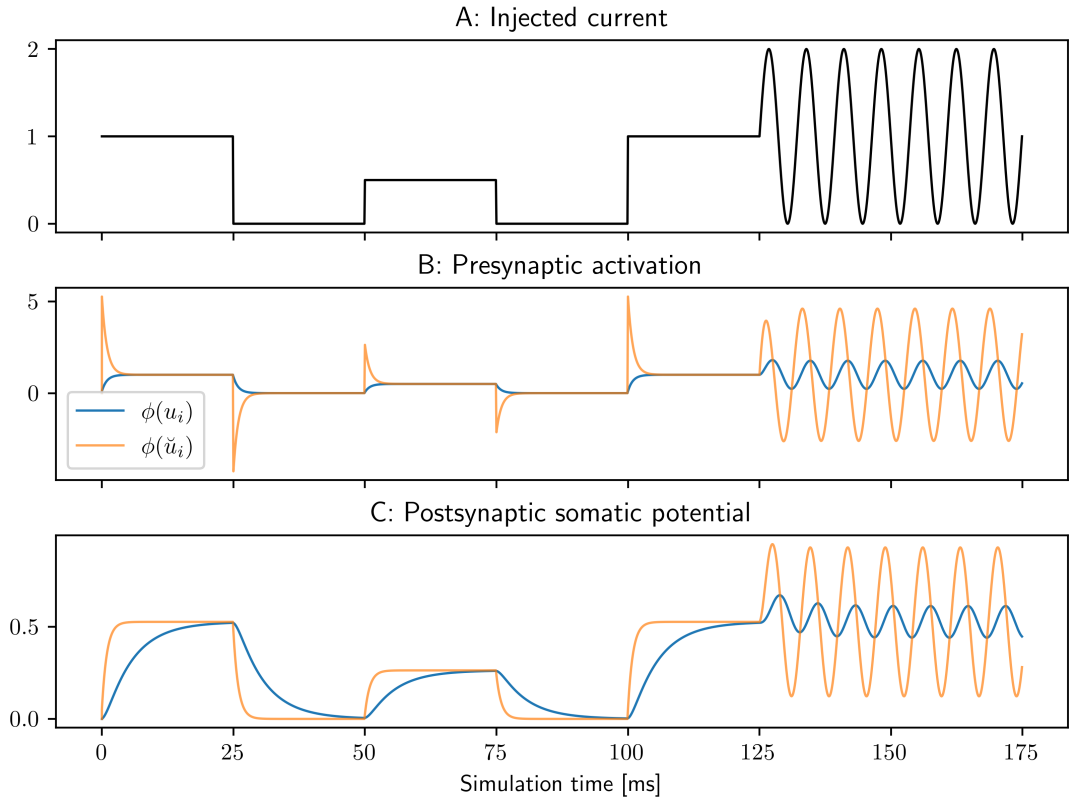

Fig. 2.2: Signal transmission of LE neurons. Shown is a connection between an input neuron i and a hidden layer pyramidal neuron j. Activations for the original Sacramento model (blue) and prospective activation using LE (orange) are compared. **A:** Current injected into the input neuron. Membrane potential slowly adapts to match this (not shown) **B:** Activation of the input neuron using instantaneous- $\boldsymbol{\phi}(u_i)$ (blue), and prospective activation $\boldsymbol{\phi}(\breve{u}_i)$ (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 mechanisms 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).

2.9 Implementational details

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

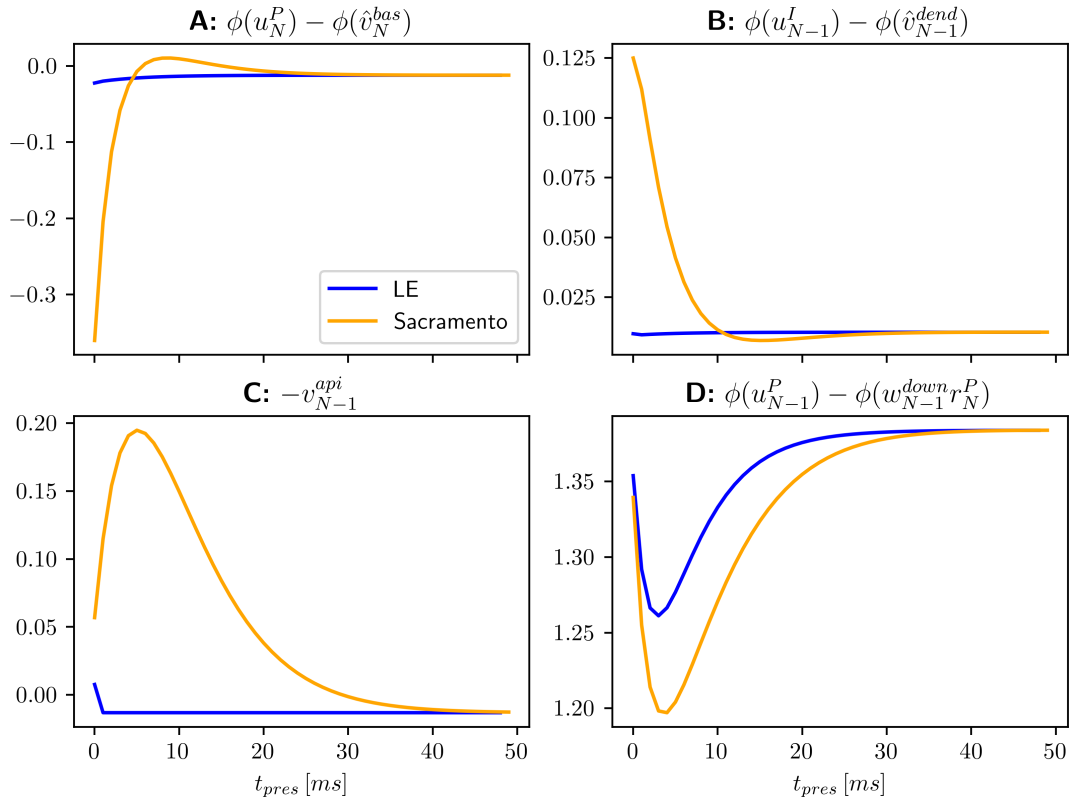

Fig. 2.3: Effects of LE dynamics on the dendritic error terms from Equations 2.13-2.16. Depicted are error terms for individual spiking neurons in a network with one hidden layer (N=3). The network was fully trained on the Bars dataset (cf. Section 3.2), so errors should ideally relax to zero. Note, 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 to zero for either network, an issue that will be discussed in Section 5.2.

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.

[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**.

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 several 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 5.1.

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 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

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

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 only for $\psi < 0.1$ did pyramidal and interneurons exhibit spike frequencies in biologically plausible range of less than $55Hz$ (Kawaguchi, 2001; Eyal et al., 2018). Additionally, each transmitted `SpikeEvent` is computationally costly, which increases training time (cf. Fig. 3.9) and therefore 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 is far removed from biologically plausible spike frequencies. 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 (FF error), Feedback weight error (FB error), Apical error and Interneuron error.* These were introduced in Section 2.3[10]. 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

[10]Observation of the network dynamics reveals that pairs of them are closely related: FF error drives interneuron error, and FB error drives apical error as soon as interneuron error is minimal. An analytical upgrade to this model might include a way for unifying these pairs of metrics.

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:

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

With M neurons in the output layer, which is again averaged over N test samples. Due to the network's relaxation period, y^{pred} can not be accurately computed instantaneously[11]. 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 layer membranes fluctuate strongly. therefore, y^{pred} is an average over recorded somatic potentials for each output neuron. This recording typically starts after $\sim 70\%$ of t_{pres} has passed.

[11] 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 lateral and feedback weights have no impact on pyramidal neuron activity. Particularly for the spiking variant, this assumption was shown to be 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 prioritization of gaining neuroscientific insights over achieving minimal test loss. It should be noted, that training the network is computationally quite costly (c.f. Section 3.7) which turned parameter studies into a time-consuming process.
Early experiments showed that the network is rather sensitive to parameter changes. The search for default parameters 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 LE implementation. Unless stated otherwise, neurons employ prospective activation functions in all simulations. So far, no drawbacks to this mechanism have presented themselves, and learning speed can be increased drastically compared to the original implementation. The full default parametrization is shown in Supplementary Table S2. 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 upcoming sections. Note that not all experimental results were given their own Figures. In these cases, plots can be found in the electronic supplementary material.

3.1 The self-predicting state

As a first comparison between the three implementations, 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

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

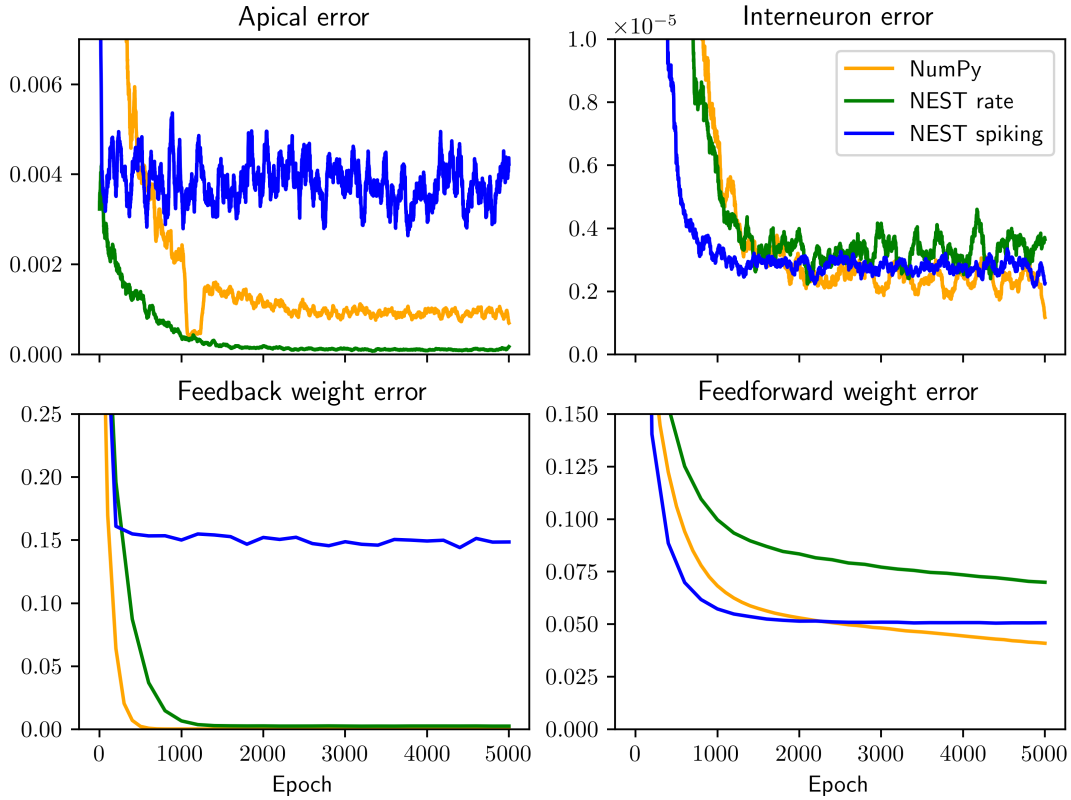

Fig. 3.1: Training towards the self-predicting state. All implementations learn to predict self-generated top-down signals. Networks were initialized with the same random weights for dimensions $[5, 8, 3]$, and stimulated with 5000 samples of random input for $100ms$ each. As described in (Sacramento et al., 2018), during this phase only $Pyr \rightarrow Intn$ and $Intn \rightarrow Pyr$ weights are plastic ($\eta^{pi} = 0.05, \eta^{ip} = 0.02375, \eta_0^{up} = \eta_1^{up} = \eta^{down} = 0$).

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 both training environment (current injections, simulation time, membrane reset, readouts, etc.) and the actual neuron model of the NEST version adequately replicate the original model.

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 FB error. After appearing to converge very quickly, the two metrics stagnate at very high levels. These high errors correlate with strong fluctuations of the apical compartment. These fluctuations can likely at least in part be attributed to low spike frequencies. This was confirmed by repeating the experiment with $\psi = 1500$, which alleviated the issue to a degree

(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 smooth out the fluctuations induced by individual spikes. This will be discussed in Section 3.4.
In most simulations in the literature, the network is initialized to an ideal self-predicting state. Furthermore, feedback weights are non-plastic in many experiments ($\eta^{pi} = \eta^{down} = 0$). Therefore, a failure to perfectly learn this weight symmetry should not fundamentally hinder learning. 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 with 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 LE. Due to the costly computation of the network under such long t_{pres}, a simple artificial classification dataset was used. 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 $Pyr \rightarrow Pyr$ and $Pyr \rightarrow Intn$ 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 spiking NEST network 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 is an important finding as it answers two open questions: Changes made for a NEST-compatible implementation were adequate and result in identical learning between the rate-based implementations. Learning performance of the spiking model is competitive, 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 LE network 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. For very low presentation times, this behavior was somewhat expected, due to the synaptic delay enforced by NEST. The NumPy variant computes a full forward pass of the network during a single simulation step,

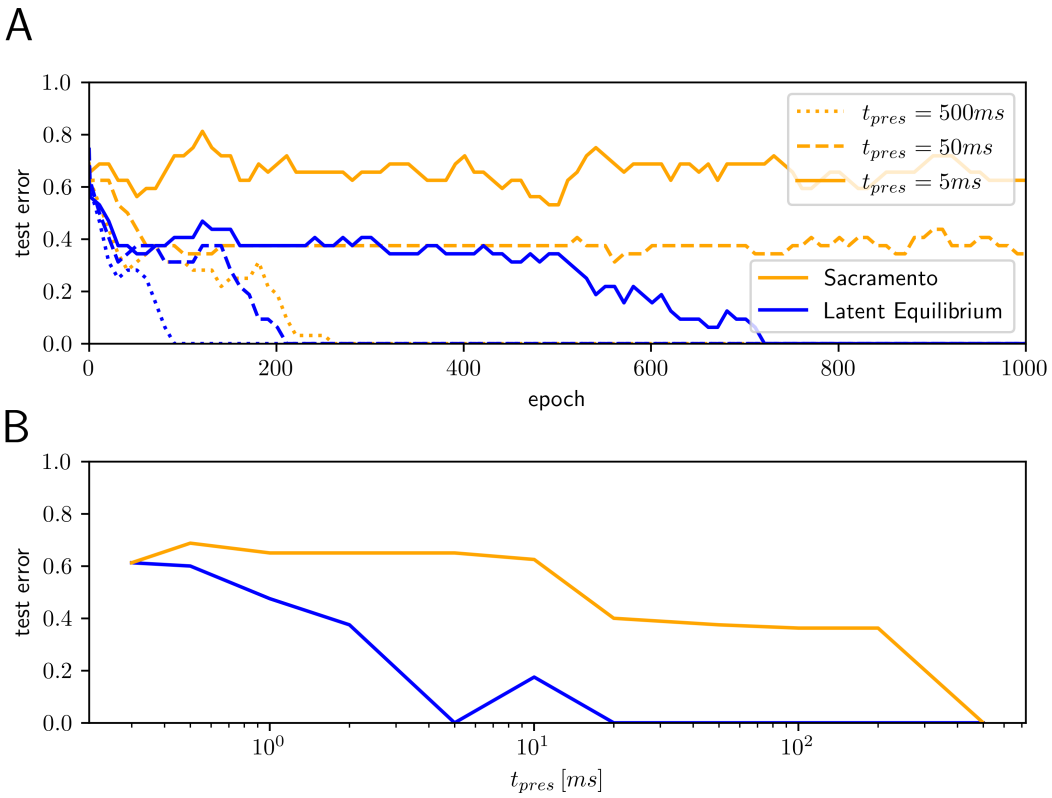

Fig. 3.2: Replication of Fig. 3 from (Haider et al., 2021) using networks of spiking neurons in the NEST simulator. **A:** Comparison between Original dendritic error network by and an identical network employing Latent equilibrium. Shown is the training of networks with 9-30-3 neurons on the Bars-dataset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

as all layers are processed in sequence. Only feedback signals from pyramidal neurons are delayed by one 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, as signals have no time to traverse the network. It remains an open question whether this feature alone explains the gradual decrease in performance observed here, or if there is an undiscovered error within the novel neuron models or simulation environment. The exceptionally short stimulus presentation times investigated by (Haider et al., 2021) 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 final accuracy, it required - at best - twice as many stimulus presentations as its direct competitor. This result, while somewhat expected, shows that for low t_{pres}, spiking communication leads to worse learning performance. It also shows that the relaxation problem affects all communication schemes equally. While the utility of LE is substantially

higher for rate neurons, it does improve performance and efficiency of the spiking variant. For this reason, LE will be turned on in all upcoming simulations.

3.3 Approximating arbitrary functions

To confirm that the spiking network is capable of learning more complex tasks, it was trained to match the input-output mapping of a separate teacher network. This is an established method for showing that a network can approximate arbitrary functions. A performance comparison of several networks with different numbers of neurons in their hidden layers is depicted in Fig. 3.3.

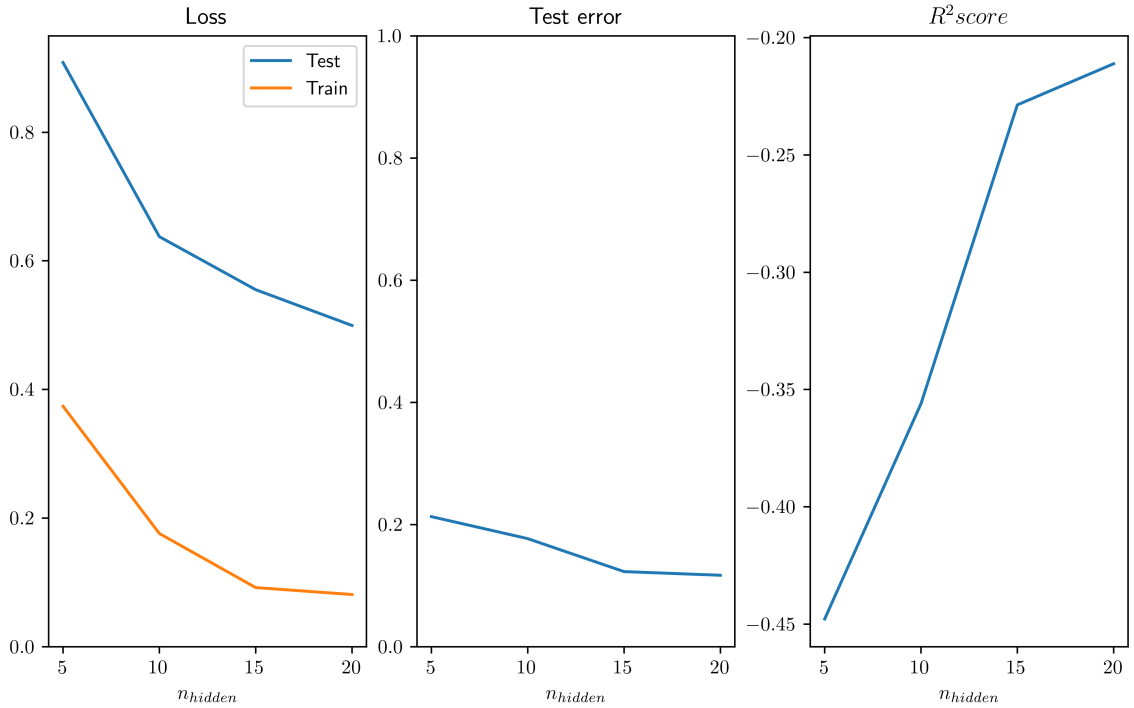

Fig. 3.3: SNN learns to match separate teacher network. Networks of size $15 - n_{hidden} - 5$ neurons per layer were trained on input-output pairings of a randomly initialized feedforward network of size $15 - 15 - 5$. Each network was trained on 500000 samples of the same teacher network while all somatic compartments received background noise with a standard deviation of 0.01. To measure how much of the teacher network's variance is predicted by the dendritic error network, the R^2-score (sklearn.metrics.r2_score) was measured for a large final test run over 500 samples.

As the task was found to be too easy when inputs to the teacher network were strictly positive (as is the case in the self-prediction paradigm), inputs were drawn from a uniform distribution $U(-1, 1)$. Note that (as is typical for many neural networks), input neurons do not employ the nonlinearity ϕ. Thus, in rate neurons inhibitory inputs are transmitted directly and multiplied with synaptic weights. For spiking neurons, any negative injected current will effectively inhibit spike generation. Thus, negative inputs can not be transmitted. To facilitate inhibitory stimulation to the spiking network, a separate input layer population was required. This

population was initialized with the inverted weights of the excitatory population. An input vector was then separated, with positive values stimulating the excitatory population and negative values stimulating inhibitory neurons. Due to this necessity, the spiking network effectively had to learn an additional set of weights, which must be considered when assessing these results.
Results show that surprisingly small networks are capable of matching the teacher network approximately. However, particularly for explaining the variance in the teacher network's output, at least an equally sized network is required. While the training task is complex, results stayed somewhat behind expectations, particularly for the network of identical size ($n_{hidden} = 15$). Nonetheless, the results show that the network successfully learns the required input-output mapping without resorting

3.4 Apical compartment capacitance

Next, an investigation was made into lowering apical- and FB errors of the spiking implementation. The hypothesis was that a smoothing of apical compartment voltage would lead to a decrease in both errors. There is physiological data supporting such experiments, as the surface area of pyramidal neuron dendrites outscales the soma by several orders of magnitude (Ishizuka et al., 1995). According to the Neuronal cable theory, an increase in surface area should correspond to an increase in overal membrane capacitance of a neuronal compartment (Niebur, 2008).
To test the effects of this, the Self-predicting experiment was repeated with numerous values for apical compartment capacitance $C_m^{api} \in \{1, 250\}pF$ (results not shown). The experiment showed that for $C_m^{api} = 50pF$, apical error is almost halved ($0.0034 \rightarrow 0.0019$), and FB error is decreased by 80% ($0.15 \rightarrow 0.027$). These values are still at least an order of magnitude higher than those in the rate implementations, but mark a substantial improvement. Increasing the parameter beyond this point further decreased apical error, but came at the cost of slower convergence. Higher membrane capacitances in general increase the relaxation period of the entire network. Thus, they require a highly undesirable increase in t_{pres} for successful learning.
A secondary objective of training with different apical capacitances was to enable learning with lower ψ and therefore approach biologically plausible firing rates. To test this, training on the Bars dataset was performed under different combinations of apical capacitance and ψ. Results of this experiment are shown in Fig. 3.4.
For $\psi = 50$, increasing the membrane capacitance correlated directly with decreased performance. Curiously, the same was true for $\psi = 5$, but not necessarily for the intermediate value of $\psi = 10$. In this special case, increasing apical capacitance to a degree did improve performance. A possible explanation for this is, that a 'sweetspot' for apical capacitance exists, in which the hypothesized stabilization of apical potential outweighs the drawbacks of increased relaxation time. This parameter study remains inconclusive about this hypothesis, as too

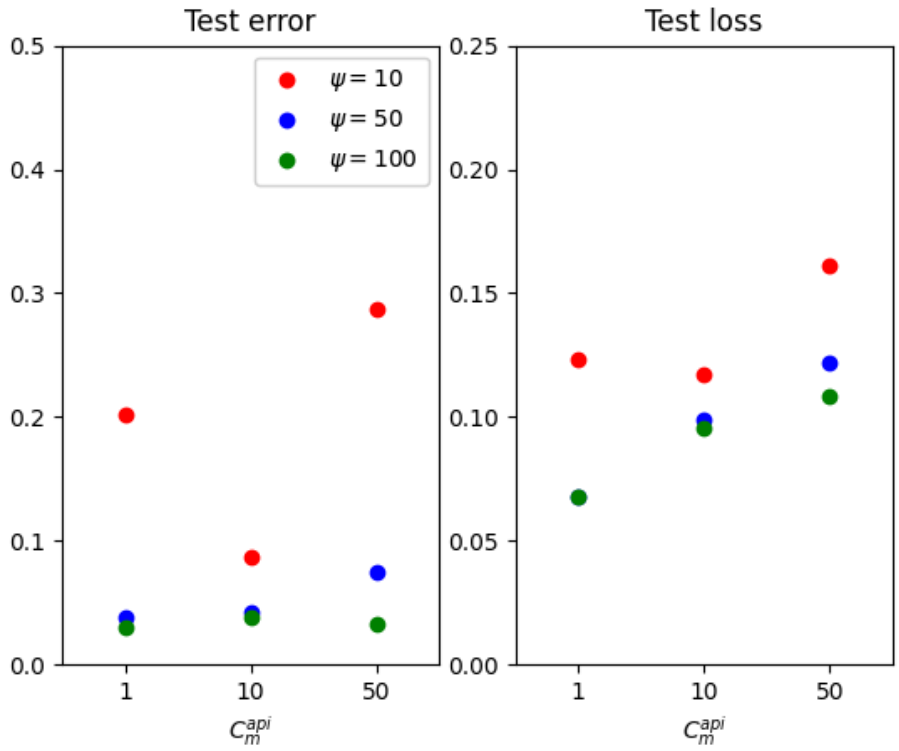

Fig. 3.4: Comparison of performance for different configurations of ψ and C_m^{api}. Networks were initialized with random weights, and trained for 750 epochs. Plasticity was enabled in all synapses ($\eta^{pi} = 0.0025, \eta^{ip} = 0.001, \eta_0^{up} = 0.0025, \eta_1^{up} = 0.00075, \eta^{down} = 0$). Stimuli were presented for $t_{pres} = 100ms$ to ensure that networks with higher apical capacitance (and therefore longer relaxation periods) were not disadvantaged too much.

few combinations of parameters are studied. Therefore, further experiments are required to determine if the utility of this parameter for self-prediction can transfer to actual learning. I expect that increased apical capacitance combined with longer stimulus presentation times could enable a decrease in ψ.

As basal compartments (and likewise FF error) did not fluctuate nearly as strongly, their capacitances were not considered in the present work. While substantially smaller than the apical tree, their membrane capacitance should similarly outscale the soma in future experiments.

3.5 Imperfect connectivity

Connectivity within cortical circuits, while structured, appears to subject to a high degree of randomness (Potjans and Diesmann, 2014). As noted before, one-to-one connections between pairs of neurons are therefore highly unlikely (Whittington and Bogacz, 2019). On the other hand, 'fully connected' populations of neurons likewise have not been observed in electrophysiological (Thomson et al., 2002) and in-vitro (Binzegger et al., 2004) analyses of

cortico-cortical connectivity. Therefore, any network proclaiming to model the cortex must invariably be capable of handling this imperfect connectivity.

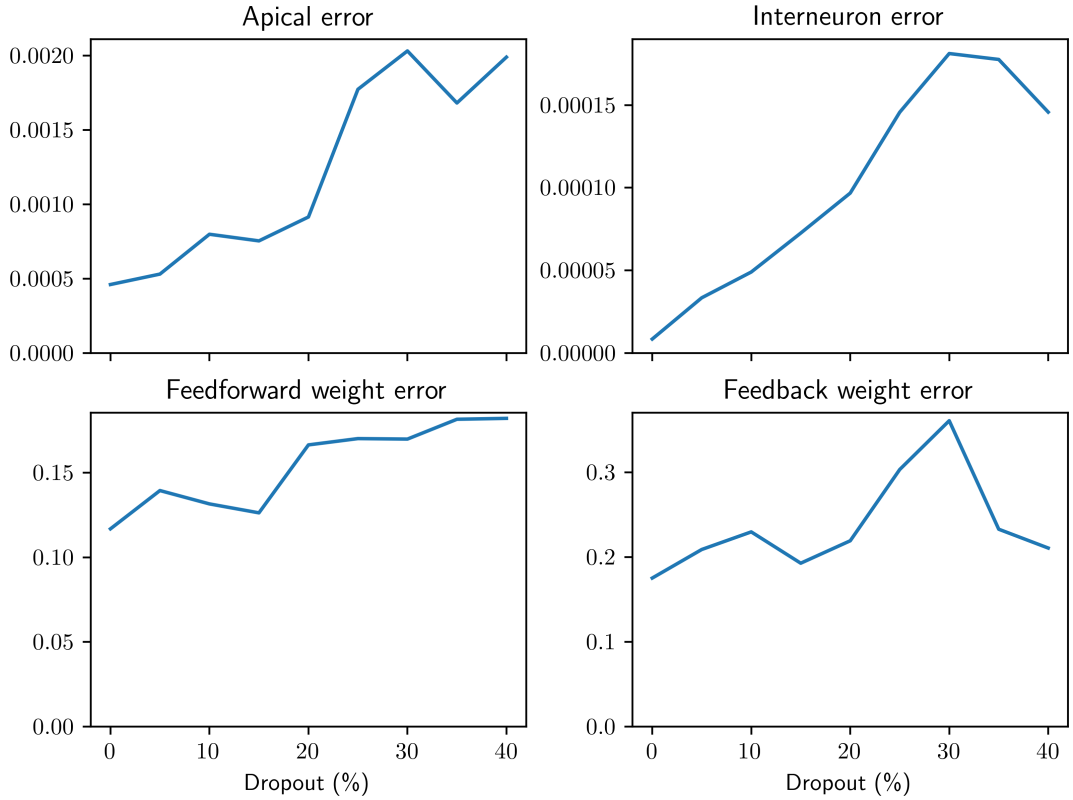

Fig. 3.5: Error terms after training with synapse dropout. Networks with $8 - 8 - 8$ neurons per layer were trained towards the self-predicting state, with different percentages of synaptic connections randomly removed. Experiments were performed with the rate-based network in NEST, each network was trained for 2000 epochs of 50ms each. Errors are averaged over 6 independent runs for each configuration.

To test if the dendritic error model fulfills this requirement, in a first step the self-prediction experiment was repeated with neuron dropout. To simulate connection probabilities $p_{conn} \in 0.6, 1.0$, an appropriate number of synapses was deleted after network setup. To avoid completely separating two neuron populations, this deletion was performed separately for each of the four synaptic populations.

As expected, removing synapses caused an increase in all four error metrics. Yet even with only 60% of synaptic connections present, the network manages to vastly improve from its random initialization. Weight errors are calculated as mean squared errors over the two matrices, which requires matrices to contain data at every cell. Thus, to compute these errors, weights of deleted synapses were set to zero in these matrices. 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. These results indicate that the dendritic error rule is capable of compensating for absent synapses by correctly identifying and depressing

corresponding feedback connections (and vice versa). Through this mechanism, the network is able to retain its self-predicting properties in spite of physiological constraints.
The extent of this capability was confirmed in SNN, by comparing performance on the Bars dataset of a control network to a *dropout network*. Both networks were initialized with random synaptic weights, and trained with full plasticity ($\eta_0^{ip} = 0.004, \eta_0^{pi} = 0.01, \eta_0^{up} = 0.01, \eta_1^{up} = 0.003$) to best enable the dropout network to compensate for missing synapses. The dropout network was initialized with 40 instead of 30 hidden layer pyramidal neurons to counteract the deletion of 15% of synapses per synaptic population. Both networks performed very similarly, with the control network reaching 100% accuracy somewhat faster (Epoch 140 vs. Epoch 200), while the dropout network exhibited slightly lower test loss at the end of training (results not shown).
These results prove that the dendritic error model is capable of learning in spite of imperfect connectivity, which must be expected to occur in the cortex. This sets it apart from the previous implementation of a predictive coding network (Whittington and Bogacz, 2017), and further supports its biological plausibility.

3.6 Separation of synaptic polarity

A dogma held in neuroscience for a long time now has been the notion that all neurons are either excitatory or inhibitory, as dictated by their type (also known as Dale's law (Kandel, 1968)). Several studies have since shown that some neurons violate this law through co-transmission or specific release sites for different neurotransmitters (Svensson et al., 2019; Barranca et al., 2022). Despite these findings, pyramidal neurons are still regarded to release exclusively Glutamate, therefore being strictly excitatory (Gerfen et al., 2018; Spruston, 2008; Eyal et al., 2018). This has been considered a rather weak criticism of the biological plausibility of Backprop (Bartunov et al., 2018). Yet given that the dendritic error model explicitly proclaims to model pyramidal neurons, this concern is ripe to be addressed.
Initial experiments showed that restricting any synaptic population in the network to just one polarity, the network is unable to reach the self-predicting state. Since the network relies on an intricate balance of excitation and inhibition (e.g. to minimize Apical error), this result is to be expected. Thus, activity in any neuron must be able to have both excitatory and inhibitory postsynaptic effects facilitated by appropriate synaptic weights. The most likely means by which a neural network could achieve this separation is, to introduce an interneuron population of opposite polarity between a population and its target.
To investigate to what degree the dendritic plasticity rule can deal with this constraint, an experiment was conducted: A population of (excitatory) pyramidal neurons A was connected to another population C with plastic synapses that were constrained to positive weights. In order to facilitate depression in C, each neuron in A was also connected to a single inhibitory interneuron in population B. This set of synapses was randomly initialized with positive

weights and non-plastic during this simulation. 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 ($B \to C$) decayed, while excitatory synaptic weights ($A \to C$) increased. Flipping the sign of that error injection had the opposite effect on weights, and likewise cancelled the artificially induced 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.

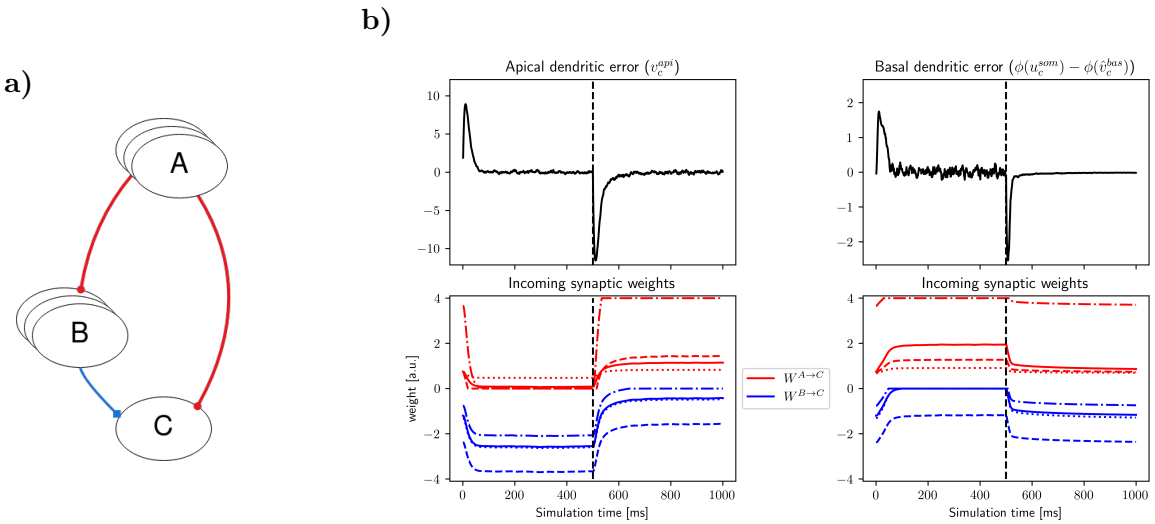

Fig. 3.6: Error minimization under biological constraints on synaptic polarity and network connectivity. **a)** Network architecture. An excitatory population A connects to a dendrite of Neuron C both directly and through inhibitory interneuron population B. Only synapses $A \to C$ and $B \to C$ are 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.

Yet, as criticized previously (Whittington and Bogacz, 2019), the one-to-one connections between A and B are untypical for cortical networks (Douglas and Martin, 2004; Gordon and Sten, 2010). Hence, a second experiment was performed in which neurons in populations A and B were fully connected. This decrease in specificity of the connections did not hinder the error-correcting learning, as shown in Fig. 3.6.

While error minimization is important, it does not necessarily imply that synaptic credit assignment is successful as well. Numerous weight configurations are conceivable 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 selected for the first test, as the employed plasticity rule had proven most resilient to parameter imperfections previously. A network of rate neurons was set up and parametrized as described in Section 3.2 ($t_{pres} = 50ms$). The Weights w^{pi} were redrawn and restricted to positive values, and a secondary inhibitory interneuron population was created and fully connected to both populations as described in Fig. 3.6. The inhibitory interneuron population was chosen to be 4 times as large as the target pyramidal population, and 30% of incoming excitatory connections were randomly deleted. The idea behind this was, to seed interneurons which were to serve as inhibitory counterparts for individual excitatory partners. From this seeding, the dendritic error rule could then ideally derive useful information about presynaptic activity.
The experiment was successful, as the network was able to learn successfully in competitive time (100% accuracy after 200 Epochs) albeit to a higher final test loss (results not shown). These results show that the dendritic plasticity rule is capable of correctly assigning credit to two separate populations when its inputs are much less sanitized. Further experiments are required to show **A:** how large such an inhibitory interneuron population needs to be and what role synapse dropout plays in it, **B:** whether this capability extends to the spiking implementation and **C:** if all neuron populations in the network can be connected in this way to separate excitatory and inhibitory pathways. Such experiments would allow for a closer investigation into how well the dendritic error network corresponds to cortical connectivity - if at all. The novel neuron populations introduced by such experiments would themselves have to have some cortical equivalent which they are to represent.
These results furthermore enable the conception a biologically plausible way for long-range pyramidal projections to innervate neurons in another layer of the dendritic error model (i.e. in a different cortical area). The steps required to facilitate this type of network are rather simple; A pyramidal neuron projection could enter a distant cortical area and spread its axonal tree to innervate both pyramidal- and inhibitory interneuron dendrites. If these interneurons themselves connect to the local pyramidal population, Dendritic errors with arbitrary signs and magnitudes could be minimized. While this is unlikely to be the case in the cortex, pyramidal-to-interneuron weights do not need to be plastic, as long as all synapses arriving at the target pyramidal population are.

3.6.1 Interneuron nudging

An easily overlooked connection of this network is the nudging signal from pyramidal neurons to their interneuron sisters. These were deliberately not included in the previous dropout studies. If any interneuron was to not receive its nudging signal, its incoming synapses would be unable to adapt their weights. As a result, both interneuron- and FF error would fail to converge, in turn impeding apical error reduction. These one-to-one connections can therefore be considered the most important communication channels in the network. If there is no redundancy in the neurons, deleting any of them breaks the network's learning scheme. Sacramento et al. claim

that the interneurons of the network resemble somatostatin-expressing (*SST*) neurons. This is a reasonable assumption, as SST cells are ubiquitous in the cortex and inhabit the same layers as pyramidal neurons. Furthermore, they share dense and recurrent synaptic connections to these pyramidal neurons (Urban-Ciecko and Barth, 2016). Finally, they have been shown to receive top-down instructive signals, which have been hypothesized to transmit prediction errors (Leinweber et al., 2017).
Several experiments similar to those on synaptic polarity were conducted in an attempt to replace these one-to-one connections with more plausible connectivity schemes. Regrettably, none of them were able to retain the learning capability of this network. Thus, these connections remain as perhaps the biologically most implausible aspect of the dendritic error network. Further work is required to investigate if and how this constraint can be relaxed.

3.7 Performance of the different implementations

As stated in (Haider et al., 2021), simulating large dendritic error networks with the full leaky dynamics quickly becomes unfeasible. While the NEST simulator can be regarded as rather fast (Van Albada et al., 2018), simulations on it by design cannot employ batched matrix multiplication, as is typical in machine learning. Thus, by computing neuron updates individually even in highly structured networks like this one, NEST was expected to perform worse than previous implementations using PyTorch and dedicated GPUs. Yet not only did the NEST implementations compute rather slowly, the spiking variant was the slowest across the board. To investigate the extent of this, as well as possible causes, several benchmark experiments were performed. These tests were run on an *AMD Ryzen Threadripper 2990WX* using 8 cores at up to $3.0GHz$. All reported simulation times t_{sim} are averaged over 5 independent runs, and only measure the time taken simulating (in seconds) without considering network initialization.

To compare how network size affects simulation time, all three implementations created for this project were trained on 10 examples of the Bars dataset with different numbers of hidden layer pyramidal neurons. The result of this comparison is shown in Fig. 3.7. 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.
The NumPy variant is an outlier, and only listed here for completeness. It is the only variant running on a single thread due to a limitation of NumPy. This could feasibly be improved greatly by using batched matrix multiplications, as are provided for example by `PyTorch`. The original implementations do this, but for practical reasons the Backend was changed here. Notably, this variant exhibits very little slowdown in response to an increase in network size. It seems, that the vectorization of updates on a single thread scales better with network size

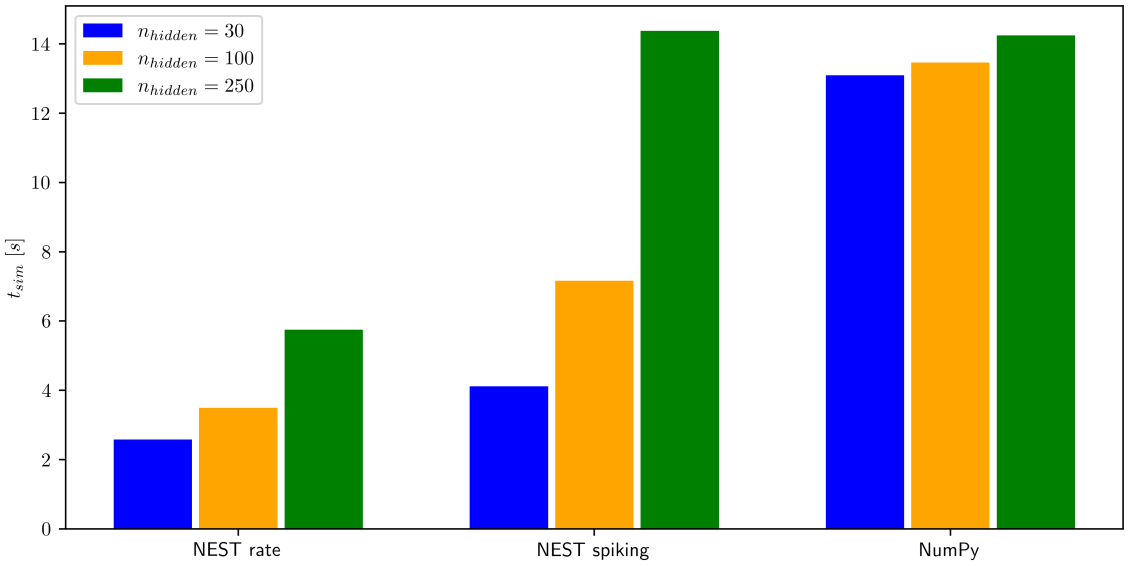

Fig. 3.7: Benchmark of the three implementations under different network sizes. Networks of $[9, n_{hidden}, 3]$ neurons per layer were instantiated with the same synaptic weights and trained for a single epoch of 10 stimulus presentations of $50ms$ each. $n_{hidden} = 30$ was chosen as a baseline, as it is the default throughout all simulations on the Bars dataset.

than the event-based communication performed by NEST.

Not only is the spiking variant of this model slower than the rate version, it also scales worse with network size. Simulation time between 100 and 250 hidden layer neurons doubled, compared to an increase of 1.6 for the rate network. The Difference between the two was even greater when simulating on an office-grade processor (*Intel Core i5-9300H* @ $2.40GHz$, results not shown). Several insights about the comparatively poor performance can be deduced from a first approximation: The most likely causes for increased compute speed are the communication of events and the synaptic plasticity rules. Updates to the neuron state are unlikely to be responsible for the worse performance, as both neuron models are modelled almost identically. These assumptions were tested experimentally.

To assess the impact of synaptic updates on computation time, both variants were simulated once with plastic, and once with static synapses. The simulation environment is set up to model synaptic populations with zero-valued learning rates as non-plastic synapses (`static_synapse` and `rate_connection_delayed` respectively). Thus, by setting learning rates to zero, it was possible to simulate an entire network without spending any time on synaptic updates. Results of this experiment are shown in Fig. 3.8.

As expected, synaptic updates in the spiking network are responsible for a much larger proportion of total simulation time than in the rate network. A much less anticipated result was that spiking networks are considerably slower even when plasticity is turned off. This is surprising, as neuron models are almost identical except for some added complexity in the spike generation process. This added complexity includes drawing from a Poisson process, which might

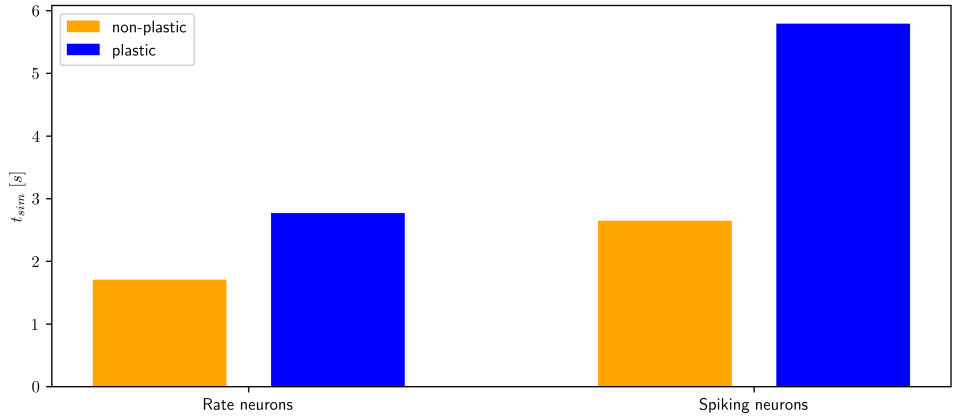

Fig. 3.8: Benchmark of both NEST implementations with plastic and non-plastic synapse types. Deep networks of $300-200-100-10$ pyramidal neurons per layer were stimulated with 5 samples of random input $\in \{0,1\}$ for $10ms$ each. synaptic weights were initialized between $\{-0.1, 0.1\}$ to avoid overstimulation of individual neurons. In the plastic paradigm, all synapses except for feedback weights w^{down} were plastic with very low learning rates $\eta = 10^{-10}$.

be time-costly depending on the underlying implementation. Another possible reason might be added complexity associated with SpikeEvents in general, which update some postsynaptic variables not employed for this model. Further work is required to more rigorously determine the reasons for this poor performance.

To investigate the degree to which synaptic plasticity and spike transmission in general contribute to computational cost, two more experiments were conducted. Training durations under different values for the scaling parameter ψ, as well as with different numbers of threads were recorded. Results are shown in Fig. 3.9.

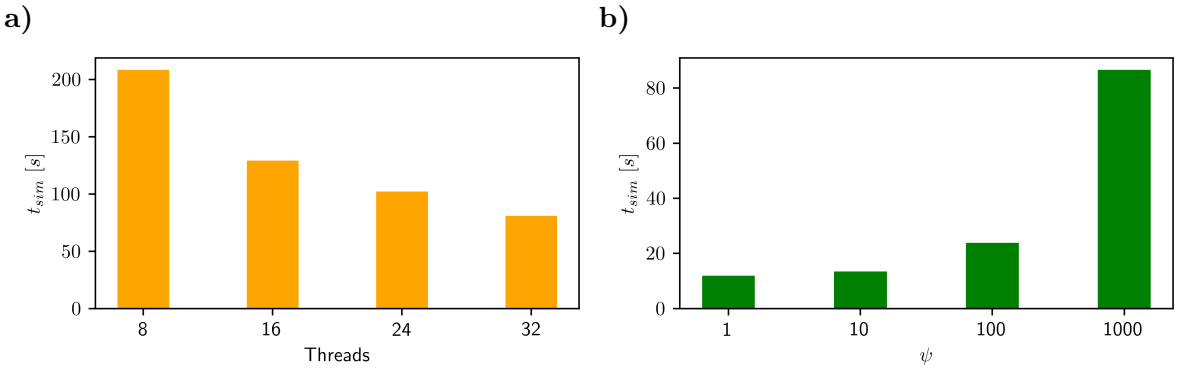

Fig. 3.9: Benchmarks for the spiking implementation. **a)** Simulation on the MNIST dataset for a network of $784-300-100-10$ neurons and $\psi = 250$ on different numbers of threads. 10 samples were presented for $50ms$ each, and weights were drawn randomly from a uniform distribution $\{-0.1, 0.1\}$. **b)** Training of a default network on the Bars dataset using different values of the scaling parameter ψ. All simulations use 8 threads.

Simulating a large network on an increasing number of threads highlights the diminishing re-

turns gained by spreading out the simulation of NEST neurons. While initial speedup is high, at some point the benefit of parallelizing neuron updates is counteracted by the need to communicate more events across threads. It is to be expected that for even higher parallelization simulation time will begin to increase for this network size.
The second figure shows that reducing ψ much lower will likewise lead to diminishing returns. On the other hand, increasing it in the hopes of improving learning performance comes at a stark cost to simulation time. These results should inform future experiments on increasing efficiency through parametrization.

3.8 Pre-training

One of the two major criticisms of the network noted in (Whittington and Bogacz, 2019) is the requirement for pre-training (cf. Supplementary Table S1). By this, the authors mean the initialization to the self-predicting state from which most simulations are started. The original paper implicitly considers three different learning configurations: In the first one, the network starts from the self-predicting state and only feedforward weights are plastic (cf. Sec. 3.2). In the second one, the network starts from random weights and $Intn \rightarrow Pyr$ synapses are plastic, so they can minimize FB error. The third variant, in which feedback $Pyr \rightarrow Pyr$ weights are plastic, is not considered here. An experiment was conducted comparing performance of the first two variants while training on the Bars dataset. These experiments showed that training was marginally slower, but led to identical loss (results not shown). While initializing the network to a self-predicting state does give it a slight 'head-start', this is by no means a condition for learning.
Furthermore, training the network towards the self-predicting state does not require any kind of structured input, let alone targets for activation. The network is driven towards this state purely by noise injection at the input layer. As background noise is trivial to generate (perhaps unavoidable) for any cortical circuit, the self-predicting state might be the default rather than the exception. For these reasons, I do not consider pre-training to be an issue that interferes with the model's biological plausibility.

3.9 Behavioral timescale learning

As a final experiment, the extent to which the network can handle learning on biological timescales was investigated. One criticism occasionally aimed at Backprop is the requirement for instructive signals to be available immediately (Bartunov et al., 2018). The assumption is, that an agent in the real world would select an action, and be informed about the consequences only after some delay. Learning algorithms should therefore be capable of handling delayed instructive signals.
Furthermore, all membrane potentials and synaptic weight derivatives of the dendritic error

network are reset after each stimulus presentation. This procedure ensures that residuals from the previous run do not interfere with learning of a subsequent stimulus. It was confirmed experimentally that networks fail to learn when this reset is not performed after every training sample (results not shown).
Two additions were made to the model to confirm that it is capable of learning without these constraints. First, the target activation was delayed to be first injected 5*ms* after the stimulus. This serves to ensure that that learning does not rely on simultaneous presentation of stimulus and target. Secondly, instead of manually resetting membrane potentials, the network was allowed to relax after each training sample. During this relaxation period (termed *soft reset*), the network is simulated for 15*ms* without any current injections. A training comparison between a vanilla network and these two additions is shown in Fig. 3.10.

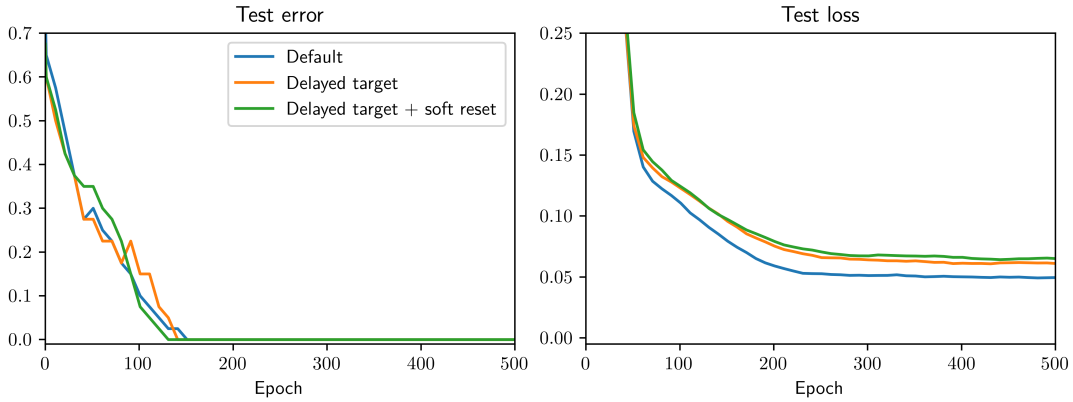

Fig. 3.10: Comparison of learning under minimal external control. **Blue:** default parametrization for the Bars dataset. **Orange:** during the first 5*ms* of a training pattern, no target is provided. Afterwards training continues as usual for the remaining 45*ms*. **Green:** Additionally, the network is not manually reset after each training sample, but simulated for another 15*ms* without stimulation. Increased loss of the delayed target paradigms might be explained by the shorter effective training time per stimulus.

All paradigms lead to equally fast learning of the Bars dataset, with delayed target presentation causing a slightly higher test loss. These results show that constraints like this have only miniscule impact on learning performance of the dendritic error network. Thus, a cortical network of this kind can be expected to be indifferent to idle time in which it is only driven by white noise. Likewise, incoming sensory information in the self-predicting state does not cause plasticity which would drive weights away from what was previously learned. Only when a target is presented to the output layer will weights adapt. This insight shows that the network requires even less external control, which might be of use for improving its efficiency **TODO: ref outlook** . More importantly, with the need to manually reset membrane potentials, another biologically implausible mechanism can be omitted from simulations of this network. It should be noted that this experiment makes the assumption that the brain is either capable of retaining an input sequence until feedback is available, or otherwise 'replay' the pattern at a later point. While such mechanisms are much less elegant than trace-based solutions for

delayed reward signalling (Bellec et al., 2020), the brain has been found capable of such replays (Marblestone et al., 2016).

Chapter 4

Discussion

4.1 Contribution

In this project the capabilities and limitations of the dendritic error network and its underlying plasticity rule were further tested. While sensitive to certain parameter changes, the network was shown to be exceptionally capable of handling various constraints that are imposed on biological neural networks. Furthermore, the performed experiments can be interpreted as dispelling some criticisms aimed at the model's biological plausibility. Here we summarize many of the major questions about the biological plausibility of Backprop from the relevant literature.

Evaluation of biological plausibility

The original dendritic error network by design solves several biologically implausible mechanisms of Backprop. It ***locally computes prediction errors*** and encodes them within membrane potentials of pyramidal neuron apical dendrites. Furthermore, it provides two separate solutions to the ***weight transport problem***: First, it is capable of learning through feedback alignment, as was used in all present simulations. Secondly, experiments employing steady-state-approximations have been successful in training feedback weights through variants of the dendritic error rule. Finally, the network relies strictly on ***Local plasticity*** (Whittington and Bogacz, 2017), and models a (somewhat limited) variability in cell types (Bartunov et al., 2018).

The present work further improves on the neuron model by showing that spike-based communication does not interfere with the dendritic plasticity rule, or the intricate balance of excitation and inhibition demanded by the network. We also show that the network can be trained with absolutely ***minimal external control*** (Whittington and Bogacz, 2017). The network requires no external interference such as manual resets or phased plasticity, and can handle background noise in between sample presentations. Exploratory experiments were conducted in support of the hypothesis that the plasticity rule is capable of credit assignment

when the network conforms strictly to ***Dale's law*** (Bartunov et al., 2018). In related experiments, the network proved very capable of Backprop-like learning when constrained by a more ***plausible architecture*** (Whittington and Bogacz, 2017) in which neuronal populations were connected imperfectly. Finally, the criticism that the network requires pre-training (Whittington and Bogacz, 2019) was found to be largely immaterial. The sum of these observations arguably makes the dendritic error model one of the most biologically plausible approximations of Backprop yet. In spite of these advances, several critical limitations remain.

4.2 Limitations

Some general concerns regarding Backprop were deliberately not addressed in this thesis, but should be noted nevertheless. It still remains unclear how an agent would come by the labels with which it might perform this type of Backprop approximation (Bengio et al., 2015). For this reason, some researchers question whether brains engage in any kind of supervised learning at all (Magee and Grienberger, 2020).
No attempts have yet been made to train the model on anything other than static inputs presented for $10 - 500ms$. Therefore, it remains to be seen whether the model is capable of handling the kind of temporal variations or sequences of patterns expected to stimulate the cortex.

Nudging signals

The requirement for one-to-one connections between pyramidal- and interneurons could not be overcome in this thesis. Thus, one of the major criticisms from (Whittington and Bogacz, 2019) remains unaccounted for. It should also be noted that these nudging connections transmit somatic activation without any kind of nonlinearity or delay, further making them biologically questionable. This property is likely the largest limitation of the model, and further work is required if it is to be addressed. While most general criticisms of Backprop do not apply, the model in its current state does not plausibly conform to cortical connectivity by this requirement alone.

Response to unpredicted stimuli

One of the predictions about cortical activity made by predictive coding is an increased network activity in response to sensory input that violates expectations. A diverse set of studies has since reported behaviour consistent with this in various primate cortical neurons (see Table 1 in (Bastos et al., 2012) for a review). As the dendritic error network encodes errors in dendritic potentials instead of neuron activations, experiments proved that it does not exhibit this property in any of its populations. In fact, overall network activity in response to a stimulus seems to increase after training. In this way, the model conflicts with empirical data

on cortical activity.

Spike frequencies

As discussed in Section 3.4, the network in its current state is unable to learn efficiently for low values of ψ. As a result, the implementation demands physiologically impossible spike frequencies from both pyramidal- and interneurons. While increasing membrane capacitances did relax this constraint somewhat, it in turn requires longer presentation times per stimulus. Further work is required to determine if the network is capable of learning when spike frequencies are as low as reported for cortical neurons.

Benchmark datasets

Training on a benchmark dataset would have been very desirable for comparing the spiking implementation to previous iterations of the dendritic error network. Yet as noted previously, the full network dynamics are prohibitively expensive for simulations of large networks. Extrapolating the results from Fig. 3.9 shows that training on the MNIST dataset is currently unfeasible. A full training with 5.000.000 sample presentations (cf. Haider et al. (2021)) would require over 1 year on 32 threads (excluding testing and validation) with the current configuration of parameters.
Experiments with downscaled images and smaller network sizes were conducted. Yet training for these still took on the order of several days, making the search for suitable parameters very costly. While I am optimistic about the network's capability in general, no parametrization was found in time under which the network was capable of learning MNIST.
The challenge of identifying adequate parameters was hindered by training speed in multiple experiments. As the implementation of the spiking neuron model took much longer than initially anticipated, time was a limiting factor for all present experiments. Particularly the simulations described in Sections 3.4, 3.3 and 3.6 should be repeated with much more varied sets of parameters. I expect results throughout to improve from this, and insights to be applicable to experiments on more complex learning tasks. Thus, computational efficiency remains a serious drawback of this model, and has been a major limiting factor for this thesis.

4.3 Future directions

Computational efficiency

The high computational demands of the network were first reported in the original paper. They were largely alleviated through steady-state approximations and the addition of Latent equilibrium. The present SNN implementation reintroduces this issue, and regrettably exacerbates it considerably. Some improvements can be expected by lowering stimulus presentation time and spike frequencies, or improving the plasticity rule. Yet to a degree, decreased speed

is an inadvertible price to be paid for a more exact modelling of neuronal processes. Therefore, any attempt at introducing new properties of biological neurons must be expected to further increase computational demands. Given the (still very high) level of abstraction of the developed model paired with its poor speed, this perspective is slightly concerning. Therefore, the model requires rigorous optimization.
Some initial directions for this are provided by the benchmarks performed in this study. The spike-based plasticity rule for example is highly costly. One possible optimization was already provided by (Stapmanns et al., 2021). In the paper, an alternative variant for the dendritic plasticity rule is discussed. Instead of a strictly event-based or time-based update rule, a hybrid algorithm called "Event-based update with compression" is presented. This variant tolerates an increased number of synapse updates, but therefore removes redundant computations. In initial tests, it proved particularly advantageous for networks in which neurons had a large in-degree. Therefore, this alternative integration scheme can be expected to perform well for training in the larger networks demanded by more complex datasets. Regrettably, it was not available in time for this thesis, so potential gains remain speculative.
Another possible improvement is approximating the plasticity rule with the instantaneous error at the time of a spike. This would eliminate the requirement for both frequent updates, and for storing and reading a history of dendritic error. Thus, a network employing this simplified plasticity rule would be much less computationally costly. As shown in Fig. 2.3, error terms in LE networks relax after only a few simulation steps. Therefore, under the condition that only static inputs are considered, this crude approximation is expected to perform fairly well.
The neuron model should likewise be investigated for potential improvements in terms of efficiency. Modeling interneurons without an apical compartment might yield some improvements (although initial experiments have dampened expectations for this). It is also possible, that the network does not require integration timesteps as low as $0.1ms$, which has not been investigated yet.
Finally, a lot could likely be gained by further tuning the network's hyperparameters. Network relaxation time, stimulus presentation time, spiking frequencies and learning rates form a complex interplay in this model which has not yet been fully explored. It is therefore to be expected that further optimization of these might yield networks that are both more powerful and more efficient.

Neuromorphic hardware

A different approach which likely would vastly improve simulation speed is a full re-implementation of the model on neuromorphic hardware. This network fits the self-described niche of such systems almost perfectly; It employs strictly local plasticity rules, its nodes use leaky membrane dynamics and communicate through binary spikes. By a rough estimation, even the first generation of Intel's Loihi chips (Davies et al., 2018) should be capable of simulating this neuron model. The chip is capable of modelling multiple dendritic trees per neuron, and the

learning engine appears capable of Urbanczik-Senn-like plasticity[1]. Of course, Loihi is only one of many neuromorphic systems. Another popular system is *BrainScaleS-2*, which appears to be spearheading the field with regard to simulating segregated dendrites (Kaiser et al., 2022). Regardless of the exact system used, neuromorphic hardware promises to reduce the high computation time which currently obstructs further research into the model.

Neuron model

Two properties that are part of most spiking neuron models, but have not been investigated here, are membrane reset and refractory periods. Combined, these modifications would change the spike generation process to that of a stochastic LIF neuron. Such neuron models have previously performed well for modelling sensory representations in the cortex (Pillow et al., 2008). Another neuron property considered in that study is *spike-frequency-adaptation*. Neurons with this mechanism increase their threshold potential in response to previous activity. Such adaptability has been observed in $\sim 20\%$ of neurons in the mouse visual cortex (AllenInstitute, 2018), and has been shown to significantly improve performance in recurrent SNN (Bellec et al., 2018, 2020). These three changes together would significantly improve correspondence of the neuron model to physiological data, while potentially also improving their computational power.
Another point which has not yet been reviewed in terms of biological plausibility is the prospective firing rate in LE neurons. Haider et al. claim that it "represents a known, though often neglected feature of (single) biological neurons" (Haider et al., 2021). They partly base this assessment on the fact that neurons show an increased sensitivity to coincident spikes through Na channel responses (Platkiewicz and Brette, 2011). Further work is required to determine wether prospective activations - particularly as a basis for spike probabilities - appropriately model such processes.

Plasticity rule

The model of the dendritic trees described here is very rudimentary, which has implications for the plasticity mechanism. Reviewing the literature yielded no arguments that the Urbanczik-Senn plasticity requires any biologically implausible mechanisms. Yet given the extensive amount of data in support of STDP, the burden of proof is on the dendritic plasticity rule to override this dogma. This includes finding brain networks which can be modeled by using it, as have been found for STDP. A fruitful approach might be to investigate STDP in multi-compartment models which simulate dendritic spikes and plateau potentials in search of similar plasticity dynamics. Exciting advances in this direction have recently been reported (Bono and Clopath, 2017; Schiess et al., 2016; Magee and Grienberger, 2020), and could potentially be integrated into the present neuron model.

[1]It is possible that the plasticity rule would need to be approximated somewhat for Loihi 1. The publicly available information about the follow-up chip Loihi 2 (Davies, 2021) is still somewhat sparse, but it claims to support a much more diverse set of learning rules.

Cortical circuitry

The circuitry surrounding the dendritic error network is very simplistic compared to the intricate networks of cortical microcircuits - not to mention its numerous connections to other parts of the brain. In this regard, the network still holds much room for improvement. Furthermore, a seminal model for how the cortical microcircuit might be able to perform predictive coding (Bastos et al., 2012) appears to conflict with the dendritic error model. While the resulting network has not yet been computationally modeled, it is backed by a vast amount of empirical data and regarded highly in the literature. One important hypothesis made by it, is that prediction errors are encoded in separate neuronal populations, rather than dendritic compartments. This claim is shared by other works (Hertäg and Clopath, 2022; Whittington and Bogacz, 2017), and further work is required to find out if the two hypotheses ca be reconciled. A first step towards an integration of the dendritic error model and the cortical circuitry is provided in this thesis. We show that the plasticity rule is capable of assigning credit indirectly when dale's law is upheld via additional interneuron populations. Extending this principle to all sets of synapses in the network would introduce novel interneuron populations demanding to be identified. The extent to which the resulting network is compatible with cortical circuitry could prove valuable for further judging the plausibility of the dendritic error model.

4.4 Conclusion

This project further establishes the dendritic error network as one of the most biologically plausible mechanisms for approximating the Backpropagation of errors algorithm. As hypothesized, the spiking variant of the Urbanczik-Senn plasticity is capable of performing well in a much more demanding setting. The model furthermore proved to be largely unhindered by the vast majority of biological constraints which were imposed on it. Particularly constraints on connectivity and synaptic polarity were shown to impede learning to only minor degrees. The model overcomes all but a few of the general arguments for claiming that the Backpropagation algorithm could not plausibly be implemented by networks of biological neurons. Only when investigating the correspondence to cortical microcircuits more closely does the network exhibit serious limitations.

The predictive coding hypothesis has had a substantial impact on the recent developments in cognitive science. While other implementations of predictive coding exist, this one stands out as one of the most promising models. Finding a biologically plausible mechanism that replaces (or alternatively explains) the inter-layer nudging signals would arguably elevate it to a prime contender for explaining supervised learning in the cortex. Nevertheless, such optimism must be paired with restraint given the vast complexity of the cortex - and associated areas - which a general model would have to encompass. Either way, further research into the dendritic error network is likely to help us better understand the exciting capabilities of the human brain.

Chapter 5

Appendix

5.1 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:

$$C_m^{dend} \dot{v}_j^{dend} = -g_l^{dend}\ v_j^{dend} + \sum_i W_{ji}\ \langle n_i \rangle \tag{5.1}$$

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:

$$0 = -g_l^{dend}\ v_j^{dend} + \sum_i W_{ji}\ r_i\ \Delta t \tag{5.2}$$

$$g_l^{dend}\ v_j^{dend} = W_{ji}\ r_i\ \Delta t \tag{5.3}$$

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 fulfills this equality in the steady state, the terms drop out from the above equation. Thus, the correct parametrization for the dendritic leakage conductance remains:

$$g_l^{dend} = \Delta t \tag{5.4}$$

It was shown experimentally that for high values of ψ, this parametrization 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 behavior 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.

5.2 Plasticity in feedback connections

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

$$\dot{w}_l^{down} = \eta_l^{down} \left(\phi(u_l^P) - \phi(w_l^{down} r_{l+1}^P)\right) \phi(u_{l+1}^P)^T \tag{5.5}$$

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. The separation of pyramidal neuron apical dendrites into a proximal and a distal tree is well documented (Ishizuka et al., 1995). Likewise, the fact that these different apical compartments are innervated by separate presynaptic populations is established (Larkum et al., 2018). A difference between plasticity mechanisms for synapses arriving at these two integration zones is likewise plausible, as vastly different membrane dynamics and ion concentrations have been measured between them (Ishizuka et al., 1995; Larkum, 2022). A more sophisticated model of the apical tree and its plasticity could therefore 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 initial tests. Making these feedback 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, which typically set learning rates of these connections to 0 except for a few experiments employing steady-state approximations. Note that under these conditions the network effectively implements a type of Feedback alignment (Lillicrap et al., 2014). Further work is required to identify why plasticity in these synapses seemed to break learning in the present model.

5.3 Electronic supplementary material

Attached to this thesis is a thumb drive containing the electronic supplementary material, which will briefly be described here. The base directory contains a variant of the NEST simulator, which is published under the GPLv2+ license. The NEST version included here was last pulled from the public repository on December 20, 2022 and therefore lies somewhere between NEST 3.3 and 3.4. Under the `models` directory, the neuron (`pp_cond_exp_mc_pyr`, `rate_neuron_pyr`) and synapse models (`pyr_synapse`, `pyr_synapse_rate`) are located with corresponding `.cpp` and `.h` files. Furthermore, the `pyr_archiving_node` responsible for storage and readout of dendritic error, is located in the `nestkernel` directory. Other changes to the simulator largely serve to embed these files and make them accessible from the PyNEST API. The easiest way to compile the NEST simulator (which is required to run the model) is through conda. The `requirements.txt` file in the base directory specifies the required packages. After creating a new environment with

```
$ conda create -n <environment-name> --file requirements.txt
```

a guide from the NEST documentation should simplify setup of the simulator.
The project directory `dendritic_error_network` (roughly) follows the "Good research handbook" (Mineault and Community, 2021):

- **data** contains all figures included in this thesis, as well as some additional ones.
- **results** contains all relevant simulation results. Parameter studies have their own subfolders, titled with the prefix `par_study_`. Each simulation stores its full parameter set (`params.json`), initial weights (`init_weights.json`), final weights (`weights.json`) and training progress (`progress.json`). Intermediate weights are stored every few epochs during training in the `data` subdirectory. If plots were created to validate training progress, they are found in the `plots` folder.
- **experimental_configs** contains JSON files to parameterize experiments. In these files, only the subset of parameters needs to be specified which deviates from the default (cf. Supplementary table S2).
- **scripts** contains scripts to be executed. The vast majority of them were written to validate individual assumptions made in this thesis. Of interest to perform experiments are `train_network.py` and `parameter_study.py`, which both support the `--help` parameter and will therefore not be detailed here.
- **src** contains a python module of reusable code (which should auto-install alongside the conda-environment). This includes networks, datasets, default parameters and some utilities.

- **tests** contains a test-suite which was used to validate the NEST implementation and fine-tune parameters. As it was only strictly necessary during the initial stages, it has not been updated throughout. Therefore most tests currently fail.

5.4 Supplementary Figures

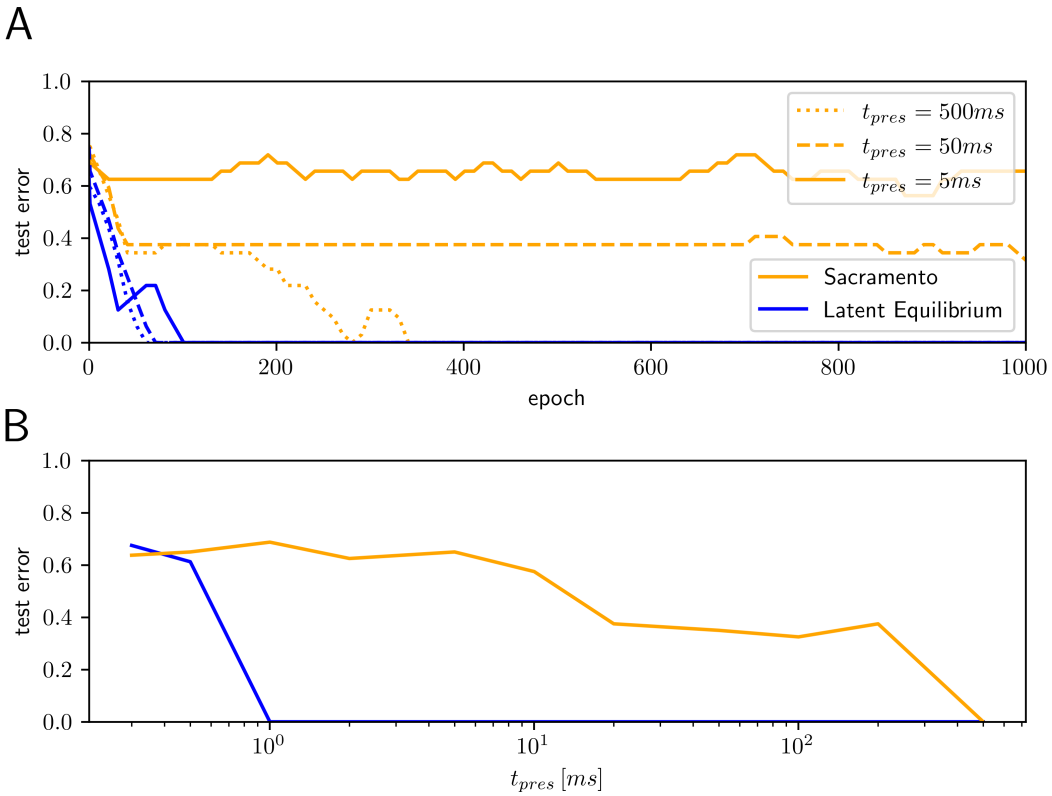

Fig. S1: Replication of Fig. 3.2 using the NumPy network. This varaint is a slightly modified version of the python code from (Haider et al., 2021). Resulting performance matches the original results closely, showing that this version can serve as a baseline for comparing performance of the NEST implementation 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.

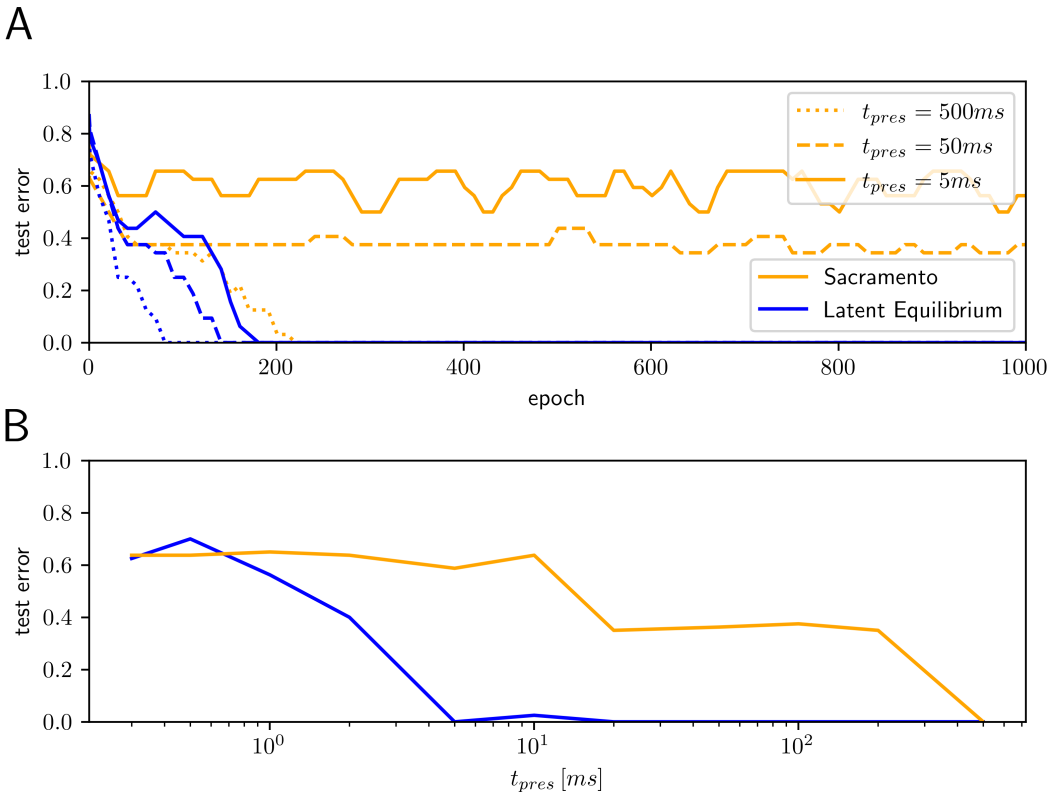

Fig. S2: Replication of Fig. 3.2 using networks of rate neurons in the NEST simulator. A notable difference to the python implementation in Fig. S1 is, that this version does not handle very low presentation times as well. This can likely be traced back to the synaptic delay enforced by NEST, which imposes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

5.5 Supplementary Tables

	Temporal-error model		Explicit-error model	
	Contrastive learning	Continuous update	Predictive coding	Dendritic error
Control signal	Required	Required	Not required	Not required
Connectivity	Unconstrained	Unconstrained	Constrained	Constrained
Propagation time	L-1	L-1	2L-1	L-1
Pre-training	Not required	Not required	Not required	Required
Error encoded in	Difference in activity between separate phases	Rate of change of activity	Activity of specialised neurons	Apical dendrites of pyramidal neurons
Data accounted for	Neural responses and behaviour in a variety of tasks	Typical spike-time-dependent plasticity	Increased neural activity to unpredicted stimuli	Properties of pyramidal neurons
MNIST performance	∼2-3	-	∼1.7	∼1.96

Table S1: Comparison between biologically plausible approximations of Backprop, adapted from (Whittington and Bogacz, 2019). From left to right: Contrastive hebbian learning (O'Reilly, 1996), Contrastive learing with continuous update (Bengio et al., 2017), Predictive coding network (Whittington and Bogacz, 2017), Dendritic error network (Sacramento et al., 2018). All algorithms were selected because they reflect some properties of biological brains, some of which are highlighted in the row "Data accounted for". All of the algorithms need to make concessions for this. In the first four rows, desirable properties are highlighted in green, while undesirable traits are highlighted in red.

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	Delay 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 all somatic 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 S2: Default parameters for the dendritic error model.

Chapter 6

additional notes and questions

6.1 TODOS

- Does the network still learn when neurons have a refractory period?
- talk about feedback plasticity
- 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

6.1.1 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)

6.1.2 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.

AllenInstitute (2018). Allen institute for brain science. allen cell types database, cell feature search. available from: celltypes.brain-map.org/data.

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., Salaj, D., Subramoney, A., Legenstein, R., and Maass, W. (2018). Long short-term memory and learning-to-learn in networks of spiking neurons. *arXiv preprint arXiv:1803.09574*.

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.

Binzegger, T., Douglas, R. J., and Martin, K. A. (2004). A quantitative map of the circuit of cat primary visual cortex. *Journal of Neuroscience*, 24(39):8441–8453.

Bono, J. and Clopath, C. (2017). Modeling somatic and dendritic spike mediated plasticity at the single neuron and network level. *Nature communications*, 8(1):706.

Branco, T., Clark, B. A., and Häusser, M. (2010). Dendritic discrimination of temporal input sequences in cortical neurons. *Science*, 329(5999):1671–1675.

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. (2021). Taking neuromorphic computing to the next level with loihi 2. Technical report, Intel.

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.

Douglas, R. J. and Martin, K. A. (2004). Neuronal circuits of the neocortex. *Annu. Rev. Neurosci.*, 27:419–451.

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*, 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.

Gerfen, C. R., Economo, M. N., and Chandrashekar, J. (2018). Long distance projections of cortical pyramidal neurons. *Journal of neuroscience research*, 96(9):1467–1475.

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.

Gordon, S. and Sten, G. (2010). *Handbook of Brain Microcircuits.* Oxford University Press.

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.

Häusler, S. and Maass, W. (2017). Inhibitory networks orchestrate the self-organization of computational function in cortical microcircuit motifs through stdp. *bioRxiv*, page 228759.

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.

Kaiser, J., Billaudelle, S., Müller, E., Tetzlaff, C., Schemmel, J., and Schmitt, S. (2022). Emulating dendritic computing paradigms on analog neuromorphic hardware. *Neuroscience*, 489:290–300.

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

Kandel, E. R. (1968). Dale's principle and the functional specificity of neurons. In *Psychopharmacology; A Review of Progress, 1957–1967*, pages 385–398. US Government Printing Office Washington, DC.

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.

Larkum, M. E., Zhu, J. J., and Sakmann, B. (1999). A new cellular mechanism for coupling inputs arriving at different cortical layers. *Nature*, 398(6725):338–341.

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.

Leinweber, M., Ward, D. R., Sobczak, J. M., Attinger, A., and Keller, G. B. (2017). A sensorimotor circuit in mouse cortex for visual flow predictions. *Neuron*, 95(6):1420–1432.

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.

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*.

Mineault, P. j. and Community, T. G. R. C. H. (2021). The good research code handbook. *Zenodo*.

Mozafari, M., Ganjtabesh, M., Nowzari-Dalini, A., Thorpe, S. J., and Masquelier, T. (2018). Combining stdp and reward-modulated stdp in deep convolutional spiking neural networks for digit recognition. *arXiv preprint arXiv:1804.00227*, 1.

Niebur, E. (2008). Neuronal cable theory. *Scholarpedia*, 3(5):2674.

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

Pillow, J. W., Shlens, J., Paninski, L., Sher, A., Litke, A. M., Chichilnisky, E., and Simoncelli, E. P. (2008). Spatio-temporal correlations and visual signalling in a complete neuronal population. *Nature*, 454(7207):995–999.

Platkiewicz, J. and Brette, R. (2011). Impact of fast sodium channel inactivation on spike threshold dynamics and synaptic integration. *PLoS computational biology*, 7(5):e1001129.

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.

Potjans, W., Diesmann, M., and Morrison, A. (2011). An imperfect dopaminergic error signal can drive temporal-difference learning. *PLoS computational biology*, 7(5):e1001133.

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.

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.

Schmidhuber, J. (2014). Who invented backpropagation. https://people.idsia.ch/~juergen/who-invented-backpropagation.html. Acessed: 2023-05-21.

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.

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.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Svensson, E., Apergis-Schoute, J., Burnstock, G., Nusbaum, M. P., Parker, D., and Schiöth, H. B. (2019). General principles of neuronal co-transmission: insights from multiple model systems. *Frontiers in neural circuits*, 12:117.

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

Thomson, A. M., West, D. C., Wang, Y., and Bannister, A. P. (2002). Synaptic connections and small circuits involving excitatory and inhibitory neurons in layers 2–5 of adult rat and cat neocortex: triple intracellular recordings and biocytin labelling in vitro. *Cerebral cortex*, 12(9):936–953.

Urban-Ciecko, J. and Barth, A. L. (2016). Somatostatin-expressing neurons in cortical networks. *Nature Reviews Neuroscience*, 17(7):401–409.

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.

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.