Deep learning frameworks for modeling how neural circuits learn

Yuhan Helena Liu

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2024

Reading Committee:

Eric Shea-Brown, Chair

Adrienne Fairhall

J. Nathan Kutz

Stefan Mihalas

Rajesh Rao

Program Authorized to Offer Degree:
Applied Mathematics

University of Washington

Abstract

Deep learning frameworks for modeling how neural circuits learn

Yuhan Helena Liu

Chair of the Supervisory Committee:
Eric Shea-Brown
Department of Applied Mathematics

The brain's prowess in learning and adapting remains an enigma, particularly in its approach to the 'temporal credit assignment' problem. How do neural circuits determine which specific states and connections contribute to future outcomes, and subsequently adjust these for enhanced learning? My thesis addresses this by combining insights from the latest large-scale neuroscience data and recent deep learning theoretical tools.

The first two projects introduce novel learning rules inspired by the Allen Institute's transcriptomics data, which revealed widespread and intricate cell-type-specific interactions among neuromodulatory molecules. This rule enables neurons to propagate credit information efficiently, enhancing learning performance beyond that of biologically plausible predecessors. Extensive computational experiments confirm the significant role of local neuromodulatory signals in learning, offering new perspectives on neural information processing.

My third project assesses the generalization capabilities of bio-plausible learning rules through the lens of deep learning theory, particularly focusing on the curvature of the loss landscape via the loss' Hessian eigenspectrum. Our findings reveal that these rules often settle in high-curvature regions of the loss landscape, indicating suboptimal generalization. This analysis led to a mathematical theorem linking synaptic weight update dynamics to landscape curvature, proposing neuromodulator-driven adjustments as a potential enhancement for learning rule performance.

Given how initial conditions can greatly influence a system's future trajectory, the fourth

project delves into the impact of initial connectivity structures on learning dynamics in neural circuits. By examining various connectivity patterns derived from neuroscience data, including recent electron microscopy data, we analyze how these structures influence learning regimes, implicating metabolic costs and risks of catastrophic forgetting. Our findings suggest that high-rank initializations utilize pre-existing high-dimensional input expansion to facilitate input decoding, leading to minimal changes post-training and increasing the propensity for lazy learning. These specific initializations thus predispose networks toward certain learning behaviors, critically affecting their ability to adapt and generalize.

ACKNOWLEDGMENTS

First, I am extremely grateful to my doctoral advisor, Prof. Eric Shea-Brown. You are central to why I had such a wonderful PhD experience at UW, and I have won the advisor lottery because of you. Thank you for the valuable connections you've helped me establish and for fostering a healthy and supportive research environment at UW CNC/Allen. Because of this, I felt comfortable asking naive questions, which allowed me to learn a great deal from others and grow as a researcher. I also appreciate you helping me to see my blind spots, challenging my thinking, and helping me to become a more well-rounded researcher. I am also very thankful for how you genuinely cared about your students' success and career development, giving me the courage and helping me to build the skills and experiences to continue my academic career. Thank you for tirelessly supporting me throughout this journey and for always being in my corner.

Next, I would like to thank Profs. Stefan Mihalas, Stephen Smith, and Uygar Sumbul, for being my long-term collaborators (and unofficial co-advisors). You were my first collaborators during my PhD. We started working together when I was just starting to learn about both the technical aspects and the social aspects of research. I have learned so much from all of you; thank you for your patience and the valuable lessons. Thank you for the enlightening debates we've had, for guiding me through the experience of optimizing conflicting "value functions" across disciplines, and for teaching me how to deal with frustrations after multiple editorial rejections.

Moreover, I want to thank Prof. Guillaume Lajoie, my long-term collaborator (and unofficial co-advisor), for being my role model. One paragraph is certainly inadequate to summarize how much I've learned from you. Working with you has helped me tremendously to develop a better understanding of how to identify niche but potentially high-impact directions. Your super clear thinking and guidance have also made me better at identifying

the core of problems and the biggest areas of improvement in our projects, not just to avoid potential reviewer complaints but also to truly push our research to the next level. Thank you also for the enjoyable discussions at Mila group events, all the career advice, and your prompt support during the stressful NeurIPS/ICLR rebuttal periods. It was a truly enriching experience interacting with you, your group, and the Mila community.

Additionally, many thanks to my undergraduate research mentees — Hanson Mo, Weixuan Liu, and Xinyue Zhang. Thank you for your meticulous analyses, dedication to consistently advancing our projects, initiatives in addressing problems as they arise, and for also being my teachers. Thank you for this wonderful journey where we grow together, and I am excited to see where you will all end up in life.

I also would like to thank my other co-authors: Blake Richards, Arna Ghosh, Jonathan Cornford, Aristide Baratin, Konrad Kording, James Hazelden, Eli Shlizerman, Chris Cueva, Robert Yang, Dana Mastrovito, Lukasz Kusmierz, and Christof Koch. Thank you for the productive and enjoyable collaborations we've had.

I would also like to thank my other committee members, Profs. Adrienne Fairhall, Nathan Kutz, and Rajesh Rao, for your insightful advice and for establishing a collaborative research environment at UW. I am also grateful to my coworkers, desk mates, group, and cohort. Whether at Lewis Hall or the Health Sciences Center, the community of good-natured, thoughtful, kind, and truly inspiring peers has set a high standard for me. I would also like to extend my gratitude to my colleagues in the NeuroAI community, whom I have met at meetings and conferences; thank you for the enlightening discussions. Thank you also to my home department for supporting such a wonderful and enriching program, and for offering me the invaluable opportunity to serve as the solo instructor for two different undergraduate courses during my PhD.

Lastly, I would like to thank my parents for their unconditional love, for encouraging me to think independently from an early age, and for letting me pursue a more applied discipline (both of them have PhD in theoretical physics). Thank you for your care and for seeing and encouraging good character throughout my life. Thank you for being my good friends and

listening to me endlessly about all kinds of stories during my PhD and major life decisions. Your support, companionship, and encouragement give me the strength on this journey of life.

DEDICATION

to my parents.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Number Page

Chapter 1

INTRODUCTION

1.1 Overview

It remains enigmatic how the brain learns to perform even simple cognitive tasks that involve processing information over just a few seconds, such as when to press a button or recall a sequence of objects being presented. It is widely accepted that learning arises from synaptic plasticity, i.e., the strengthening and weakening of synapses. Various biological learning rules have been developed to describe the activity-dependent re-weighting of synaptic connections. However, *in silico* implementations of these learning rules achieve very limited performance in challenging cognitive tasks.

On the other hand, training artificial neural networks (ANNs) using gradient descent learning via backpropagation [20] has achieved stunning success in solving challenging tasks (e.g., game playing and object recognition), which has tempted many modelers to look for inspiration from it. However, there are many features of biological neural circuits that gradient descent learning via backpropagation does not capture accurately, making it problematic for modeling biological learning. Consequently, there is a surge in interest in developing biologically plausible alternatives to backpropagation and examining how far one can push learning performance [9, 12, 21, 22].

Chapter 2 proposes an efficient and bio-inspired approximation of backpropagation; proof-of-concept simulations demonstrate its advancement from existing biologically plausible rules. Chapter 3 discusses an extension of this work. Chapter 4 characterizes different learning rules and specifically investigates their generalization properties. Chapter 5 studies the role of initialization on learning, particularly how the connectivity structure of ANNs upon initialization influences the learning regimes.

1.2 A simplified view of neural circuits in the brain

The human cortex consists of billions of interconnected neurons, which are the basic processing units of the brain. A neuron in the brain consists of a cell body (soma), dendrites, and an axon. Dendrites receive signals from other neurons via synapses. This information is then integrated in the soma, and if the signal is strong enough, an action potential (spike) is generated and travels down the axon. When it reaches the end of the axon, neurotransmitter release is triggered, passing this information through the synaptic cleft to the dendrites of another neuron.

It should be stressed that the underlying process of neuronal communication is much more complicated than the description above. For one, dendritic integration is highly nonlinear and can yield diverse responses depending on the neuronal morphology. The sophisticated organization of dendritic branches allows the superposition of only specific incoming pulses that exhibit spatiotemporal proximity. Additionally, the biophysical properties of the axon can yield diverse spike waveforms, such as spike frequency adaptation [23]. Moreover, neuromodulators and glial cells (brain cells that are not neurons) can impact neuronal excitability (how likely a neuron will fire an action potential in response to a given input stimulus) [24]. The overwhelming complexity of neural circuits makes it infeasible to capture all biological details in a single computational model. One of the most classical and fundamental neuronal models is the Hodgkin-Huxley model [25]. However, detailed biophysical models pose serious scalability issues: when investigating the behavior of thousands or more interconnected neurons, it is not only computationally expensive but also infeasible due to the unavailability of precise parameter values.

A widely adopted simplification is an artificial neuron model that loosely captures the nonlinear integration of incoming pulses and the generation of an action potential [26]. The output z_j of the j^{th} artificial neuron is produced by combining a nonlinear function $f(\cdot)$ with a weighted sum of its inputs x by synaptic weights w_j such that $z_j = f(w_j x + b)$. Unlike the computationally expensive detailed biophysical models [25] and high-level population dynamic models [27], this model has a resolution in between the two and is a scalable choice for studying how individual neurons interact to give rise to overall network functions. Most

artificial neural network models can be defined as variants of this model.

One successful variant is recurrent neural networks (RNNs), which have been useful for processing time series (before the domination of transformers) and popularized for modeling neural circuits. RNNs consist of recurrently connected units that pass activity signals to each other, which then influence their firings at the next time step. This "bouncing" of activity allows information to propagate over time steps. This makes RNNs a natural choice for modeling neural circuits in this proposed work for the following reasons. First, RNNs are high-dimensional dynamical systems composed of units that loosely mimic biological neurons. Second, recurrent connections are highly prominent in the brain. Third, many cognitive functions performed by the brain require temporal processing (e.g., decision making, language processing). Task-trained RNNs that match neural data can offer insight into how the brain solves computational problems [28–30]. These works investigate neural representations that are formed once a task is learned and have not answered the process of learning itself. We discuss learning in neural circuits next.

1.3 Biological learning rules

Learning is a high-level cognitive function that can arise from a combination of biological mechanisms. One way learning occurs, which has received both theoretical and experimental support, is the re-weighting of synaptic connections in neural circuits. In 1949, Donald Hebb conjectured a simple re-weighting rule, where the strengthening or weakening of a given synaptic connection depends on the coincidence of pre- and postsynaptic neuronal activity [31]. Hebb's rule is one of the most fundamental learning rules and serves as the foundation for many modern learning rules.

Actual synaptic plasticity was first discovered in the 1970s, where repeated synchronous activation of pre- and postsynaptic neurons in the rabbit hippocampus produced synaptic strengthening [32], which remained potentiated for hours. The primary underlying mechanism is thought to rely on the special properties of N-methyl-D-aspartate glutamate receptors (NMDARs), which serve as coincidence detectors for glutamate binding and membrane potential depolarization [33]. Thus, many forms of synaptic plasticity fall under the correlative Hebbian category, including spike timing-dependent plasticity (STDP), which refines Hebbian

plasticity by requiring the precise timing of pre- and postsynaptic spikes [34].

Hebbian learning can autonomously detect low-order input statistical structures [35, 36], but its unsupervised nature makes it difficult to direct the network toward a precise target. Experimental studies later indicated that synaptic plasticity can also be conditioned on the presence of neuromodulators, such as dopamine [37, 38], which are strongly correlated with reward prediction error [1, 39]. This suggests that synaptic updates can be driven by pre- and postsynaptic activity and a third factor that guides plasticity. Therefore, modern learning rules incorporate an additional third factor that can parametrically modulate the magnitude and direction of weight updates [8, 22, 40–42].

A plethora of computational studies have modeled how these mechanisms could support the learning of complex behaviors [4, 9, 22, 43, 44], yet the performance of these models on behavioral tasks still lags significantly compared to machine learning algorithms that rely on the richer backpropagated error signals [12]. This suggests that these models still lack certain processes to fully explain biological learning. Chapter 2 proposes a novel form of learning that achieves closer performance to backpropagation compared to existing biologically plausible rules.

1.4 Potential inspirations from gradient-descent learning via backpropagation

Adjusting network parameters (synaptic weights) to solve a supervised learning task (learn a desired input and output mapping) can be thought of as an optimization problem. Here, the learning objective is to minimize a loss function E, which quantifies the mismatch between the actual and desired output. In artificial neural networks, the error E is usually a differentiable function of the parameters, and parameters can be adjusted iteratively using gradient descent learning. The error gradient indicates the direction in which the parameters should be adjusted for the steepest descent. By adjusting parameters in that direction by a small amount (as gradient descent learning is only a first-order method), the loss E is reduced, reflecting a better match between the actual and desired output. One can compute this error gradient in ANNs using backpropagation [26], a recursive algorithm that uses the chain rule to track dependencies between variables in the network.

Due to the stunning success of gradient descent learning via backpropagation, many

modelers who study biological learning have turned to it for inspiration. Although there is no direct evidence that the brain uses a backprop-like algorithm for learning, a growing body of work has shown that backprop-trained models can account for observed neural responses [45–54]. For example, [55] demonstrated that deep neural networks rival the representation of the primate IT cortex for core visual object recognition. This work revealed that multilayer models trained with backprop to classify objects tend to perform better than other models at matching the representations along the visual ventral stream in primates. Models not trained with backprop do not perform as well as backprop-optimized networks, and their representations do not match those in the inferior temporal cortex as well as the representations discovered by backprop-trained models [56]. To our knowledge, no one in the machine-learning community has been able to train high-performing deep networks on difficult tasks, such as classifying the objects in ImageNet photos, using any algorithms other than backprop.

However, evidence on performance and representational match alone cannot establish that backprop-like mechanisms are employed by the brain. In fact, backpropagation involves many biologically implausible features [56, 57], which makes many argue that it is unsuitable for modeling biological learning. These issues include the weight transport problem [12], where symmetric feedback weights are required for weight updates, and locality [9], where the update of individual weights depends on knowing nearly all other weights in the network. The following chapter explains and focuses on the issue of locality.

Despite its biological implausibility, recent works suggest that there may be more compatibilities between biological learning and learning theory in artificial neural networks than previously appreciated [42], notably how error signals could be propagated in biological neural networks, such as evidence supporting the encoding of reward prediction error via dopamine [39] and top-down regulation of dendritic plateaus that play a crucial role in synaptic plasticity [58]. Moreover, with backprop-trained networks being the only models with performance and representational match, it is likely that current neuroscience models are missing something crucial to achieving what ANNs can do in modeling learning and cognition. This has prompted many to bridge artificial and biological learning by approximating backpropagation as closely as possible using biologically plausible mechanisms [4, 9, 12, 22, 59].

Studies have shown that, in general, algorithms that follow the exact gradient more closely tend to learn in fewer iterations [60, 61]. **Therefore, the goal of this line of research is not to outperform backpropagation, but rather to treat it as a gold standard and see how close one can approach its performance using biologically plausible processes.** We look to continue this line of research and advance biologically plausible learning in Chapter 2.

1.5 Relevance to the neuroscience and machine learning communities

Our study proposes how biological neural networks may leverage the widespread, cell-type-specific modulatory signals uncovered by recent experimental studies at the Allen Institute [3, 62] to solve temporal credit assignment tasks. While multiple studies assign key roles for local/targeted modulatory signaling in synaptic plasticity, learning in the brain is typically studied in a network of neurons connected by synapses alone. Our model reveals a new, joint view of synaptic and modulatory (broadcast) signaling as a *multidigraph* that drives network learning. While cell types are a major theme in modern neuroscience, their roles in computation and learning remain largely unknown. Moreover, the extent to which cell types can be studied as discrete units versus regions in a continuum is a problem of practical and conceptual significance. Our results suggest a novel link between cell types, neuromodulation, and temporal learning. Inspired by recent transcriptomic evidence, we propose a normative theory that explains both the success of discrete approximations and the phenotypic variability around class representatives. We demonstrate its utility on three temporal learning tasks.

The backpropagation algorithm has been a driving force behind the recent, remarkable advances in machine learning. Yet, bottlenecks in energy consumption, training time, and reliance on batch updates present significant scaling issues as the community tackles more complicated tasks. In the training of recurrent neural networks (RNNs), two equivalent algorithms that use different ways to track the gradient scale poorly either with sequence length or network size. In real-time recurrent learning (RTRL) [63], the memory scales as $O(N^3)$ with N being the number of neurons, which is impractical for large networks. In backpropagation through time (BPTT) [64], the algorithm must remember the entire

time horizon, and the memory scales as $O(TN^2)$ with T being the sequence length, which is problematic for credit assignment over long sequences. Proof-of-concept simulations in Chapter 2 show that our learning rule, with $O(N^2)$ memory, can offer a lightweight solution [65]. We also develop an extension in Chapter 3 [66] and investigate learning by leveraging theoretical deep learning tools in Chapters 4 [67] and 5 [68].

Chapter 2

A SOLUTION TO TEMPORAL CREDIT ASSIGNMENT USING CELL-TYPE-SPECIFIC MODULATORY SIGNALS

This chapter has been published in [65] (Copyright 2021 National Academy of Sciences).

2.1 Introduction

Mathematical "gradient backpropagation" algorithms [20, 69] now solve the problem of credit assignment for artificial neural networks effectively enough to have ushered in an era of shockingly powerful artificial intelligence. Nevertheless, their exact implementation on advanced tasks can be extremely costly in terms of computation, storage, and circuit interconnects [70], driving a search for more efficient credit assignment algorithms such as approximate gradient methods [71–74] which, e.g., limit temporal contributions to learning [75] or exploit neuromorphic methods to improve energy efficiency [76, 77]. Neuroscientists meanwhile recognize that exact gradient backpropagation demands precise but nonlocal communication that is implausible in the biological brain, and instead propose approximate learning rules that sidestep the demands. These have shown impressive performance, largely in feedforward networks [12, 22, 59, 78–84] with recent extensions to the more enigmatic case of recurrently connected networks [4, 9]. This said, biological neural networks feature a spectacular array of dynamical and signaling mechanisms whose potential contributions to credit assignment have not yet been considered. Taken together, this creates a remarkable need and opportunity for bio-inspired network learning algorithms to advance both neuroscience and computer science research. Here, we follow this path and present evidence for a previously unrecognized temporal credit assignment mechanism inspired by recent advances in brain genetics.

Prior efforts to address the biology of synaptic credit assignment have focused on Hebbian spike-timing-dependent synaptic plasticity and "top-down" signaling by dopamine [37, 79, 85, 86], a monoamine neuromodulator released from axons that ramify extensively throughout

the brain from small midbrain nuclei. All cellular actions of dopamine are exerted by activation of G protein-coupled receptors (GPCRs), which can strongly modulate the timing-dependence of Hebbian synaptic plasticity [86–88]. While such actions clearly contribute to synaptic credit assignment and recent evidence suggests spatiotemporal sculpting of the dopaminergic signal [1, 89], biologically plausible models based on these principles significantly underperform gradient backpropagation algorithms, let alone the brain, and many gaps in our understanding remain [78, 79].

Transcriptomic studies have now revealed that genes encoding hundreds of other modulatory GPCRs, including those selective for serotonin, norepinephrine, acetylcholine, amino acids and the many neuropeptides, are expressed throughout the brain. Downstream actions of these other GPCRs on nerve membranes and synapses are similar to those of dopamine receptors, suggesting that they, too, could participate in credit assignment. Single-cell RNA-Seq studies now show also, however, that expression of this diverse array of GPCR genes is highly neuron-type-specific [62]. Furthermore, virtually every neuron expresses one or more GPCR-targeting neuropeptide ligands, again in highly neuron-type-specific patterns. These new single-cell transcriptomic data thus suggest the prospect of an interplay between synaptic and local peptidergic modulatory networks that could help to guide credit assignment.

The new genetic results have led us to formulate a theory of network learning which casts neuronal networks in terms of interacting synaptic and modulatory connections, with discrete neuron types as common nodes. To explore this normative theory, we have instantiated the simplified computational model schematized by Figs. 2.1-2.2. The model comprises both dopamine-like top-down and neuropeptide-like local modulatory signaling, shown with a network of arrow-spray glyphs connecting populations of cells, in addition to synaptic transmission via discrete spikes, shown with a network of lines connecting individual cells (Figs. 2.1a). Our model has multiple neuron types distinguished by both their synaptic connectivity and differential expression of modulatory ligand-receptor pairs that regulate Hebbian synaptic plasticity (Fig. 2.1b,c). Our proof-of-concept implementation shows significant improvements over previous literature. The key development is that neurons utilize modulatory networks to actively broadcast their own contribution to the network performance to nearby neurons via cell-type-specific local neuromodulation (Fig. 2.1d and

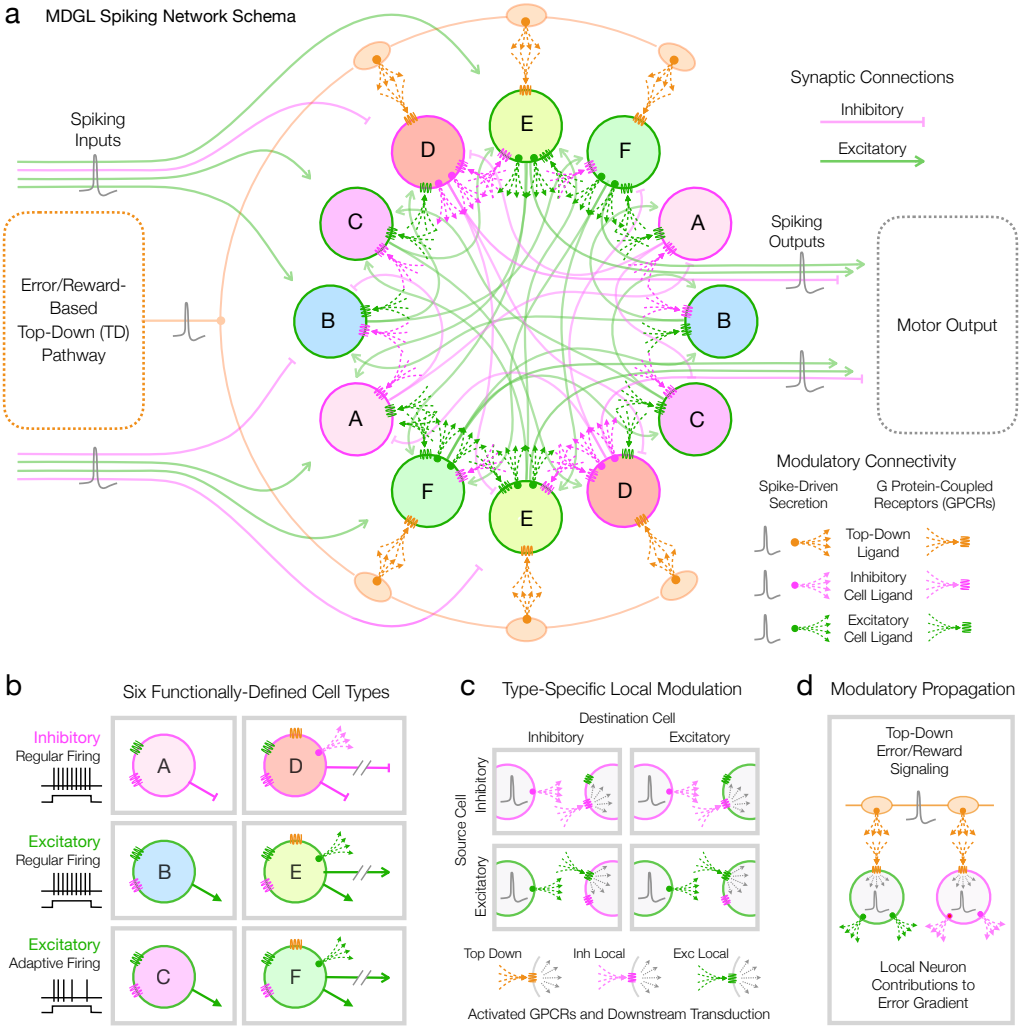

Figure 2.1: Multidigraph learning (MDGL) network schema. a) Six diametrically paired circles (labeled A-F) represent six types of spiking neurons, each defining a population of units on the basis of differential synaptic and modulatory connection and affinity statistics. Inhibitory and excitatory synaptic connections are cartooned here by faint curving lines, while both top-down and local modulatory connections are indicated by arrow-spray glyphs representing secretion of top-down and local modulatory ligands and activation of modulatory GPCRs, all differentially color coded as captioned. Learning tasks are defined by temporal patterns of the indicated spike inputs and outputs as described in main text. b) Six cell types based on excitatory vs. inhibitory synaptic actions, regular vs. adaptive spiking and internal-only vs. output connectivity. Excitatory and inhibitory cells are further distinguished by which neuropeptide-like modulators they secrete, while only output cells are directly responsive to the dopamine-like top-down modulator (TD). c) Cell-type-specific channels of local modulatory signaling established by activity-dependent secretion of two different modulatory ligands and two differentially selective receptors. d) An error/reward-encoding TD signal impacts target neurons and synapses both (1) directly via activity-dependent secretion of TD ligand and (2) indirectly via activity-dependent secretion of local modulatory ligands.

2.2c) – specifically, each cell broadcasts its own direct contribution to the overall task "error" signal. This is a major departure from more global roles for modulators previously proposed, such as carrying error or reward signals. From a neuroscience perspective, our study proposes a new model of cortical learning shaped by the interplay of local modulatory signaling carrying credit assignment information and synaptic transmission, and potentially brings us closer to understanding biological intelligence. From a computer science perspective, our method offers significantly smaller number of interconnects for on-chip neuro-inspired artificial intelligence.

2.2 Results

2.2.1 Overview of multidigraph learning in recurrent spiking neural networks

Gradient descent on the task error (or negative reward) E can iteratively adjust synaptic weights to learn the task. However, computing the error gradient in a recurrent network requires unwrapping the dynamics over time because weights influence future activity in synaptically far-away neurons. The Backpropagation Through Time (BPTT) and Real-Time Recurrent Learning (RTRL) algorithms calculate this error gradient by allowing cell-specific, nonlocal communication among synapses in adjusting their weights; for example, in Fig. 2.1a, the synapse between the uppermost cells labeled C and E would receive information about the many synaptic weights and cell activities downstream of that cell E. They also require either non-causal dependencies (BPTT) or infeasible memory scalability (RTRL) (detailed in Methods, Eq. 2.2-2.17 and Fig. 2.6). Faced with this, a key step in state-of-the-art rate-based [9] and spike-based [4, "e-prop"] biologically plausible learning algorithms is to drop the nonlocal terms so that the activities of only the pre- and post-synaptic neurons would be needed to update the weight of the synapse between them. As illustrated in Fig. 2.2a, the resulting three-factor local learning rules represent this pre- and post-synaptic information as a time-dependent eligibility trace (ET), and combine it with top-down (TD) signals to update the weight Δw of each synapse [8, 38, 42, 90] (detailed further in Methods Figs., 2.5b, 2.6dii; see also [74]).

We propose a novel role for cell-type-based modulatory signals in recovering a key part of

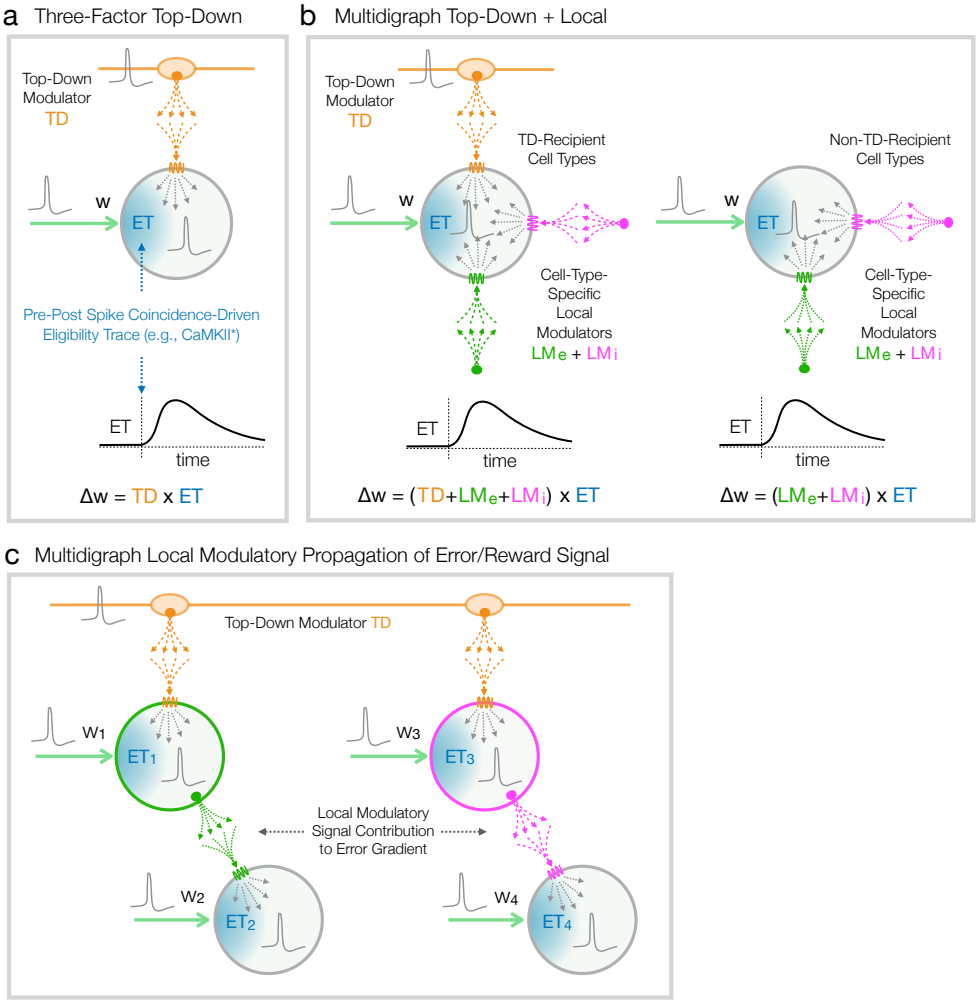

Figure 2.2: Modulator-based neo-Hebbian local learning rules. a) A conventional three-factor local learning rule models action of a "third", top-down (TD) GPCR-activating ligand (e.g. dopamine) that governs synapse re-weighting (Δw) in proportion to temporal coincidence of the two Hebbian factors (presynaptic and postsynaptic activity). Such models generally require a lingering "eligibility trace" (ET) to sustain information about Hebbian coincidence until arrival of the TD signal. b) Embracing new genetic evidence for local GPCR-based modulatory machinery, the MDGL theory introduces additional factors that allow spike-dependent secretion of neuropeptide-like local modulators (LM_e from excitatory neurons and LM_i from inhibitory neurons) to participate in governing synapse re-weighting (Δw) [35]. As indicated here and in Fig. 1, the present MDGL model comprises both directly TD-recipient cells (types D-F, left) and non-TD-recipient cells (types A-C, right). Synapse re-weighting requires combined GPCR activation with a persistent ET for all cell types, but GPCRs are activated on non-TD-recipient cells only by the local modulatory ligands. c) Propagation of TD error/reward signal via spike-dependent secretion of local modulators from both excitatory and inhibitory cell types to cells lacking direct access to TD modulatory signal. For simplicity, this schema represents only the four subscripted synapses/weights, while the full model represents many more synaptic inputs per cell.

the error gradient information that is lost by dropping nonlocal terms in such conventional three-factor rules. We describe this in terms of the update Δw to a synapse $p|q$ from neuron q to neuron p with strength w. To begin, we consider the contributions to the error gradient of those neurons that are one synapse away from the post-synaptic site (i.e., neurons j such that synapse $j|p$ exists) (illustrated in Methods Figure 2.5c). We find that this set of contributions can be realized by activity-dependent signals emitted by neurons j and the ET for the synapse $p|q$ (Eq. 2.19). Intriguingly, this signal is precisely neuron j's contribution to the task error, thereby taking into account the indirect contribution of the synapse $p|q$ to the network performance via neurons j for more accurate synaptic credit assignment.

This initial form, however, still requires the knowledge of cell-specific signals from cells j not participating in the synapse of interest. We further make the key observation that when just the contributions from cells up to two synapses away are considered, those terms only appear under a sum: the mechanism updating the synapse $p|q$ does not need to know the contributions from individual "indirect" neurons j, as their sum suffices. This observation is critical in elucidating a role for diffusive neuromodulatory signalling in carrying this summed, indirect signal and thus serving as an additional factor in synaptic plasticity.

To fully remove cell-specific dependencies in the indirect signal, we further approximate the cell-specific weights w_{jp} that it contains with the *cell-type*-specific terms $w_{\alpha\beta} = \langle w_{jp} \rangle_{j \in \alpha, p \in \beta}$ when postsynaptic cell j belongs to type α and presynaptic cell p belongs to type β. We postulate that $w_{\alpha\beta}$ represents the affinity of GPCRs expressed by cells of type β to peptides secreted by cells of type α (Fig. 2.1c, and Methods Eq. 2.20 and Fig. 2.5d) and this cell-type-specific variable is genetically determined. The local diffusion assumption [91] suggests a further idealization, where this type of signaling is registered only by local synaptic partners and therefore preserves the connectivity structure of w_{jp} (Eq. 2.23). It is also worth noting the rich set of ligand and receptor types with different downstream actions [62] can support that $w_{\alpha\beta}$ is a signed term.

Bringing these together, we have the new learning rule illustrated in Fig. 2.2b. The weight update is given as

$$\Delta w_{pq} \propto \left(\mathrm{TD}_p + \sum_{\alpha \in C} \mathrm{LM}_{\alpha\beta} \right) \times \mathrm{ET}_{pq}$$

$$\mathrm{LM}_{\alpha\beta} = (\text{affinity } w_{\alpha\beta}) \times \sum_{j \in \alpha, p \to j} \underbrace{\mathrm{TD}_j \times (\text{activity } j)}_{\text{modulatory signal } j} \tag{2.1}$$

where neuron j is of type α and neuron p is of type β, $p \to j$ denotes that synapse $j|p$ exists, C denotes the set of neuronal cell types, TD_p denotes the top-down signal received by p, ET_{pq} denotes the eligibility trace for $p|q$, affinity $w_{\alpha\beta}$ denotes the effect of ligands secreted by class α neurons on class β receptors and $\mathrm{LM}_{\alpha\beta}$ denotes the contribution of local modulation to synaptic plasticity that has been ignored so far. Thus, our update rule suggests a set of new modulatory terms that combine with the eligibility trace in order to more accurately assign credit across a network when updating its synapses. Neurons that receive top-down feedback regarding their role on the circuit goals propagate this information to nearby synaptic partners via cell-type-specific local modulatory signals (Figure 2.2c, see also Eq. 2.24 for details). Specifically, the modulatory signal j in Eq. 2.1 is precisely the contribution of cell j to the task error, as measured by the (partial) derivative of the error with respect to cell j's membrane potential. Moreover, this framework proposes that cell-type-specific GPCR affinities allow these local signals to be informative without the need to know precise synaptic weights. The ability to assess the *indirect* impact of neurons on the overall loss via such communication is critical to accurate synaptic re-weighting and improved performance over existing biologically plausible rules as we demonstrate next (Figure 2.3, Supplementary Figure S2).

In summary, we have proposed a new rule for updating a synapse w_{pq}, which we refer to as the multidigraph learning rule or MDGL, where the Hebbian eligibility trace is compounded not only with top-down learning signals – as in modern biologically plausible learning rules [8, 42] – but also with cell-type-specific, diffuse modulatory signals.

2.2.2 Simulation framework for testing multidigraph learning in recurrent spiking neural networks

To test the MDGL formulation, we study its performance in recurrent spiking neural networks (RSNNs) learning well-known tasks involving temporal processing: pattern generation, delayed match to sample, and evidence accumulation. We use two main cell classes, inhibitory (I)

and excitatory (E) cells, and obey experimentally observed constraints (e.g., refractoriness, synaptic delay, connection sparsity). We further endow a fraction of the E cells with threshold adaptation [92]. This mimics the hierarchical structure of cell types [93] through the simple example of two main cell types, one of which has two subtypes (E cells with and without threshold adaptation). The existence/lack of synaptic connections to output neurons further divides each population into two, thus bringing the cell type tally to 6 in our conceptual model (Figure 2.1). Our implementation does not involve rapid and random formation of new synapses after each experience [5], further increasing its biological plausibility.

We compare the learning performance of MDGL (Fig. 2.2b) with the state-of-the-art biologically plausible learning rule (e-prop [4]), Figures 2.2a, S1. As a three-factor rule, e-prop does not involve local cell-type-specific signaling and restricts the update to depend only on pre- and postsynaptic activity as well as a top-down instructive signal. To provide a lower bound on task error, we also compare performance with BPTT (Figure 2.5a), which uses exact error gradients to update weights. These learning rules are further illustrated in Methods Figure 2.5a,b,d.

2.2.3 Multidigraph learning guides temporal credit assignment in benchmark tasks

We first study pattern generation with RSNNs, where the aim is to produce a one-dimensional target output, generated from the sum of five sinusoids, given a fixed Poisson input realization [94]. We change the target output and the Poisson input along with the initial weights for different training runs (Figure S3a), and illustrate the learning curve in Figures 2.3a, S4a across five such runs. We observe that MDGL performs significantly better than e-prop.

Next, to study how RSNNs can learn to process discrete cues that impact delayed rewards, we consider a delayed match to sample task [95]. Here, two cue alternatives are encoded by the presence/absence of input spikes. The RSNN is trained to remember the first cue and learn to compare it with the second cue delivered at a later time (Figure S3b). Figures 2.3b, S4b, S5 display the learning curve for novel inputs. We observe that the same general conclusions as for the pattern generation task hold; introducing cell-type-specific neuromodulation significantly improves learning outcomes.

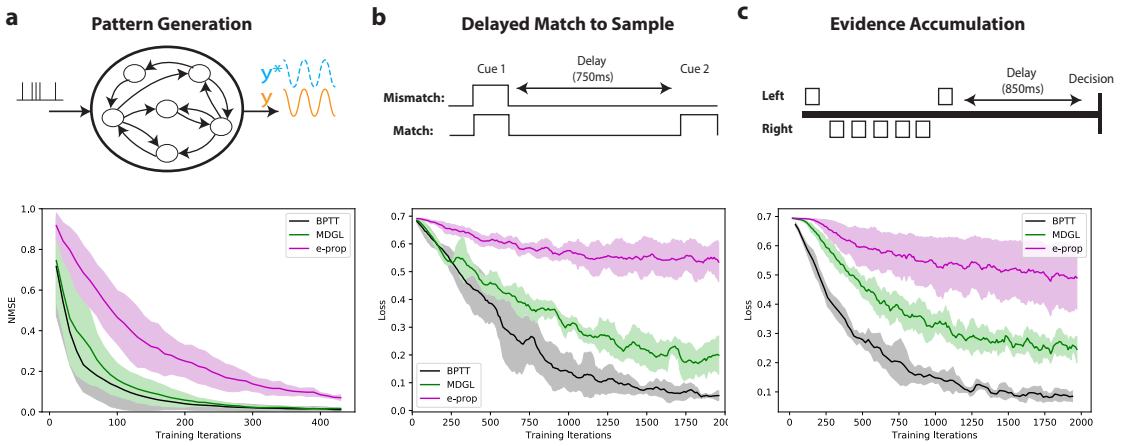

Figure 2.3: Cell-type-specific neuromodulation guides learning across multiple tasks. a) Learning to produce a time-resolved target output pattern. b) A delayed match to sample task, where two cue alternatives are represented by the presence/absence of input spikes. c) An evidence accumulation task [1, 2]. Bottom row: Addition of cell-type-specific modulatory signals improves learning outcomes across tasks. In line with these results, Figure S2 shows that gradients approximated by MDGL are more similar to the exact gradients than those approximated by e-prop.

Solid lines/shaded regions: mean/standard deviation of loss curves across runs (Methods).

Finally, we study an evidence accumulation task [1], which involves integration of several cues to produce the desired output at a later time: a simulated agent moves along a path while encountering a series of sensory cues presented either on the right or left side of a track (Figures 2.3c, S4c, S3c). When it reaches a T-junction, it decides if more cues were received on the left or right. We test our learning rule to see if the addition of diffuse modulatory signals can indeed bring the learning curve closer to BPTT, without relying on rapid and random rewiring [5]. Figure 2.3c demonstrates that the performance trends of the previous two experiments continue to hold. Figure S6 illustrates that without threshold adaptation and recurrent connectivity, the network cannot significantly decrease loss and thus is unable to learn this task. In line with these experiments, gradients approximated by MDGL are more similar to the exact gradients (Figure S2), shedding light on its superior performance. We also observe that MDGL's performance depends only weakly on the hypothesized link (Eq. 2.23) between abstract cell type-based connectivities and modulatory receptor affinities (Figure S7), enabling flexible implementations *in-vivo* and *in-silico*.

We conducted further studies to better quantify how a model's network architectures impact the performance of MDGL relative to other learning rules. First, as depicted in Figures 2.1 and 2.2, recall that only output projecting neurons receive top-down (TD) signals, which is a consequence of gradient-based learning [4], and only these neurons secrete local neuromodulators (i.e., non-zero LM in Eq. 2.1). Consistent with this, we found that MDGL is most advantageous relative to e-prop in cases where relatively small fractions of recurrently connected neurons are output projecting, as we may expect in many biological networks (Figure S8). In these "sparse-output" cases, while many neurons do not receive learning signals in the e-prop formulation, MDGL still allows these neurons to receive such signals via local modulation. The result is more accurate approximation of gradients and more efficient learning. Next, Figure S9 demonstrates the effectiveness of using the cell-type-specific, rather than more precise cell-specific, weights the modulatory signals within the MGDL framework. We find that the cell-type based approximation does degrade performance, but that this effect is relatively minor.

Finally, we note that while our proof-of-concept implementation assumes symmetric feedback weights for the output connections (i.e. the same output projection weight is used during the computation of top-down feedback signal), the random feedback alignment approach [12] or approximating the feedback weights using the same cell-type-based calculations in Eq. 2.23 both offer improved biological plausibility for this single-layer feedforward problem.

2.2.4 Spatiotemporal extent of multidigraph learning

Owing both to extracellular ligand diffusion biophysics and the complex metabolic nature of GPCR-based signal transduction, neuromodulation time scales are generally much longer than those of conventional synaptic transmission [3]. Since our model does not explicitly limit the communication bandwidths of either channels, comparing the frequency content offers an important check for biological plausibility. It also provides a test of our approximation that the summation over cells in Eq. 2.19 acts as a smoothing operation. Figures 2.4a-c, S10 demonstrate that the modulatory input indeed has significantly lower frequency content than the synaptic input, for all three tasks.

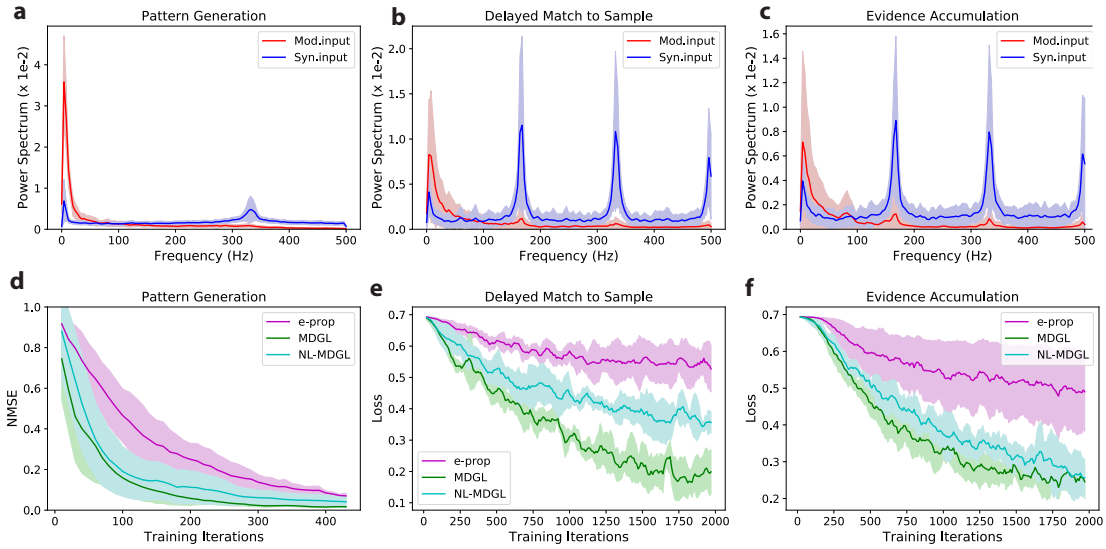

Figure 2.4: Spatiotemporal characteristics of local neuromodulation. a-c) Power spectra of modulatory (Mod.input; total cell-type-specific modulatory signal detected by each cell – Eq. 2.21) and synaptic inputs (Syn.input; total input received through synaptic connections by each cell – Eq. 2.22) are compared after learning for all tasks. Solid lines denote the average and shaded regions show the standard deviation of power spectrum across recurrent cells. Raw input traces are included in Figure S10. d-f) Performance degrades when neighborhood specificity of modulatory signaling (NL-MDGL) is removed so that cell-type-specific modulatory signals diffuse to all cells in the network without attenuation. Learning with spatially non-specific modulation still outperforms that without modulatory signaling (e-prop).

The distance ranges of diffusive modulatory signals remain uncertain [96]. For most of the simulations described here (those according to Eq. 2.23) modulatory signaling was limited to synaptically-coupled pairs (a token of anatomic proximity), representing an idealization of the short-range signaling case. We also examined the opposite extreme case, where modulatory signals extend to all cells (independent of any anatomic proximity), referring to this nonlocal form of MDGL as NL-MDGL. This would make $w_{\alpha\beta}$ a worse approximation to w_{jp}, presumably degrading the quality of the gradient estimate. Indeed, removing the locality of modulatory signals degrades performance while remaining superior to that in the absence of modulatory signaling (Figure 2.4d-f) – suggesting that the biophysics of diffusive modulatory signaling may condition the efficiency of synaptic learning.

2.3 Discussion

Here we have presented a novel multidigraph theory and instantiated simple models based on this theory that explicitly represent diverse neuron types classified by their synaptic and neuromodulatory connections. Simulations based on these simple models show that diverse signaling modes can facilitate credit assignment and enhance learning. A wealth of new genetic data provide strong support for the biological plausibility of this array of signaling modes, and furthermore argue strongly that most or all modern eumetazoans (all multicellular animals except sponges) comprise numbers of cell types and modulatory signals far in excess of those represented in our simulations [96]. We believe therefore that conceiving of neuronal networks as multidigraphs, involving multiple modulatory and synaptic signals, integrated by discrete cell-type nodes, may offer fruitful new paths toward the understanding of synaptic credit assignment in biological neuronal networks. This multidigraph theory may also lead to new, more computationally efficient local learning rules for neural-network-based artificial intelligence.

In addition to established elements of Hebbian plasticity, eligibility traces and reward feedback signals, our normative theory posits important roles for neuronal cell type diversity and local neuromodulatory communication in enabling efficient synaptic credit assignment. In particular, our findings predict that neurons secrete information about top-down feedback signals they receive to nearby neurons in an activity-dependent and cell-type-specific manner using local modulation. As a consequence, levels of local modulatory signals may reflect the learning process. Indeed, our computational experiments imply that the level of modulatory input decreases over training and sharply rises in response to changes in task condition (Figure S11). It is also interesting to note that phylogenomic studies now suggest that peptidergic neuromodulation may evolutionarily predate dopamine signaling [96] and thus may have actually provided the foundation upon which dopaminergic top-down signaling evolved.

The nature of "intermediate" cells [93], whose phenotypes appear to be a mixture of "pure" cell types is a key problem in cell types research. Our findings may explain the existence of such phenotypes from a connectivity perspective: while average connectivities between

types remain relatively constant during training, connectivities of individual cells can deviate significantly from those averages (Figure S12). We hypothesize a link between abstract cell type-based connectivities and modulatory receptor affinities, where the average synaptic connection weights between types are taken to be the cell-type-specific modulatory receptor affinities (Eq. 2.23). How tightly the individual synaptic weights and cell-type-specific receptor affinities should be coupled may be explored in future work. Figure S2 indicates that even with imprecise GPCR affinities, MDGL can still improve gradient approximation while Figure S7 suggests that the effect of imprecise GPCR affinities on the performance is task-dependent.

Learning rules often explicitly minimize a loss function, and the error gradient, if available, tells exactly how much each network parameter should be adjusted to reduce this loss function. Rules that follow this gradient, real time recurrent learning (RTRL) and backpropagation through time (BPTT), are well established, but are not biologically plausible and have unmanageably vast memory storage demands. However, a growing body of studies have demonstrated that learning rules that only partially follow the gradient, while alleviating some of these problems of the exact rules, can still lead to desirable outcomes [61, 97]. An example is the seminal concept of feedback alignment [12], which rivals backpropagation on a variety of tasks even using random feedback weights for credit assignment. In addition, approximations to RTRL have been proposed [9, 71–76] for efficient online learning in RNNs. Our learning rule has $O(N^2)$ complexity, where N is the number of neurons, which is less expensive than SnAp-2 that has a storage cost of $O(N^3)$ [75] (for simplicity, connection sparsity factor is neglected here). It also outperforms biological learning rules with similar complexity scale [4, 9]. Thus, our model further advances approximated gradient-based learning methods and continues the line of research in energy-efficient on-chip learning through spike-based communications [77, 98]. Such efficient approximations of the gradient computation can be especially important as artificial networks become ever larger and are used to tackle ever more complex tasks under both time and energy efficiency constraints.

Examination of the learning capability of MDGL under a broader range of tasks and conditions represents a valuable future avenue. For instance, our preliminary simulations on modulating the delay period in the match to sample task (Figure S13) suggests such

studies can help reveal the reasons underpinning the observed animal behavior [6] as well as limitations of MDGL. In addition, brain cells are extremely diverse [93, 99] with a matching diversity in the expression of peptidergic genes [62]. Further studies can also investigate the interplay of task complexity and cell diversity. A starting point for that could be further dividing inhibitory cells into subtypes with and without threshold adaptation.

Our work suggests that multiple cell-type-specific, diffuse and relatively slow modulatory signals should be considered as possible bases for credit assignment computations. Though inspiration for the present work came primarily from new transcriptomic data on local NP signaling in neocortex [3, 62], it is quite possible that other cell-type-specific neuromodulators could likewise contribute to credit assignment. Many of these alternative agents act, as do NPs, via GPCRs (e.g., the monoamines, amino acids, acetylcholine, endocannabinoids), but our multidigraph template might even apply to other neuronally secreted neuromodulators, such as the neurotrophins and cytokines, that act via different classes of receptor [100, 101]. While experimental tests of such hypotheses have not seemed feasible up until now, emerging methods for genetically addressed measurement of various neuromodulatory signals in specific cell types [62, 102] are now bringing the necessary critical tests within reach (e.g. [103]).

2.4 Methods

Visual summary of learning rules:

An overview of our network model and the mathematical basis of the learning rules used in this work is given in the beginning of Results. Here, we first present an additional, more detailed visual illustration of these learning rules in Fig. 2.5, beginning with the exact gradient update (panel (a)), as for BPTT, and leading from its dramatic truncation in the e-prop rule (panel (b)), to MGDL (panels (c-e)), which partially recovers gradient information lost in this truncation.

Spiking neuron Model: We consider a discrete-time implementation of recurrent spiking neural networks (RSNNs). The model, as shown in Figure 2.6a, denotes the observable states, i.e. spikes, as z_t at time t, and the corresponding hidden states as s_t. For leaky integrate-and-fire (LIF) cells, the state s_t corresponds to membrane potential and the dynamics of

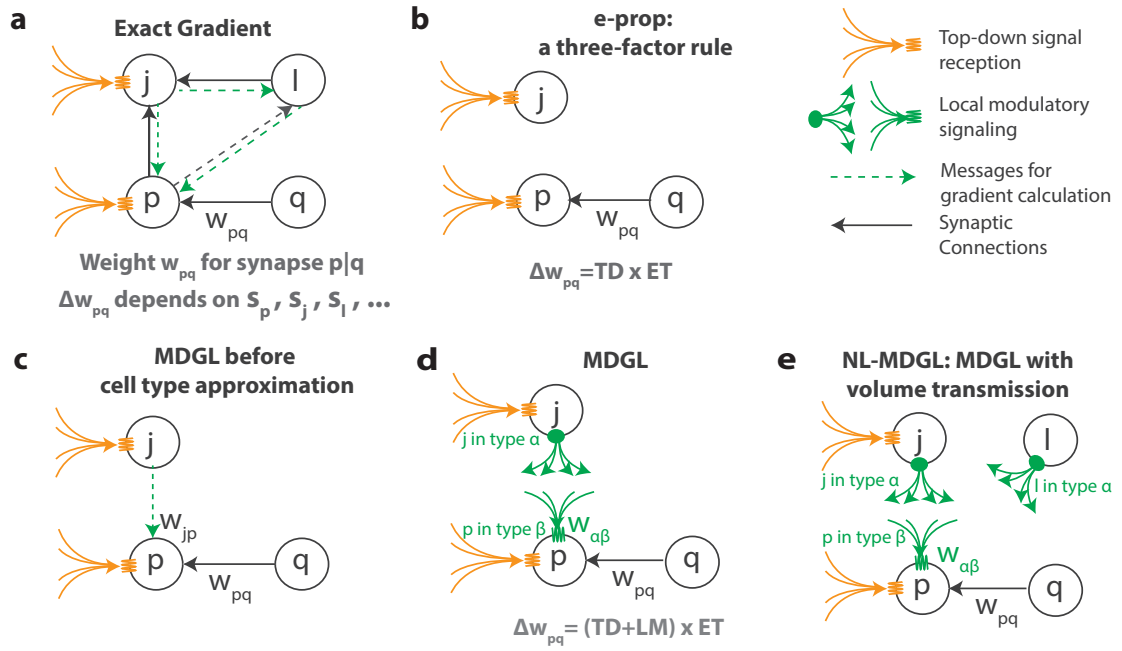

Figure 2.5: Cartoon summary of learning rules explored in this work. a) the exact gradient: updating weight w_{pq}, the synaptic connection strength from presynaptic neuron q to postsynaptic neuron p, involves nonlocal information inaccessible to neural circuits, i.e. the knowledge of activity (e.g., voltage s) for all distant neurons j and l in the network. This is because w_{pq} affects the activities of many other cells through indirect connections, which will then affect the network output at subsequent time steps (Eq. 2.17 in Methods). b) E-prop, a state-of-the-art biologically plausible learning rule, restricts weight update to depend only on pre- and post-synaptic activity and top-down learning signal, as in a three-factor learning rule (Figure 2.2a). c) We allow the weight update to capture dependencies within one connection step, which are omitted in e-prop. The activity of neuron j could be delivered to p through local modulatory signaling. d) For the signaling in c) to be cell-type-specific, as consistent with experimental observation in [3] and biologically plausible mechanisms, we approximate the cell-specific gain with cell-type-specific gain (Eq. 2.23), which leads to our multi-digraph learning rule (MDGL). Effect of this cell-type approximation is explored in Figure S9. e) A nonlocal version of MDGL (NL-MDGL), where modulatory signal diffuses to all cells in the network without attenuation. (See Figure 2.4 and Spatial extent of cell-type-specific modulatory signaling, Methods.)

those states are governed by

$$z_{j,t} = H(s_{j,t} - v_{\text{th}})$$
$$s_{j,t+1} = \eta\, s_{j,t} + (1-\eta)\left(\sum_{l\neq j} w_{jl}\, z_{l,t} + \sum_m w^{\text{IN}}_{jm}\, x_{m,t+1}\right) - z_{j,t}\, v_{\text{th}} \tag{2.2}$$

where $s_{j,t}$ denotes the membrane potential for neuron j at time t, v_{th} denotes the spiking threshold potential, $\eta = e^{-dt/\tau_m}$ denotes the leak factor for simulation time step dt and membrane time constant τ_m, w_{lj} denotes the weight of the synaptic connection from neuron j to l, w^{IN}_{jm} denotes the strength of the connection between the input neuron m and neuron

j, x_t denotes the external input spike at time t and H denotes the Heaviside step function.

Figure 2.6: Computational graph and gradient propagation. a) Schematic illustration of the recurrent neural network used in this study. b) The mathematical dependencies of input x, state s, neuron spikes z and loss function E unwrapped across time. c) The dependencies of state s and neuron spikes z unwrapped across time and cells. d) The computational flow of $\mathrm{d}\, s / \mathrm{d}\, w_{pq}$ is illustrated for (di) exact gradients computed using exact calculation (Eq. 2.17), (dii) e-prop and (diii) our truncation in Eq. 2.18, where dependency within one connection step has been captured. Black arrows denote the computational flow of network states, output and the loss; for instance, the forward arrows from z_t and s_t going to s_{t+1} are due to the neuronal dynamics equation in Eq. 2.2. Green arrows denote the computational flow of $\mathrm{d}\, s / \mathrm{d}\, w_{pq}$ for various learning rules.

Following [4], which implemented adaptive threshold LIF (ALIF) units [92] and observed that this neuron model improves computing capabilities of RSNNs relative to networks with LIF neurons only, we also include ALIF cells in our model. In addition to the membrane potential, ALIF cells have a second hidden variable, b_t, governing the adaptive threshold. The spiking dynamics of both LIF and ALIF cells can be characterized by the following set of equations:

$$
\begin{aligned}
s_{j,t+1} &= \eta\, s_{j,t} + (1-\eta)(\sum_{l\neq j} w_{jl}\, z_{l,t} + \sum_{p} w^{\mathrm{IN}}_{jm}\, x_{m,t+1}) - z_{j,t}\, v_{\mathrm{th}} && (2.3)\\
z_{j,t} &= H(s_{j,t} - A_{j,t}) && (2.4)\\
A_{j,t} &= v_{\mathrm{th}} + \beta b_{j,t} && (2.5)\\
b_{j,t} &= \rho b_{j,t-1} + (1-\rho) z_{j,t-1}, && (2.6)
\end{aligned}
$$

where the voltage dynamics in Eq. 2.3 is the same as Eq. 2.2. A spike is generated when the voltage $s_{j,t}$ exceeds the dynamic threshold $A_{j,t}$. Parameter β controls how much adaptation affects the threshold and state $b_{j,t}$ denotes the variable component of the dynamic threshold. The decay factor ρ is given by e^{-dt/τ_b} for simulation time step dt and adaptation time constant τ_b, which is typically chosen on the behavioral task time scale. For regular LIF neurons without adaptive threshold, one can simply set $\beta = 0$.

Network output and loss function: Dynamics of leaky, graded readout neurons is implemented as

$$
y_{k,t} = \kappa\, y_{k,t-1} + (1-\kappa) \sum_j w^{\mathrm{OUT}}_{kj} z_{j,t} + b^{\mathrm{OUT}}_k, \qquad (2.7)
$$

where w_{kj}^{OUT} denotes the strength of the connection from neuron j to output neuron k, b_k^{OUT} denotes the bias of the k-th output neuron, $\kappa \in (0,1)$ defines the leak and $\kappa = e^{-dt/\tau_{OUT}}$ for output membrane time constant τ_{OUT}.

We quantify how well the network output matches the desired target using error function E:

$$E = \begin{cases} \frac{1}{2}\sum_{k,t}(y_{k,t}^* - y_{k,t})^2, \text{ for regression tasks} \\ -\sum_{k,t}\pi_{k,t}^* log \pi_{k,t}, \text{ for classification tasks} \end{cases} \tag{2.8}$$

where $y_{k,t}^*$ is the time-dependent target, $\pi_{k,t}^*$ is the one-hot encoded target and $\pi_{k,t} = \mathrm{softmax}_k(y_{1,t}, \ldots, y_{N_{OUT},t}) = \exp(y_{k,t})/\sum_{k'}\exp(y_{k',t})$ is the predicted category probability. We provide all simulation and training parameters in Supplementary Note 3.

While the tasks involving time-delayed rewards studied in this manuscript can be labeled as regression and classification tasks due to the nature of the objective function, we note that the theoretical development is general and applies to all loss functions whose partial derivative with respect to spiking activity can be expressed as

$$\frac{\partial E}{\partial z_{j,t}} = \sum_{t' \geq t} \kappa^{t'-t}\phi_{j,t'}. \tag{2.9}$$

As an important example, our derivation is immediately applicable to the actor-critic reinforcement learning framework [4]. For the regression task,

$$\phi_{j,t'} = (1-\kappa)\sum_k (y_{k,t'} - y_{k,t'}^*) w_{kj}^{\mathrm{OUT}} \tag{2.10}$$

and for the classification task,

$$\phi_{j,t'} = (1-\kappa)\sum_k \pi_{k,t'}^* \pi_{k,t'} \sum_{k'} (w_{k'j}^{\mathrm{OUT}} - w_{kj}^{\mathrm{OUT}}) \exp(y_{k',t'} - y_{k,t'}). \tag{2.11}$$

One can see that when the leak κ is not 0, the error derivative will depend on future errors, which seemingly poses an obstacle to online learning. We provide the online implementation for this readout convention in Supplementary Note 1. In addition to accuracy optimization described above, we added a firing rate regularization term $E_{reg} = \frac{1}{2}c_{\mathrm{reg}}\sum_j (f_j^{\mathrm{av}} - f_j^{\mathrm{target}})^2$ to the loss function to ensure sparse firing [4]. Here, f_j^{target} and $f_j^{\mathrm{av}} = \frac{1}{T}\sum_t z_{j,t}$ are the desired and actual average firing rate for cell j, respectively, and c_{reg} is a positive coefficient that controls the strength of the regularization.

Notation for Derivatives: Following the notation in [4], there are two types of computational dependencies in RSNNs; direct and indirect dependencies. For example, variable w_{pq} can impact state $s_{p,t}$ directly through Eq. 2.2 as well as indirectly via its influence through other cells in the network. We distinguish direct dependencies versus all dependencies (including indirect ones) using partial derivatives (∂) versus total derivatives (d).

Gradient descent learning in RSNNs: We study iterative adjustment of all synaptic weights (input weights w^{IN}, recurrent weights w and output weights w^{OUT}) using gradient descent on loss E:

$$\begin{aligned} w_{pq,\text{new}} &= w_{pq,\text{old}} - \lambda \Delta w_{pq}, \\ \Delta w_{pq} &= \frac{\mathrm{d}\, E}{\mathrm{d}\, w_{pq,\text{old}}}, \end{aligned} \tag{2.12}$$

where λ denotes the learning rate, and the gradient of the error with respect to the synaptic weights must be calculated. This error gradient can be calculated with classical machine learning algorithms, backpropagation through time (BPTT) and real time recurrent learning (RTRL), by unwrapping the RSNN dynamics over time (Figure 2.6b). While these two algorithms yield equivalent results, their bookkeeping for chain rule differs. Gradient calculations in BPTT depend on future activity, which poses an obstacle for online learning and biological plausibility. Our learning rule derivation follows the RTRL factorization because it is causal. Therefore, we focus our analysis on RTRL and factor the error gradient across time and space as

$$\frac{\mathrm{d}\, E}{\mathrm{d}\, w_{pq}} = \sum_{j,t} \frac{\partial E}{\partial z_{j,t}} \frac{\mathrm{d}\, z_{j,t}}{\mathrm{d}\, w_{pq}}, \tag{2.13}$$

$$\frac{\mathrm{d}\, z_{j,t}}{\mathrm{d}\, w_{pq}} = \frac{\partial z_{j,t}}{\partial s_{j,t}} \frac{\mathrm{d}\, s_{j,t}}{\mathrm{d}\, w_{pq}}, \tag{2.14}$$

following the derivative notation explained above. The factor $\frac{\partial E}{\partial z_{j,t}}$ in Eq. 2.13 is related to the top-down learning signal $L_{j,t} := \sum_k w^{OUT}_{kj}(y_{k,t} - y^*_{k,t})$ [4]. Supplementary Notes 1, 2 show that the leak term of the output neurons makes these two terms different, and derives an online implementation that uses $L_{j,t}$. We thus take top-down learning signals to be cell-specific rather than global, which is justified in part by recent reports that dopamine signals [1, 89]

and error-related neural firing [104] can be specific to a population of neurons [4]. Moreover, approximating the sum in $L_{j,t}$ as in our main derivation below following the argument on cognate receptors or using the random feedback alignment theory [12] (on only the outgoing connections between spiking neurons and output units) suggest further biologically plausible implementations.

We now discuss the second factor in Eq. 2.13, i.e. $\frac{\mathrm{d}\, z_{j,t}}{\mathrm{d}\, w_{pq}}$. This is expanded into two factors in Eq. 2.14. The first factor, $h_{j,t} := \frac{\partial z_{j,t}}{\partial s_{j,t}}$ is problematic to compute for spiking neurons due to the discontinuous step function H in Eq. 2.3, whose derivative is not defined at 0 and is 0 everywhere else. We overcome this issue by approximating the decay of the derivative using a piece-wise linear function [4, 77, 98, 105]. Here, the pseudoderivative $h_{j,t}$ is defined as follows:

$$
\begin{aligned}
h_{j,t} &= \frac{\mathrm{d}\, z_{j,t}}{\mathrm{d}\, s_{j,t}} && (2.15)\\
&\approx \gamma \max\left(0, 1 - \left|\frac{s_{j,t} - A_{j,t}}{v_{\mathrm{th}}}\right|\right), && (2.16)
\end{aligned}
$$

The dampening factor γ (typically set to 0.3) dampens the increase of backpropagated errors in order to improve the stability of training very deep (unrolled) RSNNs [4]. Throughout this study, neuronal firing displays refractoriness, where $h_{j,t}$ and $z_{j,t}$ are fixed at 0 after each spike of neuron j (See Supplementary Note 3).

Key problems that RTRL poses to biological plausibility and computational cost reside in the factor $\frac{\mathrm{d}\, s_{j,t}}{\mathrm{d}\, w_{pq}}$ that arises during the factorization of the gradient (Eq. 2.13 and Eq. 2.14). The factor $\frac{\mathrm{d}\, s_{j,t}}{\mathrm{d}\, w_{pq}}$ keeps track of all direct and indirect dependencies of neuron state j on weight w_{pq}. In other words, this factor accounts for both the spatial and temporal dependencies in RSNNs: state dependencies across time t, as explained above, result from unwrapping the temporal dependencies illustrated in Figure 2.6b; state dependencies across space, however, are due to the indirect dependencies (of all z_t on w and all $z_{t'}$ $(t' < t)$) arising from recurrent connections (Figure 2.6c). These recurrent dependencies are all accounted for in the $\frac{\mathrm{d}\, s_{j,t}}{\mathrm{d}\, w_{pq}}$

factor, which can be obtained recursively as follows:

$$\begin{aligned}\frac{\mathrm{d}\, s_{j,t}}{\mathrm{d}\, w_{pq}} &= \frac{\partial s_{j,t}}{\partial w_{pq}} + \sum_l \frac{\partial s_{j,t}}{\partial s_{l,t-1}} \frac{\mathrm{d}\, s_{l,t-1}}{\mathrm{d}\, w_{pq}} \\ &= \frac{\partial s_{j,t}}{\partial w_{pq}} + \frac{\partial s_{j,t}}{\partial s_{j,t-1}} \frac{\mathrm{d}\, s_{j,t-1}}{\mathrm{d}\, w_{pq}} + \underbrace{\textstyle\sum_{l \neq j} w_{jl} \frac{\partial z_{l,t-1}}{\partial s_{l,t-1}} \frac{\mathrm{d}\, s_{l,t-1}}{\mathrm{d}\, w_{pq}}}_{\text{depends on all weights } w_{jl}}. \end{aligned} \tag{2.17}$$

Thus, the factor $\frac{\mathrm{d}\, s_{j,t}}{\mathrm{d}\, w_{pq}}$ is a memory trace of all inter-cellular dependencies (Figures 2.6di), requires $O(N^3)$ memory and $O(N^4)$ computations. This makes RTRL expensive to implement for large networks. Moreover, this last factor poses a serious problem for biological plausibility: it involves nonlocal terms, so that knowledge of all other weights in the network is required in order to update the weight w_{pq}.

To address this, Murray [9] and Bellec *et al.* [4] ("e-prop") dropped the nonlocal terms so that the updates to weight w_{pq} would only depend on pre- and post-synaptic activity (Figures 2.6dii, 2.5b), and applied this truncation to train rate-based and spiking neural networks, respectively. While both works succeed in improving over previous biologically plausible learning rules, a significant performance gap with respect to the full BPTT/RTRL algorithms remains.

Derivation of multidigraph learning in RSNNs: We continue from the previous section in giving a detailed derivation of our learning rule. To reveal a potential role for cell-type-based modulatory signals in synaptic plasticity as well as improve upon the aforementioned biologically plausible gradient descent approximations, we begin by partially restoring non-local dependencies between cells – those within one connection step. This is the *truncated* RTRL framework (Figures 2.6diii, 2.5d), and the memory trace term $\frac{\mathrm{d}\, s_{j,t}}{\mathrm{d}\, w_{pq}}$ becomes

$$\frac{\mathrm{d}\, s_{j,t}}{\mathrm{d}\, w_{pq}} \approx \begin{cases} \frac{\partial s_{j,t}}{\partial z_{p,t-1}} \frac{\partial z_{p,t-1}}{\partial s_{p,t-1}} \frac{\mathrm{d}\, s_{p,t-1}}{\mathrm{d}\, w_{pq}} = w_{jp} \frac{\partial z_{p,t-1}}{\partial s_{p,t-1}} \frac{\mathrm{d}\, s_{p,t-1}}{\mathrm{d}\, w_{pq}}, & p \neq j \\ \frac{\partial s_{j,t}}{\partial w_{jq}} + \frac{\partial s_{j,t}}{\partial s_{j,t-1}} \frac{\mathrm{d}\, s_{j,t-1}}{\mathrm{d}\, w_{jq}}, & p = j \end{cases} \tag{2.18}$$

Thus, when $j = p$, our truncation implements $\frac{\mathrm{d}\, s_{p,t}}{\mathrm{d}\, w_{pq}} \approx \frac{\partial s_{j,t}}{\partial w_{jq}} + \frac{\partial s_{j,t}}{\partial s_{j,t-1}} \frac{\mathrm{d}\, s_{j,t-1}}{\mathrm{d}\, w_{jq}}$, which coincides with e-prop. Eq. 2.18 adds the case when $p \neq j$, for which $\frac{\mathrm{d}\, s_{j,t}}{\mathrm{d}\, w_{pq}}$ was simply set to 0 in e-prop. We note that the truncation in Eq. 2.18 resembles the n-step RTRL approximation recently proposed in [75], known as SnAP-n, which stores $\frac{\mathrm{d}\, s_{j,t}}{\mathrm{d}\, w_{pq}}$ only for j such that parameter w_{pq}

influences the activity of unit j within n time steps. The computations of SnAp-n converge to those of RTRL as n increases, resonating with our improved performance when more terms of the exact gradient are included. Our truncation in Eq. 2.18 is similar to SnAp-n with $n = 2$ with two differences: (i) we apply it to spiking neural networks, (ii) we drop the previous time step's Jacobian term $\frac{\mathrm{d}\, s_{j,t-1}}{\mathrm{d}\, w_{pq}}$, which would necessitate the maintenance of a rank-three ("3-d") tensor with costly storage demands ($O(N^3)$) and for which no known biological mechanisms exist. Thus, the truncation in Eq. 2.18 requires the maintenance of only a rank-two ("2-d") tensor specific to synapse $p|q$, which can be realized via an eligibility trace as we explain next.

By substituting equation Eq. 2.18 into Eq. 2.13 and Eq. 2.14, we approximate the overall gradient as

$$\begin{aligned}\widehat{\frac{\mathrm{d}\, E}{\mathrm{d}\, w_{pq}}} &= \sum_t \left[\frac{\partial E}{\partial z_{p,t}} \frac{\partial z_{p,t}}{\partial s_{p,t}} \frac{\mathrm{d}\, s_{p,t}}{\mathrm{d}\, w_{pq}} + \sum_{j \neq p} \frac{\partial E}{\partial z_{j,t}} \frac{\partial z_{j,t}}{\partial s_{j,t}} w_{jp} \frac{\partial z_{p,t-1}}{\partial s_{p,t-1}} \frac{\mathrm{d}\, s_{p,t-1}}{\mathrm{d}\, w_{pq}} \right] \\ &= \sum_t L_{p,t} e_{pq,t} + \underbrace{\textstyle\sum_j a_{j,t} w_{jp} e_{pq,t-1}}_{:=\widehat{\Gamma_{pq,t}}}, \end{aligned} \tag{2.19}$$

where $L_{p,t} := \frac{\partial E}{\partial z_{p,t}}$ is the top-down learning signal to cell p, $a_{j,t}$ (Eq. 2.24) denotes the activity-dependent modulatory signal emitted by neuron j at time t and $e_{pq,t}$ (Eq. 2.25) is the *eligibility trace* maintained by postsynaptic cell p to keep a memory of the preceding activity of presynaptic cell q and postsynaptic cell p. In Eq. 2.19, the first term $L_{p,t} e_{pq,t}$ alone gives exactly the e-prop synaptic update rule. The second term, which we define as $\widehat{\Gamma_{pq,t}}$, is a synaptically non-local term due to contributions from local modulatory signals. As seen in Eq. 2.19, our truncation requires maintaining a $\{p, q\}$-dependent double tensor (for $e_{pq,t}$) instead of a triple one, thereby reducing the memory cost of RTRL from $O(N^3)$ to $O(N^2)$.

Importantly, we observe that, for the update to synapse w_{pq} in Eq. 2.19, the terms that depend on cells j *only appear under a sum.* Therefore, the mechanism updating the synapse $p|q$ does not need to know the individual terms indexed by j. Rather, only their sum suffices.

While it is tempting to consider the first factors in $\widehat{\Gamma_{pq,t}}$, $a_{j,t} w_{jp}$, as the modulatory signal emitted by neuron j, the involvement of the synapse from neuron p via w_{jp} and a lack of

known mechanisms in calculating this neuron-specific composite signal suggest that this is unlikely to be a biological solution. Instead, inspired by the cell-type-specific (rather than neuron-specific) affinities for peptidergic neuromodulation [3, 62], we propose to approximate the signaling gain w_{jp} in Eq. 2.19 by the average value $w_{\alpha\beta}$ across its pre- and postsynaptic cell types. More specifically, when postsynaptic cell j belongs to type α and presynaptic cell p belongs to type β, we approximate neuron-specific weight w_{jp} with cell-type-specific gain $w_{\alpha\beta} =< w_{jp} >_{j\in\alpha,p\in\beta}$. We hypothesize that $w_{\alpha\beta}$ represents the affinity of the G-protein coupled receptors expressed by cells of type β to the modulators secreted by cells of type α. (See Cell-type-specific receptor affinities below.)

Thus, the gradient estimate at time t due to our learning rule involves compounding eligibility trace with both top-down and local modulatory signals, thereby recovering the general form introduced in Eq. 2.1:

$$\left.\frac{\mathrm{d}\,E}{\mathrm{d}\,w_{pq}}\right|_{t,\mathrm{MDGL}} \approx L_{p,t}e_{pq,t} + \Gamma_{pq,t},$$
$$\Gamma_{pq,t} = \left(\sum_{\alpha\in C} w_{\alpha\beta} \sum_{j\in\alpha,p\to j} a_{j,t}\right) e_{pq,t-1}, \tag{2.20}$$

where neuron p is of type β, C denotes the set of neuronal cell types, $p \to j$ denotes that there is a synaptic connection from neuron p to j, and $\Gamma_{pq,t}$ approximates the second term in Eq. 2.19 with cell-type-specific weight averages.

In summary, cell p receives local modulatory input $Mod.input_p$ that gets combined with the eligibility trace (as per Eq. 2.20) in addition to synaptic input $Syn.input_p$ (as per Eq. 2.2):

$$Mod.input_p := \sum_{\alpha\in C} w_{\alpha\beta} \sum_{j\in\alpha,p\to j} a_{j,t} \tag{2.21}$$

$$Syn.input_p := \sum_{l\neq p} w_{pl}\, z_{l,t} + \sum_{p} w^{\mathrm{IN}}_{pm}\, x_{m,t+1}. \tag{2.22}$$

It may be instructive to note the dichotomy in the functions of these two different inputs: the cell uses $Mod.input_p$ to regulate its synaptic plasticity but not to change its internal state, and it uses $Syn.input_p$ to change its internal state but not to regulate synaptic plasticity.

Hence, our update rule suggests a new additive term to compute the plasticity update at synapse $p|q$ at time t, $\Gamma_{pq,t}$, which calculates multiplicative contributions of the modulatory

signal $a_{j,t}$ secreted by neuron j, the affinity of receptors of cell type β to ligands of type α, $w_{\alpha\beta}$, and the eligibility trace at the synapse $p|q$, $e_{pq,t}$. The following two sections explain how two main components of Γ, cell-type-specific signals and eligibility trace, can be implemented.

Cell-type-specific receptor affinities: We explain cell-type-specific signaling implementation, notably how type-specific receptor affinity $w_{\alpha\beta}$ is defined. As introduced in our learning rule derivation, $w_{\alpha\beta}$ is an approximation of gain w_{jp} (Eq. 2.19 and Eq. 2.20), and we proposed to define $w_{\alpha\beta}$ as the weight average across its pre- and postsynaptic cell types:

$$w_{jp} \approx \begin{cases} w_{\alpha\beta}, & p \to j \\ 0, & \text{otherwise} \end{cases} \tag{2.23}$$

where $p \to j$ denotes that there is a synaptic connection from neuron p to j, motivated by the local diffusion assumption discussed in [91], in which this type of signaling is registered only by local synaptic partners and therefore preserves the connectivity structure of w_{jp}. One obvious variant of this receptor affinity definition is one with a different spatial extent, for which we examine the opposite extreme in Figure 2.4d-f, where modulatory signals diffuse to all cells in the network. More specifically, the signaling gain $w_{\alpha\beta}$ is replaced by $w^{NL}_{\alpha\beta} =< w_{jp} >_{j\in\alpha,p\in\beta}$ even for $w_{jp} = 0$ so that modulatory signals diffuse to all cells with the same strength in the network.

For a proof of concept, we implemented MDGL with modulatory types mapped to the two main cell classes; i.e. cell-type-specific signaling gain, $w_{\alpha\beta} =< w_{jp} >_{j\in\alpha,p\in\beta}$ with $\alpha, \beta \in \{E, I\}$. We demonstrate the effectiveness of this cell-type discretization by comparing its learning performance to the case without cell type discretization ($w_{\alpha\beta} = w_{jp}$, i.e. each cell is its own type) and observed little difference in performance (Figure S9. On the other hand, increasing the number of modulatory types involved in the cell type discretization could be key to realizing the potential of MDGL in more complicated tasks, suggesting an explanation for the observed diversity of cell types in the brain.

We implemented receptor affinities as weight averages across types, but how tightly coupled are those modulatory gains and synaptic weights is a subject for future investigation. Figure S7 explores the sensitivity of the learning performance to imprecise receptor affinities.

Activity-dependent modulatory emission implementation: As introduced in

Eq. 2.20, activity-dependent modulatory signal emitted by neuron j at time t, an important component of MDGL, is defined as

$$a_{j,t} = \frac{\partial E}{\partial z_{j,t}} \frac{\partial z_{j,t}}{\partial s_{j,t}}. \tag{2.24}$$

As defined, $a_{j,t}$, is a package of two components: $\frac{\partial E}{\partial z_{j,t}}$, which is referred to as the top-down signal [4], and $h_{j,t} = \partial z_{j,t}/\partial s_{j,t}$, which is the pseudo-derivative of spiking activity as a function of cell j's membrane potential explained above. While Eq. 2.20 suggests an "online" implementation with the update at t, the factor $a_{j,t}$ cannot be calculated causally unless the output is not leaky. Supplementary Note 1 derives an online update for the more general case, which has the same form as Eq. 2.20.

Eligibility trace implementation: As introduced in Eq. 2.20, eligibility trace, another important component of MDGL, is defined as

$$e_{pq,t} := \frac{\partial z_{p,t}}{\partial s_{p,t}} \frac{\mathrm{d}\, s_{p,t}}{\mathrm{d}\, w_{pq}}, \tag{2.25}$$

$$\frac{\mathrm{d}\, s_{p,t}}{\mathrm{d}\, w_{pq}} = \frac{\partial s_{p,t}}{\partial w_{pq}} + \frac{\partial s_{p,t}}{\partial s_{p,t-1}} \frac{\mathrm{d}\, s_{p,t-1}}{\mathrm{d}\, w_{pq}}, \tag{2.26}$$

where Eq. 2.26 follows directly from Eq. 2.18. $\frac{\mathrm{d}\, s_{p,t}}{\mathrm{d}\, w_{pq}}$ can be obtained recursively and is referred to as the eligibility vector [4]. $e_{pq,t}$ keeps a fading memory of activity pertaining to presynaptic cell q and postsynaptic cell p. A discussion on interpreting eligibility traces as derivatives can be found in [4]. Here, we briefly explain its implementation by expanding the factors in Eqs. 2.25 and 2.26 for both LIF and ALIF cells.

For LIF cells, there is no adaptive threshold so the hidden state consists only of the membrane potential. Thus, we have factors $\frac{\partial z_{p,t}}{\partial s_{p,t}} = h_{p,t}$ with pseudo-derivative $h_{p,t}$ defined in Eq. 2.15, $\frac{\partial s_{p,t}}{\partial w_{pq}} = z_{q,t-1}$ and $\frac{\partial s_{p,t+1}}{\partial s_{p,t}} = \eta - v_{\mathrm{th}} h_{p,t}$ following Eq. 2.2.

For ALIF cells, there are two hidden variables so the eligibility vector is now a two dimensional vector $\frac{\mathrm{d}\, s_{p,t}}{\mathrm{d}\, w_{pq}} = [\frac{\mathrm{d}\, s^v_{p,t}}{\mathrm{d}\, w_{pq}}, \frac{\mathrm{d}\, s^b_{p,t}}{\mathrm{d}\, w_{pq}}] \in \mathbb{R}^{2x1}$ pertaining to membrane potential $v_{p,t}$ and adaptive threshold state $b_{p,t}$. Following Eq. 2.3, one can obtain factors $\frac{\partial z_{p,t}}{\partial s_{p,t}} = [\frac{\partial z_{p,t}}{\partial v_{p,t}}, \frac{\partial z_{p,t}}{\partial b_{p,t}}]$ $= [h_{p,t}, -\beta h_{p,t}] \in \mathbb{R}^{1x2}$, $\frac{\partial s_{p,t}}{\partial w_{pq}} = [z_{q,t-1}, 0] \in \mathbb{R}^{2x1}$ and $\frac{\partial s_{p,t}}{\partial s_{p,t-1}}$ is now a 2-by-2 matrix:

$$\frac{\partial s_{p,t}}{\partial s_{p,t-1}} = \begin{bmatrix} \frac{\partial v_{p,t}}{\partial v_{p,t-1}} & \frac{\partial v_{p,t}}{\partial b_{p,t-1}} \\ \frac{\partial b_{p,t}}{\partial v_{p,t-1}} & \frac{\partial b_{p,t}}{\partial b_{p,t-1}} \end{bmatrix} = \begin{bmatrix} \eta - v_{\mathrm{th}} h_{p,t} & v_{\mathrm{th}} \beta h_{p,t} \\ (1-\rho) h_{p,t} & \rho - (1-\rho)\beta h_{p,t} \end{bmatrix} \in \mathbb{R}^{2x2}. \tag{2.27}$$

Thus, the eligibility trace $e_{pq,t}$ is scalar valued regardless of the dimension of the eligibility vector.

S2.1 Online modulatory signaling for leaky output

As mentioned in Methods, we allow leaky outputs

$$y_{k,t} = \kappa\, y_{k,t-1} + \sum_j w_{kj}^{\mathrm{OUT}} z_{j,t} + b_k^{\mathrm{OUT}},$$

where the output at the current step depends on the previous time step through the leak constant κ.

Based on this leaky output and loss $E = \sum_{k,t}(y_{k,t}^* - y_{k,t})^2$, we have the following partial derivative that appears in the naive implementation of modulatory emission (Eq. 2.24):

$$\frac{\partial E}{\partial z_{j,t}} = \sum_k w_{kj}^{OUT} \sum_{t' \geq t} (y_{t',k}^* - y_{t',k}) \kappa^{t'-t}, \tag{S2.1}$$

which is dependent on future errors (and problematic for online learning).

Consider the two additive components of our learning rule in Eq. 2.20. The first term, $\frac{\mathrm{d}\,E}{\mathrm{d}\,w_{pq}}\Big|_{e-prop} = L_{p,t} e_{pq,t}$, is implementable online following a simple change of summation order [4] (derivation can be generalized to the classification task with a simple replacement of $(y_{k,t}^* - y_{k,t})$ by $(\pi_{k,t}^* - \pi_{k,t})$):

$$\begin{aligned}
\frac{\mathrm{d}\,E}{\mathrm{d}\,w_{pq}}\Big|_{e-prop} &= \sum_{t'} \frac{\partial E}{\partial z_{p,t'}} e_{pq}^{t'} \\
&= \sum_{k,t'} w_{kj}^{OUT} \sum_{t \geq t'} (y_{k,t}^* - y_{k,t}) \kappa^{t-t'} e_{pq}^{t'} \\
&= \sum_{k,t} w_{kj}^{OUT} (y_{k,t}^* - y_{k,t}) \underbrace{\textstyle\sum_{t' \leq t} \kappa^{t-t'} e_{pq}^{t'}}_{\mathcal{F}_\kappa(e_{pq}^t)},
\end{aligned} \tag{S2.2}$$

where the order of summations was changed in the last line, and operator $\mathcal{F}_\kappa$ denotes low-pass filtering with $\mathcal{F}_\kappa(x_t) = \kappa \mathcal{F}_\kappa(x_{t-1}) + x_t$. In our actual implementation, we used an exponential smoothing with $\mathcal{F}_\kappa(x_t) = \kappa \mathcal{F}_\kappa(x_{t-1}) + (1-\kappa) x_t$, but dropped the factor $(1-\kappa)$ in writing for readability.

We again apply the change of summation order trick to the second term of our learning rule (Eq. 2.20), $\Gamma_{pq,t}$, and assume that activity of neuron j is not correlated with the eligibility

trace of synapse pq:

$$
\begin{aligned}
\sum_{t'} \Gamma_{pq,t'} &= \sum_{t',j\neq p} \frac{\partial E}{\partial z_{j,t'}} h_{j,t'} w_{\alpha\beta} e_{pq}^{t'-1} \\
&= \sum_{k,t',j\neq p} w_{kj}^{OUT} \sum_{t\geq t'} (y_{k,t}^{*} - y_{k,t}) \kappa^{t-t'} h_{j,t'} w_{\alpha\beta} e_{pq}^{t'-1} \\
&\overset{(a)}{=} \sum_{t} \sum_{j\neq p} \sum_{k} (y_{k,t}^{*} - y_{k,t}) w_{kj}^{OUT} w_{\alpha\beta} \underbrace{\textstyle\sum_{t'\leq t} \kappa^{t-t'} h_{j,t'} e_{pq}^{t'-1}}_{\approx (t-t'+1)\mathbb{E}_{t'\leq t}\left[\kappa^{t-t'} h_{j,t} e_{pq}^{t-1}\right]} \\
&\overset{(b)}{\approx} \sum_{t} \sum_{j\neq p} \sum_{k} (y_{k,t}^{*} - y_{k,t}) w_{kj}^{OUT} w_{\alpha\beta} \mathbb{E}_{t'\leq t}\left[h_{j,t}\right] \underbrace{(t-t'+1)\mathbb{E}_{t'\leq t}\left[\kappa^{t-t'} e_{pq}^{t-1}\right]}_{=\mathcal{F}_{\kappa}(e_{pq}^{t-1})} \\
&= \sum_{t} \sum_{j\neq p} \sum_{k} (y_{k,t}^{*} - y_{k,t}) w_{kj}^{OUT} w_{\alpha\beta} \underbrace{\mathbb{E}_{t'\leq t}\left[h_{j,t}\right]}_{\approx \mathcal{F}_{\kappa}(h_{j,t})} \mathcal{F}_{\kappa}(e_{pq}^{t-1}) \\
&\overset{(c)}{\approx} \sum_{t} \sum_{j\neq p} \sum_{k} (y_{k,t}^{*} - y_{k,t}) w_{kj}^{OUT} w_{\alpha\beta} \mathcal{F}_{\kappa}(h_{j,t}) \mathcal{F}_{\kappa}(e_{pq}^{t-1}) \\
&= \sum_{t} \sum_{j\neq p} \underbrace{\left[\textstyle\sum_{k} (y_{k,t}^{*} - y_{k,t}) w_{kj}^{OUT} \mathcal{F}_{\kappa}(h_{j,t})\right]}_{:=\bar{a}_{j,t}} w_{\alpha\beta} \mathcal{F}_{\kappa}(e_{pq}^{t-1}) \\
&\overset{(d)}{=} \sum_{t} \mathcal{F}_{\kappa}(e_{pq}^{t-1}) \sum_{\alpha\in C} w_{\alpha\beta} \sum_{j\in\alpha} \bar{a}_{j,t},
\end{aligned}
\tag{S2.3}
$$

where (a) changes the summation order; (b) assumes uncorrelatedness between activity $h_{j,t}$ and $\kappa^{t-t'} e_{pq}^{t-1}$ such that $\mathbb{E}_{t'\leq t}\left[\kappa^{t-t'} h_{j,t} e_{pq}^{t-1}\right] \approx \mathbb{E}_{t'\leq t}\left[h_{j,t}\right] \mathbb{E}_{t'\leq t}\left[\kappa^{t-t'} e_{pq}^{t-1}\right]$; (c) approximates the temporal average of $h_{j,t}$ using an exponential filter $\mathbb{E}_{t'\leq t}\left[h_{j,t}\right] \approx \mathcal{F}_{\kappa}(h_{j,t})$; (d) is a simple change of summation order. We test the validity of the above approximation in Figure S2.4 and observe no significant performance degradation due to this approximation.

S2.2 Detailed Breakdown of MDGL's Components

In the main text, we stated that our MDGL learning rule combines the eligibility trace with both top-down learning signals and cell-type-specific weighted summation of secreted, diffuse modulators. We so far only expressed these components as derivatives (see Methods). With the derivation of the online implementation for MDGL in Eq. S2.3, we are now ready to provide the detailed expressions for each of these components. Combining Eq. S2.3 with Eq. 2.20 and rearranging the summation order gives the following component breakdown for

our online approximation to MDGL:

$$\widehat{\frac{\mathrm{d}\,E}{\mathrm{d}\,w_{pq}}} \approx \left[\sum_k (y^*_{t-1,k} - y_{t-1,k}) \left(w^{OUT}_{kp} + \underbrace{\sum_{\alpha\in C} w_{\alpha\beta} \sum_{j\in\alpha} w^{OUT}_{kj} \mathcal{F}_\kappa(h_{j,t-1})}_{\text{Addition due to local modulatory signals}} \right)\right] \mathcal{F}_\kappa(e_{pq,t-1}). \tag{S2.4}$$

The non-neuron-specific error signal $(y^*_{t,k} - y_{t,k})$ is passed to cells through neuron-specific feedback weights w^{OUT}_{kj}, thereby forming neuron-specific learning signal at the receiving end $L_{j,t} = \sum_k w^{OUT}_{kj}(y^*_{t,k} - y_{t,k})$. Here, we took top-down learning signals to be cell-specific rather than global, which is justified in part by recent reports that dopamine signals [1] and error-related neural firing [104] can be specific to a population of neurons [4]. On the other hand, loosening this neuron-specificity of learning signal can be achieved through approximations to the feedback weights, such as cell-type-specific approximations as in Eq. 2.23. Upon receipt, neuron j multiplies $L_{j,t}$ with $\mathcal{F}_\kappa(h_{j,t})$, its low-pass filtered activity, and sends the packaged signal $a_{j,t} = L_{j,t}\mathcal{F}_\kappa(h_{j,t})$. In updating w_{pq}, our addition allows postsynaptic cell p to collect information regarding the activities and learning signals of other cells through cell-type-specific gain $w_{\alpha\beta}$, and combine the received modulatory input with its low-pass filtered eligibility trace.

S2.3 Analysis and Simulation Details

Throughout this study, we used alignment angle to quantify the similarity between two vectors. The alignment angle θ between two vectors, a and b, was computed by $\theta = acos(\|a^T b\|/\|a\|\|b\|)$. The alignment between two 2D matrices was computed by flattening the matrices into vectors. For spectral analysis, we first performed root mean square normalization on the signal and then computed the power spectral density using Welch's method [106].

For the pattern generation task in Figure 2.3a, our network consisted of 400 LIF neurons. All neurons had a membrane time constant of $\tau_m = 30$ms, a baseline threshold of $v_{\text{th}} = 0.01$ and a refractory period of 2ms. Input to this network was provided by 100 Poisson spiking neurons with a rate of 10Hz. The fixed target signal had a duration of 2000ms and given

by the sum of five sinusoids, with fixed frequencies of 0.5Hz, 1Hz, 2Hz, 3Hz and 4Hz. For learning, we used mean squared loss function and for visualization, we used normalized mean squared error NMSE $= \frac{\sum_{k,t}(y^*_{k,t}-y_{k,t})^2}{\sum_{k,t}(y^*_{k,t})^2}$ for zero-mean target output $y^*_{k,t}$. All weight updates were implemented using Adam with default parameters [107] and a learning rate of 1×10^{-3}. In addition, we applied firing rate regularization with $c_{\text{reg}} = 10$ and $f^{\text{target}} = 10Hz$.

For the delayed match to sample task in Figure 2.3b, our implementation of the task began with a brief fixation period (no cues) followed by two sequential cues, each lasting 0.15s and separated by a 0.75s delay (Figure 2.3b). A cue of value 1 was represented by 40Hz Poisson spiking input, whereas a cue of value 0 was represented by the absence of input spiking. The network was trained to output 1 (resp. 0) when the two cues have matching (resp. non-matching) values. Our network consisted of 50 LIF neurons and 50 ALIF neurons (100 LIF neurons and 80 ALIF neurons for the alternative setup in Figure S2.5). All neurons had a membrane time constant of $\tau_m = 20$ms, a baseline threshold of $v_{\text{th}} = 0.01$ and a refractory period of 5ms. The time constant of threshold adaptation was set to $\tau_b = 1400$ms, and its impact on the threshold was set to $\beta = 1.8$. Input to this network was provided by three populations, as illustrated in Figure 2.3B. The first (resp. second) population consisted of 20 units and produced Poisson spike trains with a rate of 40Hz when the first (resp. second) cue takes a value of 1, otherwise it stays quiescent. The last input population of 10 units produced Poisson spike trans of 10Hz throughout the trial in order to prevent the network from being quiescent during the delay. For the alternative setup in Figure S2.5, the first (resp. second) input population produced Poisson spike trains with a rate of 40Hz when cue 1 (resp. cue 2) is presented, otherwise it fires at 10Hz. For learning, we used cross-entropy loss function and the target corresponding to the correct output was given at the end of the trial. As done in the evidence accumulation task, a weight update was applied once every 64 trials and the gradients were accumulated during those trials additively. All weight updates were implemented using Adam with default parameters [107] and a learning rate of 2.5×10^{-3}. In addition, we applied firing rate regularization with $c_{\text{reg}} = 0.1$ and $f^{\text{target}} = 10$Hz.

For the evidence accumulation task in Figure 2.3c, our network consisted of 50 LIF neurons and 50 ALIF neurons. All neurons had a membrane time constant of $\tau_m = 20$ms, a baseline threshold of $v_{\text{th}} = 0.01$ and a refractory period of 5ms. The time constant of

threshold adaptation was set to $\tau_b = 2000$ms, and its impact on the threshold was set to $\beta = 1.8$. Input to this network was provided by four populations of 10 neurons each, as illustrated in Figure 2.3c. Each cue is represented by 40 Hz Poisson spiking input for 100ms and cues are separated by 50ms. The first (resp. the second) population produced Poisson spike trains with a rate of 40Hz when a cue was presented on the left (resp. right) side of the track. The third input population spiked randomly through the decision period with a firing rate of 40Hz and was silent otherwise. The last input population produced Poisson spike trains with a rate of 10Hz throughout the trial in order to prevent the network from being quiescent during the delay. For learning, we used the cross-entropy loss function and the target corresponding to the correct output was given at the end of the trial. A weight update was applied once every 64 trials and the gradients were accumulated during those trials additively. All weight updates were implemented using Adam with default parameters [107] and a learning rate of 2.5×10^{-3}. In addition, we applied firing rate regularization with $c_{\text{reg}} = 0.1$ and $f^{\text{target}} = 10Hz$. In the main text, all simulated tasks are constrained at 10% sparsity. This connection sparsity is maintained by fixing inactive synapses with zero weights. Cells have synapses that are sign constrained with 80% of the population being excitatory and the rest inhibitory. For a proof of concept, we implement MDGL with modulatory types mapped to the two main cell classes, thus obtaining four cell-type-specific gain values (Cell-type-specific receptor affinities, Methods). The same learning rate is used for all methods within a given task. We also compared the methods at their best learning rate within $\{1e-3, 2e-3, 5e-3, 1e-2, 2e-2, 5e-2\}$ and the trend still holds: BPTT learns the fastest, whereas MDGL leads to faster loss reduction over training iterations than e-prop. For all simulations, we used a time step of 1ms. We also assumed a synaptic delay of 1ms for all synapses.

Lastly, we note that while input, recurrent and output weights are all being trained, biologically plausible learning rules (i.e. e-prop and MDGL) only apply to input and recurrent weights. All approaches update the output weights using backpropagation, as output weights do not suffer the aforementioned nonlocality issue. (For updating the weights of a single output layer, random feedback alignment [12] has also been shown to be an effective and biologically plausible solution.)

Supplementary Figures

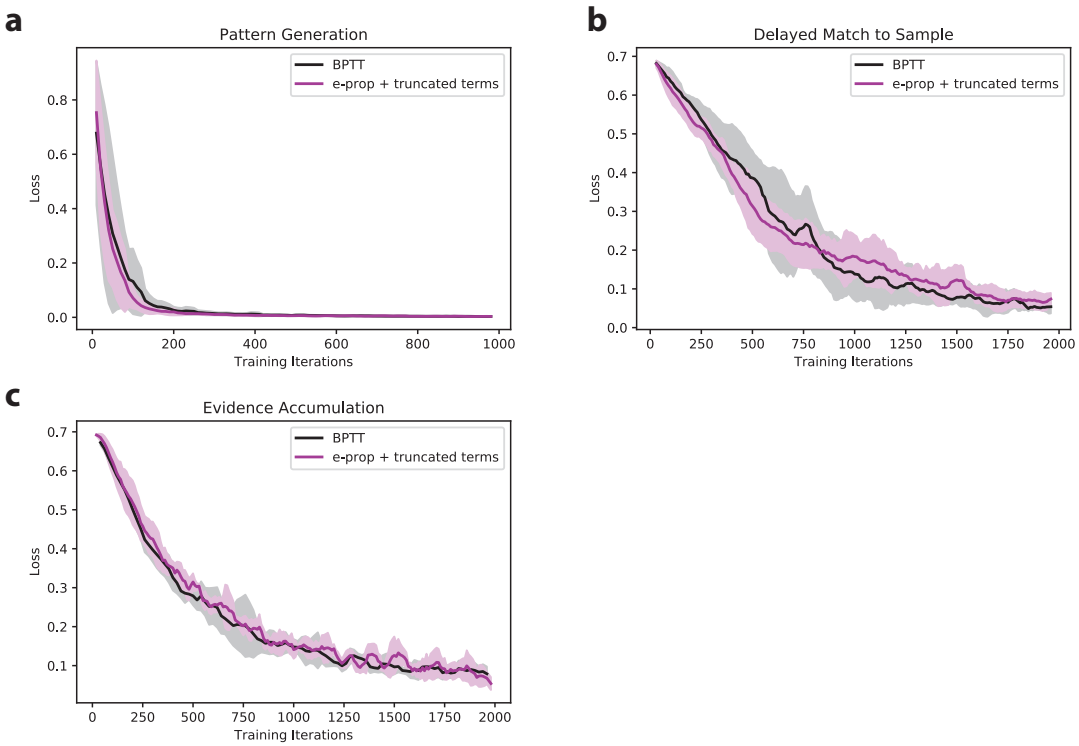

Figure S2.1: Checking e-prop implementation – recovering ignored terms recovers the performance of BPTT. As a sanity check, learning curves are plotted for e-prop plus all the truncated terms (see Eq. 2.17) to verify that the resulting learning rule recovers the performance of BPTT. The check is applied to a) pattern generation, b) delayed match to sample and c) evidence accumulation tasks. Solid lines show the mean averaged across five runs and shaded regions show the standard deviation. For all tasks, the learning curves do not differ significantly, suggesting the e-prop implementation is accurate.

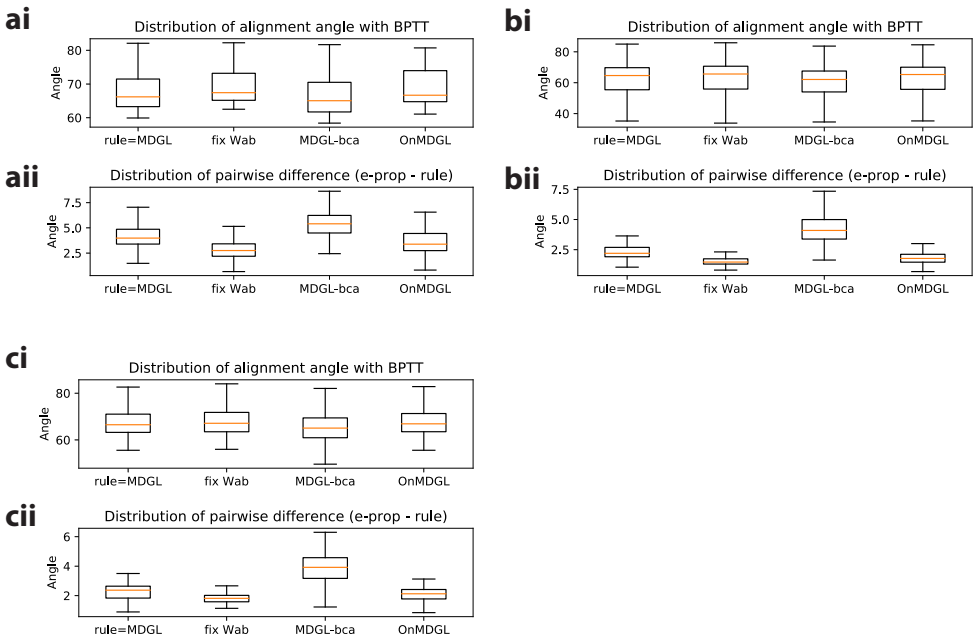

Figure S2.2: Alignment angle comparison shows that gradients approximated by MDGL are more similar (than e-prop) to the exact gradients. We quantify the similarity between approximated and exact gradients via the alignment angle, which describes the similarity in the direction of the two update vectors (Supplementary Note S2.3) for a) pattern generation, b) delayed match to sample and c) evidence accumulation tasks. In all top-panels (ai, bi, ci), the alignment angles between MDGL variants and BPTT are all less than 90°, which indicate that the approximated gradients are aligned with the exact gradient, despite the high-dimensionality of the update vectors. All bottom panel plots (aii, bii, cii) suggest that MDGL variants achieve smaller alignment angle (hence better alignment) with BPTT than e-prop does. To ensure a fair comparison, we examine the statistics of pairwise difference, so that the point on the loss landscape - where the comparison is done - is matched. This is achieved by training the network using BPTT across five different runs and sampling the approximated gradient once every 50 training iterations. Alignment analysis illustrated here is for recurrent weight gradients, and similar trends are observed for the input weights as well.

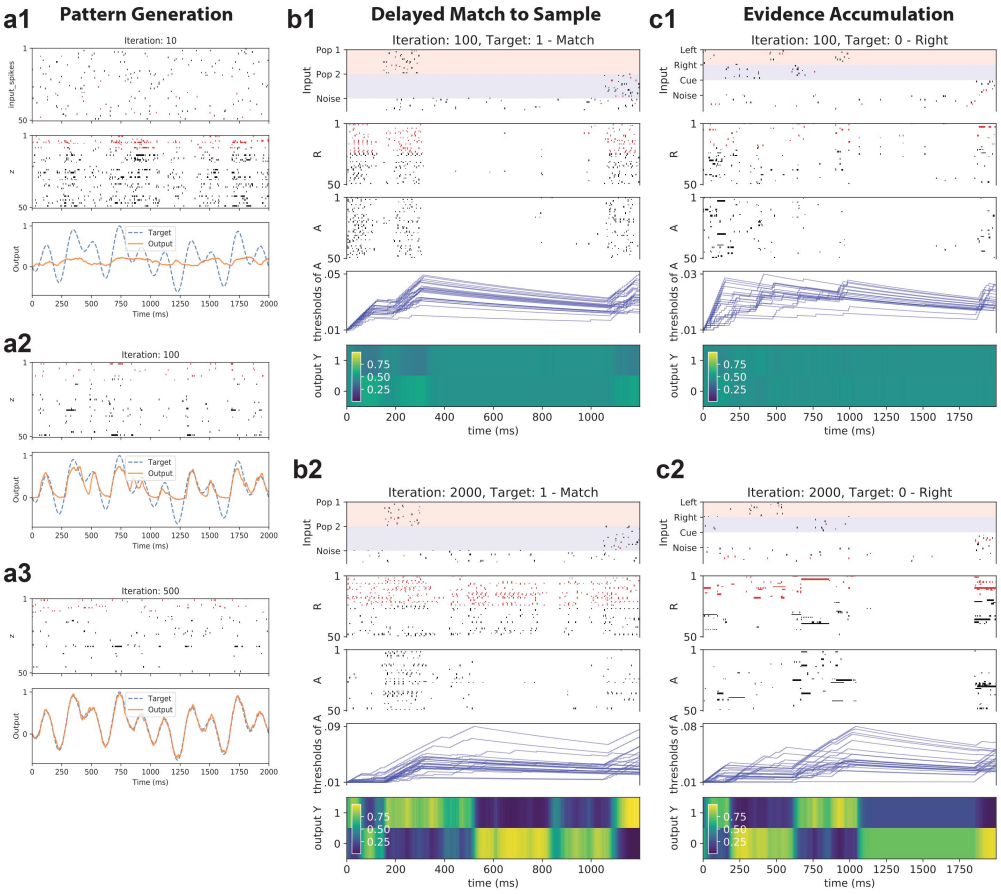

Figure S2.3: Network dynamics across multiple tasks investigated in Figure 2.3. a) Dynamics of the input, output and recurrent units are shown after 1, 100 and 500 iterations of training for the pattern generation task in Figure 2.3a using the MDGL method. Raster plots are shown for 50 selected sample cells, and E cells and I cells are color coded using black and red, respectively. All recurrent units have fixed thresholds for this task. Recurrent unit spikes are irregular throughout training. Network output approaches the target as training progresses. b) Network dynamics of an example trial after 100 and 2000 iterations of training for the delayed match to sample task in Figure 2.3b using MDGL. To emphasize the change in dynamics over training iterations, we used the same cue pattern for the illustrations. Again, E cells and I cells are color coded using black and red, respectively. For this task, both recurrent units with adaptive threshold (labeled as A) and without (labeled as R) are involved [4]. Threshold dynamics of sample neurons are illustrated. The network makes the correct prediction with greater confidence as training progresses. c) Network dynamics (Input spikes, recurrent unit spikes and readout) of an example trial after 100 and 2000 iterations of training for the evidence accumulation task in Figure 2.3c using MDGL. The network makes the correct prediction with greater confidence as training progresses. For all methods, results were obtained without using stochastic rewiring, which would allow for random formation of new synapses in each experience (Deep R) [5].

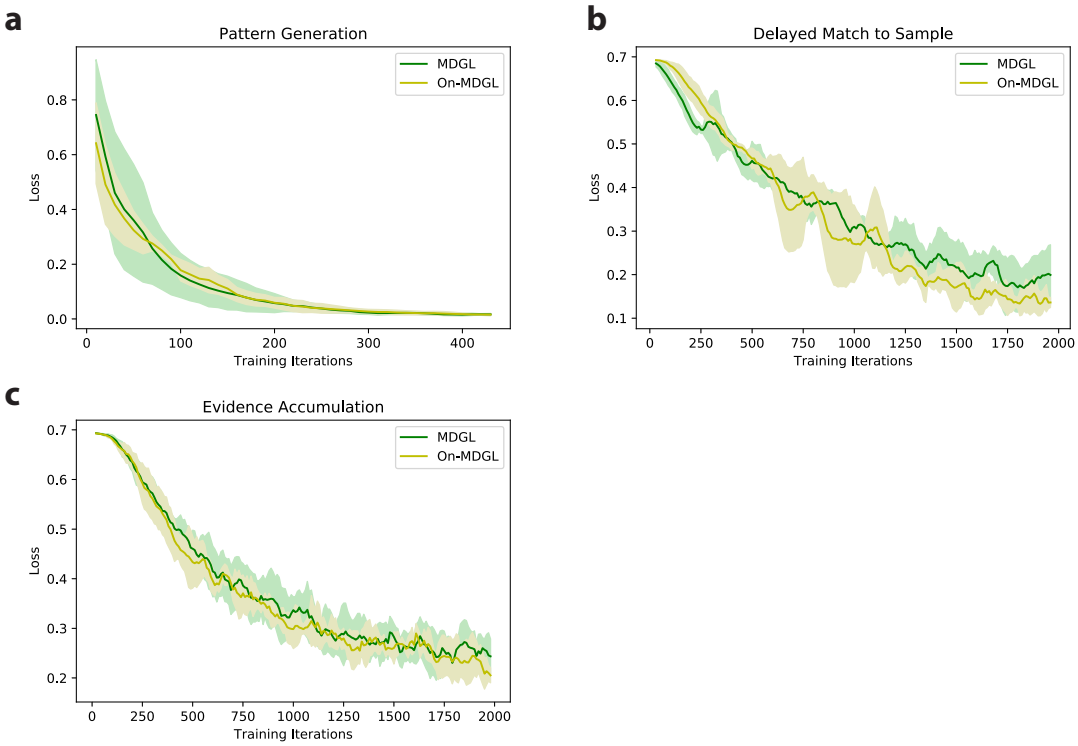

Figure S2.4: No significant degradation in performance observed for the online approximation of MDGL in Eq. S2.3. The naive implementation of activity dependent modulatory emission (Eq. 2.24 in Figures 2.3–2.4 depends on future errors, as explained in Supplementary Note S2.1 when the readout is leaky (depends on past output value). Therefore, we introduce an approximation in Eq. S2.3 for online implementation of MDGL. To check if this approximation leads to significant degradation in performance, learning curves are plotted for a) pattern generation, b) delayed match to sample and c) evidence accumulation tasks. For all tasks, there is no significant deviation in learning curves between MDGL and our proposed online approximation (On-MDGL).

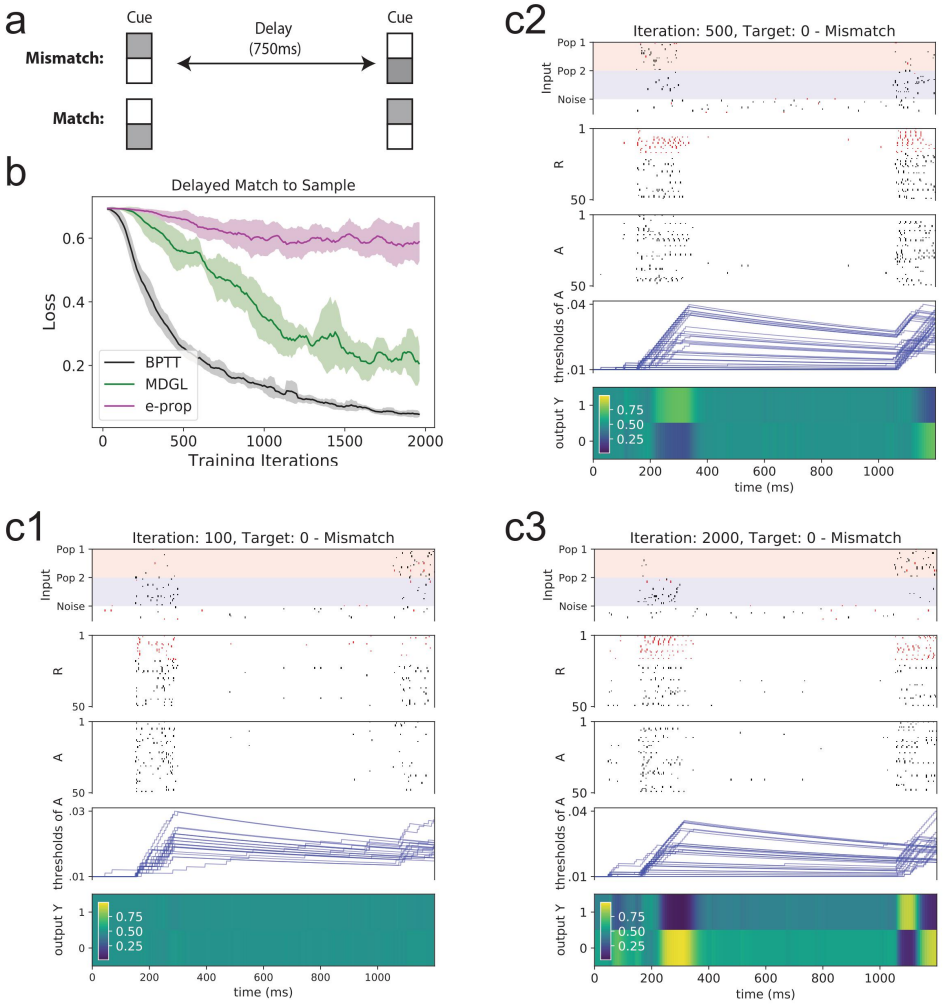

Figure S2.5: Similar observations for alternative task parameters in the task of Figure 2.3b. The delayed match to sample task in Figure 2.3b is repeated here, but with nonzero firing rates for the second cue alternative. As before, two input populations take on two different firing statistics to represent the two cue alternatives, and the agent is tasked with determining if the cue presented before and after the delay period correspond to the same cue alternative. The rates of these two populations are provided in Supplementary Note S2.3. The plotting conventions are the same as those of Figure 2.3 and Figure S2.3, except that a larger network is used (Supplementary Note S2.3), and 50 units are selected for the raster plots. The same conclusions as Figure 2.3b and S2.3b are observed here: comparing the performance of e-prop with the MDGL method suggests that the addition of cell-type-specific modulatory signals improves learning outcomes; the network makes the correct prediction with greater confidence as training progresses.

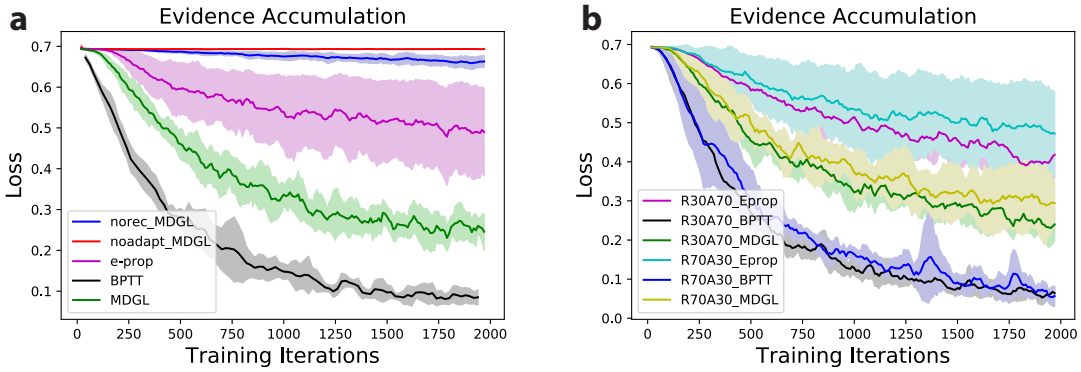

Figure S2.6: Threshold adaptation analysis for the evidence accumulation task in Figure 2.3c. a) Both threshold adaptation and recurrence are needed for successful completion of the task. MDGL trained on a network without ALIF cells (red) or with recurrent connections removed (blue) shows little decrease in loss over training iterations. b) Simulating with 70 LIF to 30 ALIF cells (R70A30) as well as 30 LIF to 70 ALIF cells (R30A70) led to similar ordering of performance for different learning methods as the default (50 LIF to 50 ALIF cells).

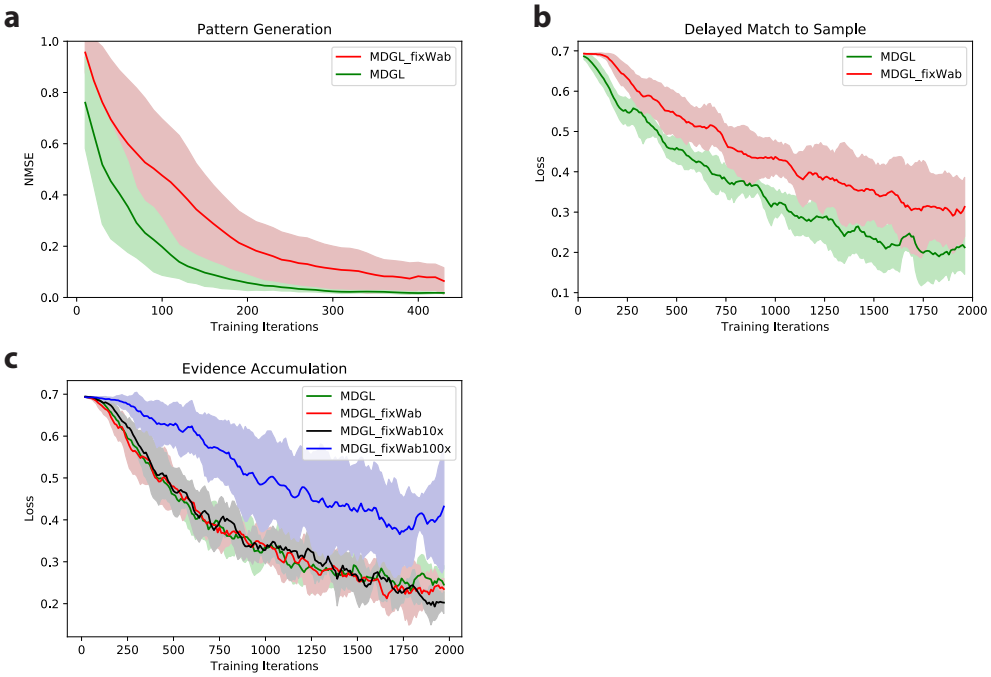

Figure S2.7: Comparing the learning curves for default MDGL versus MDGL with random fixed cell-type-specific receptor effcacies. As explained in Methods (Eq. 2.23), cell-type-specific receptor affinities were taken to be average connection weights. To explore the sensitivity of the learning performance to imprecise receptor affinities, here, the magnitude of each receptor affinity $w_{\alpha\beta}$ (for $\alpha \in \{I, E\}$, $\beta \in \{I, E\}$) is taken as the absolute value of a Gaussian random variable with zero mean and variance of $\frac{1}{\sqrt{N}}$ (N is the number of neurons), while the sign is kept as the neuron sign of type β; $w_{\alpha\beta}$ is randomized upon each initialization and fixed throughout the training. We observe a relatively mild degradation in performance for the pattern generation task and delayed match to sample task using this fixed random $w_{\alpha\beta}$ (labeled as MDGL_fixWab in each panel). For the evidence accumulation task, we did not observe any degradation even when the randomly generated $w_{\alpha\beta}$ was multiplied by a factor of 10 (labeled as MDGL_fixWab10x). A factor of 100 (labeled as MDGL_fixWab100x) pushes the network outside of an efficient operating range for the evidence accumulation task, suggesting that different tasks exhibit different degrees of tolerance to deviations in receptor affinities. This comparison is done for a) pattern generation, b) delayed match to sample and c) evidence accumulation tasks.

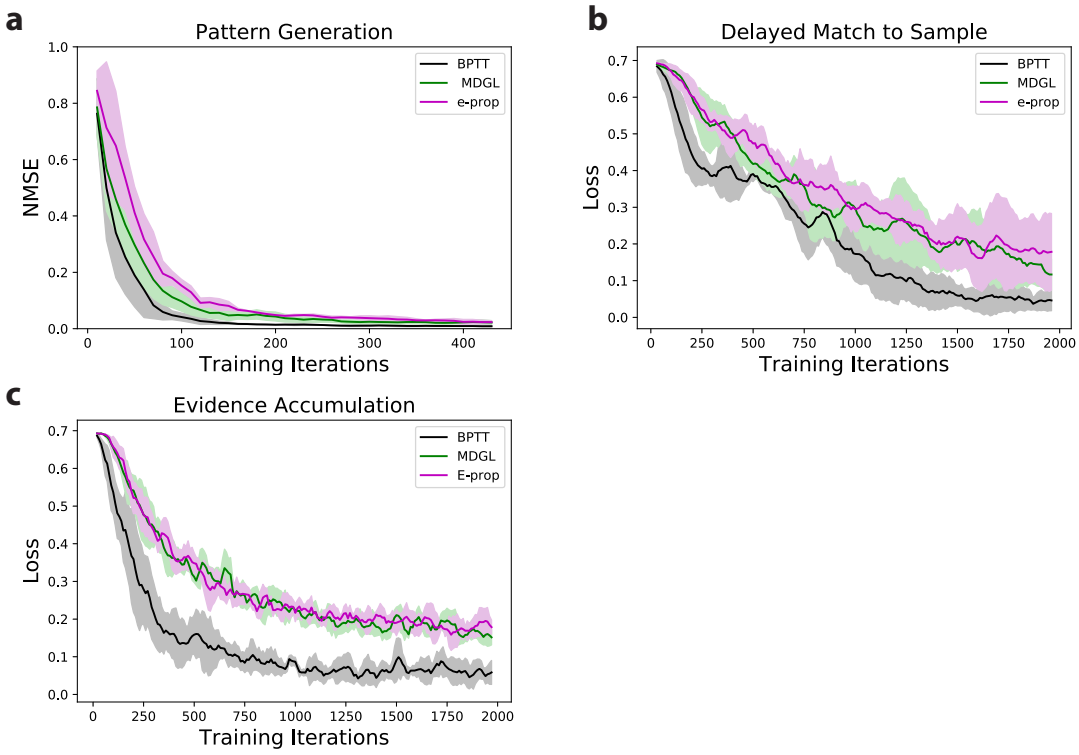

Figure S2.8: Advantage of MDGL disappears when the readout projection is dense. Through the paper, we instantiate the network with sparse connectivity to resemble neuronal circuits. Here, we repeat the tasks using networks with a dense connection to the readout (while keeping connections sparse within the recurrent network), and we find the competitive advantage of MDGL goes away. However, neurons are rarely fully connected to the readout in biological neural circuits. When the readout layer is sparse, not all neurons receive top-down learning signals in the e-prop formulation, and MDGL allows these neurons to receive learning signals as well.

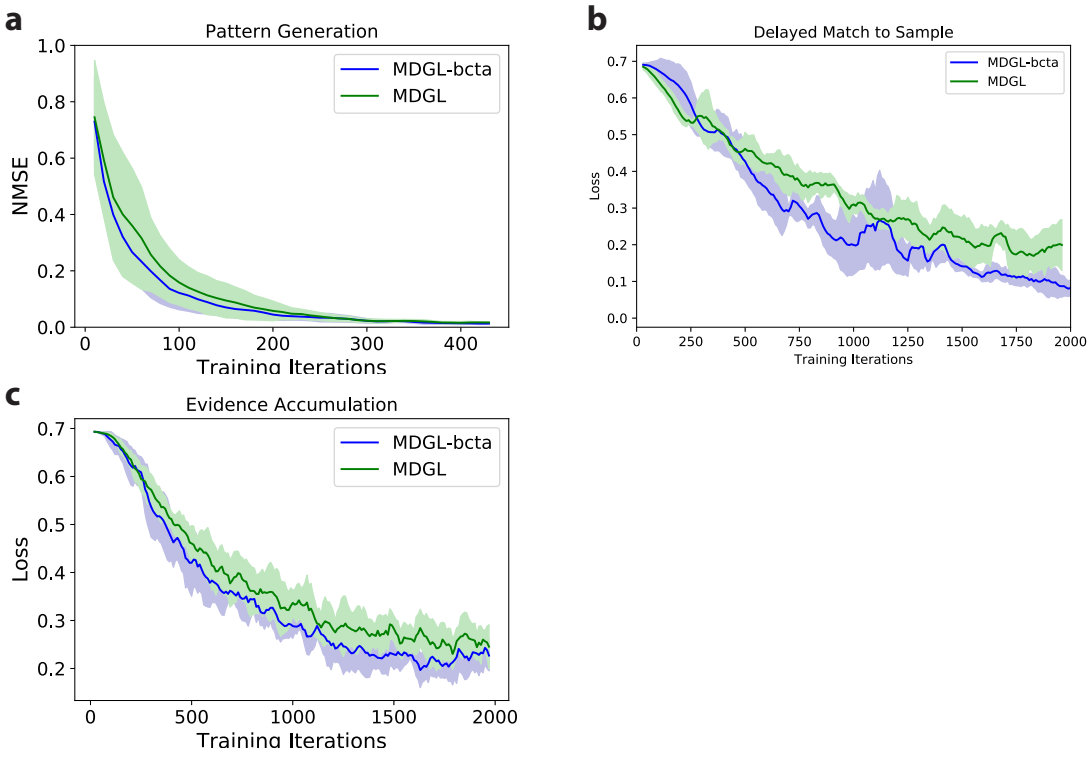

Figure S2.9: Effectiveness of minimal cell-type discretization. MDGL-bcta: MDGL before cell type approximation. In Methods, we defined the cell-type-specific receptor affinities to be average connection weights between cell types, i.e. $w_{\alpha\beta} =< w_{jp} >_{j\in\alpha,p\in\beta}$ (Eq. 2.23); we considered a minimal implementation of modulatory types mapped to the two main cell classes ($\alpha, \beta \in \{E, I\}$). We compare its learning performance to MDGL before cell type approximation as in Figure 2.5 (MDGL-bcta), which does not involve cell-type approximation, i.e. $w_{\alpha\beta} = w_{jp}$ (each cell is its own type). A cartoon illustration of MDGL-bcta can be found in Figure 2.5c. This comparison is applied to a) pattern generation, b) delayed match to sample and c) evidence accumulation tasks. The proximity of learning curves for MDGL-bcta and MDGL illustrates the effectiveness of such cell-type discretization.

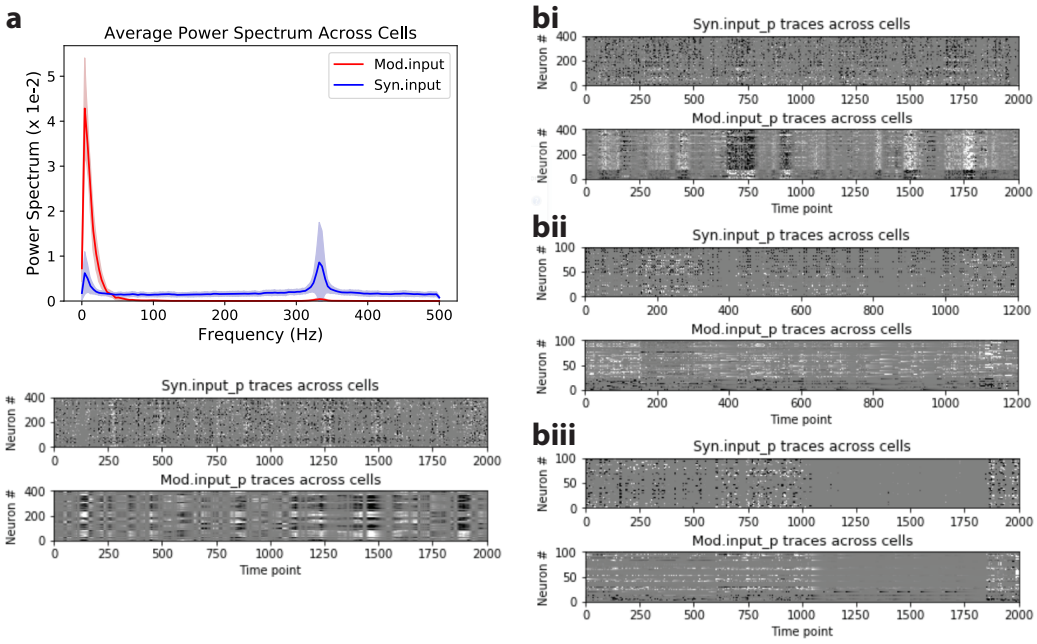

Figure S2.10: Slowness in modulatory signaling for the online approximation of MDGL. As explained in Figure S2.4, we proposed and tested an online implementation of activity-dependent modulatory release in Eq. S2.3. a) We repeat the spectral analysis in Figure 2.4a-c for this online implementation, i.e. replace $a_{j,t}$ in $Mod.input_p$ (Eq. 2.21) with online activity-dependent modulatory release $\bar{a}_{j,t}$ defined in Eq. S2.3. The observations here match those of Figure 2.4, where modulatory input is significantly slower than synaptic input. We note that the analysis here is done on the pattern generation task only, because for the other two tasks, the error signal is not available until the end of the trial, making the modulatory input too short (see Eq. S2.3) for any meaningful spectral analysis. Phenomenologically, the "slowness" of modulatory signaling can be explained by the modulatory input being a weighted summation of slow changing leaky outputs and low-pass filtered activity (Eq. S2.3). Raw synaptic and modulatory input (across time steps and cells) used for the frequency analysis are included beneath the frequency analysis plot. b) Raw synaptic and modulatory input traces for the frequency analysis in Figure 2.4a-c for i) pattern generation, ii) delayed match to sample and iii) evidence accumulation tasks.

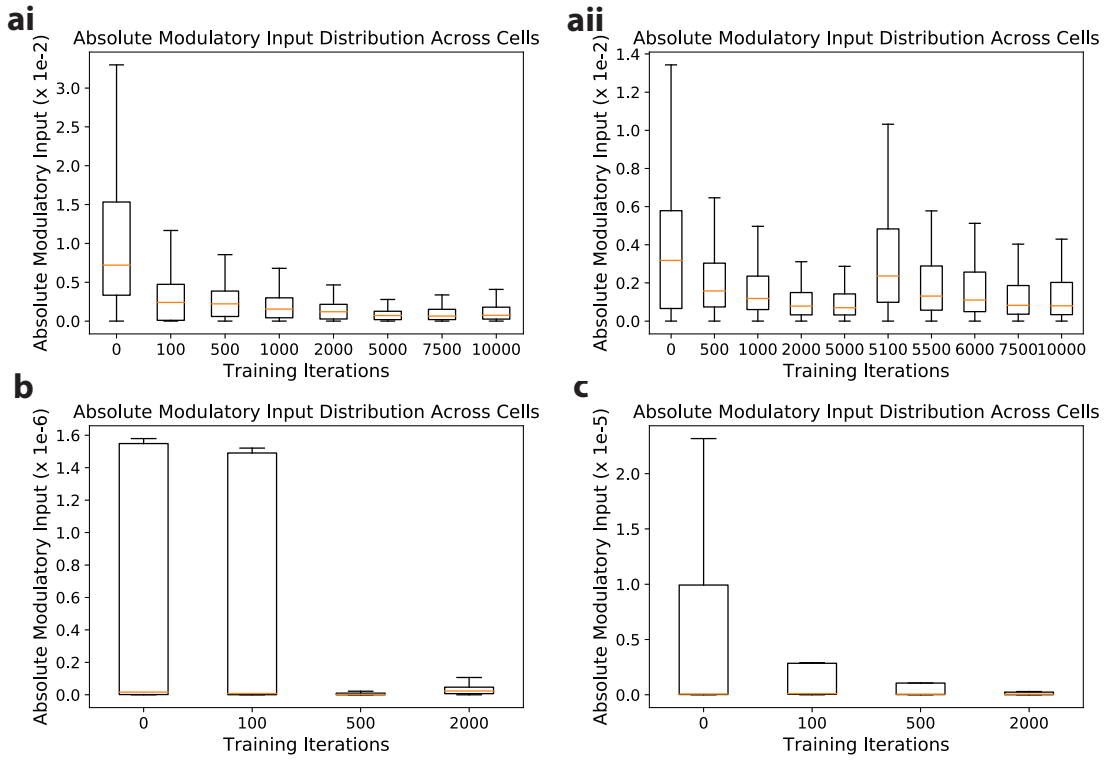

Figure S2.11: Cell-type-specific modulatory signaling level decreases over training iterations. Box plots for absolute cell-type-specific modulatory input distribution across cells show that modulatory signalling level drops over training iterations for ai) pattern generation, b) delayed match to sample and c) evidence accumulation tasks. aii) The target for the pattern generation task was changed after 5000 iterations, which resulted in a rapid increase in modulatory input immediately after the change, and a progressive decrease as training continued. This agrees with the prediction of our learning rule (Eq. 2.20, 2.24 and Supplementary Note S2.2) that cell-type-specific signaling carries information pertaining to top-down learning signals so that their levels can reflect the learning progress. To capture the magnitude of the signaling level, the absolute modulatory input for each cell p is defined similar to $Syn.input_p$ in Eq. 2.21, but with the absolute value of each factor; $\sum_{\alpha \in C} |w_{\alpha\beta}| \sum_{j \in \alpha, p \to j} |\bar{a}_{j,t}|$ ($\bar{a}_{j,t}$ defined in Eq. S2.3).

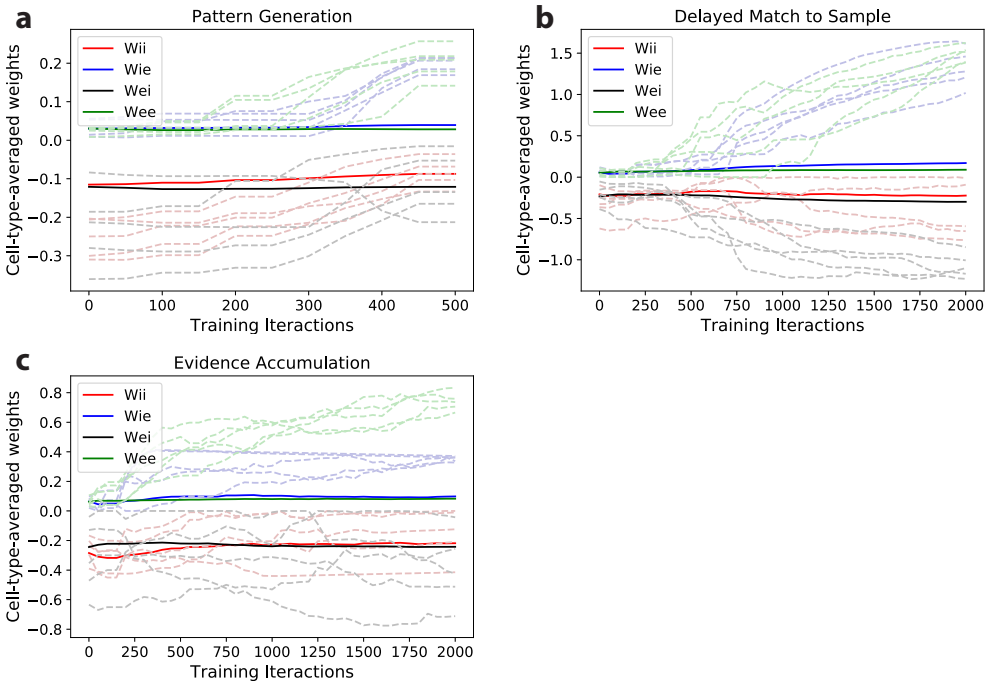

Figure S2.12: Average connection weight between types drifts slowly over training. Four average synaptic connection weight values (E to E, E to I, I to E and I to I) are illustrated in solid lines for the three tasks investigated. Several sample individual weights with large changes are illustrated in faint dashed lines, which can deviate significantly from the type averages. As explained earlier in Methods (Eq. 2.23) as well as Figure S2.9, these four average connection weights are used as our minimal implementation of cell-type-specific receptor affinities, $w_{\alpha\beta}$ with $\alpha, \beta \in \{E, I\}$ (see Eq. 2.23 and Eq. 2.20). How tightly the individual synaptic weights and cell-type-specific receptor affinities co-adapt may be explored in future work. Figure S2.7 suggests that the effect of imprecise GPCR affinities on the performance is task-dependent.

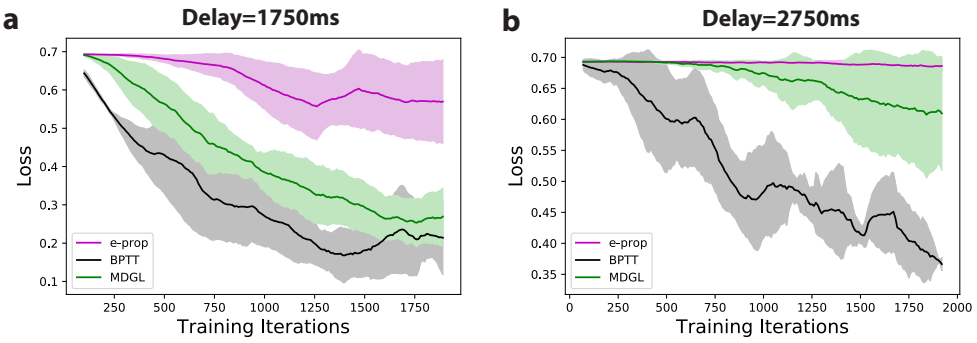

Figure S2.13: The performance of MGDL degrades when the delay period length is increased beyond a certain point. Here, the delay period is parametrically modulated for the delayed match to sample task. In a), the delay period is increased by 1000ms to 1750ms, and the network can still learn via MDGL. In b), the delay period is increased by 2000ms to 2750ms, and the network struggles to learn with MDGL while it still learns via BPTT. All other parameters (notably threshold adaptation time constant) are fixed in these simulations. It is interesting to note that worsened learning performance with increased delay period has previously been observed in animal experiments [6].

Chapter 3

BIOLOGICALLY-PLAUSIBLE BACKPROPAGATION THROUGH ARBITRARY TIMESPANS VIA LOCAL NEUROMODULATORS

3.1 Introduction

Recurrent connectivity is a hallmark of neuronal circuits. While this feature enables rich and flexible computation, mechanisms enabling efficient task learning in large circuits remain a central problem in neuroscience and artificial intelligence research. Fundamentally, the problem stems from the fact that potentially all history of all neurons, including synaptically far away ones, can affect neuronal activity and contribute to task output. Motivated by the success of gradient descent learning, several biological learning models approximate the exact gradient in recurrent neural networks using known biological processes and gain insights into computational principles of how the brain might learn [4, 9, 11].

By ignoring dependencies beyond a few recurrent steps — thus severely truncating the gradient computational graph — these existing models succeed in representing their approximate gradient-based update rule as a combination of terms that resemble known synaptic physiological processes: "eligibility trace", which maintains a fading memory of coincidental activation of presynaptic and postsynaptic neurons [8, 38, 42, 108, 109], combined with a third “modulatory” factor — top-down learning or “reward” signals [37, 79, 85, 86], for which dopamine is a prominent candidate [110]. Despite the impressive performance of such approximations on a variety of tasks, truncating relevant credit information results in a significant performance gap compared to algorithms using exact gradient information; backpropagation-through-time (BPTT) and real time recurrent learning (RTRL) [4, 9, 11, 71].

How might neural circuits account for long-term recurrent interactions to assign credit (or blame) to neuronal firing that happened arbitrary steps before the presentation of reward? Given the rich repertoire of dynamical and signaling elements in the brain, one avenue could be to examine biological processes that have been underexplored in existing

models. Dopamine — whose cellular actions are exerted by activation of G protein-coupled receptors (GPCRs), which can greatly impact STDP — is not the only neuromodulator involved in learning [111–114]. More recently, transcriptomic studies have uncovered strong evidence for many other neuromodulatory pathways throughout the brain that also act via the activation of GPCRs, leading to similar downstream actions as dopamine [62, 93]. This suggests that, similar to dopamine, they could also play a role in shaping synaptic credit assignment. A conspicuous family within these pathways is neuropeptide signaling because peptidergic genes are densely and abundantly expressed in the forebrains of divergent species, including the human, in a cell-type-specific fashion [115], suggesting widespread interaction between synaptic and peptidergic modulatory networks for synaptic credit assignment [3, 11]. Moreover, intra-cortical expression allows neuropeptides to potentially carry information **local** to the cortical network, cell type specificity enables sculpted signals for different recipient cells, and their diffusive nature could enable communication between neurons that are not synaptically connected. Finally, peptidergic signals have timescales much longer than the time scales for axonal propagation of action potentials or synaptic delays [91]. Taken together, these properties make cell-type-specific local neuromodulation seem promising for propagating credit signal over multiple recurrent steps. Developing explicit computational principles of how these local modulatory elements could propagate credit signal over arbitrary recurrent steps could advance our understanding of biological learning and may inspire more efficient low-complexity bio-inspired learning rules.

Motivated by the question above as well as shortcomings of gradient approximations based on severe temporal truncations, we investigate how biological credit signals could be propagated through arbitrary recurrent steps via widespread cell-type-specific neuropeptidergic signaling [3, 62]. While Ref. [11] recently introduced a framework exploiting properties of neuropeptidergic signaling for temporal credit assignment, similar to [9] and [4], their approach performs severe temporal truncation of the error gradient and does not consider credit propagation beyond disynaptic connections. Our main contributions are summarized as follows:

- We derive a theory that provides mechanism and intuition for the effectiveness of

synapse-type-specific modulatory backpropagation (through time) weights (Theorem 1).

- We develop a model predicting how modulatory signaling could be the basis for biological backpropagation through **arbitrary** time spans (Figures 3.1 and 3.2). Unlike temporal truncation-based approximations [4, 9, 11], our model enables each neuron to receive filtered (rather than precise) credit signals regarding its contribution to the outcome via neuromodulation.

- We demonstrate the effectiveness of modulatory signaling via synapse-type-specific, rather than synapse-specific, modulatory weights on learning tasks that involve temporal credit assignment (Figure 3.4). In particular, we demonstrate an **online** learning setting where weights are updated causally and in real-time (Figure 3.5).

- We also derive a low-complexity in silico implementation of our algorithm suitable for **online learning** (Proposition 1).

3.2 Related works

Neuromodulatory factors in synaptic plasticity: One of the most fundamental learning rules, Hebbian plasticity, attributes lasting changes in synaptic strength and memory formation to correlations of spike timing between particular presynaptic and postsynaptic neurons [116, 117]. However, multiple experimental and theoretical investigations now indicate that the Hebbian rule alone is insufficient. First, there have been numerous suggestions that some persistent "eligibility trace" must exist to bridge the temporal gap between correlated firings at millisecond timescales and behavioral timescales lasting seconds [8, 38, 42, 90, 108, 118]. Moreover, impacts of correlated spike timing must be augmented by one or more additional modulatory factors to steer weight updates toward desired outcome [8, 22, 40, 42, 56, 59, 86, 110, 119–123]. This is commonly known as learning with three factors. A prominent candidate for such a modulatory factor is dopamine [37]. Dopamine influences receiving neurons via activation of G protein-coupled receptors (GPCRs), which can regulate membrane excitability and key parameters of synaptic plasticity rules.

Besides dopamine, recent transcriptomic evidence has uncovered genes encoding a plethora of other neuromodulatory pathways throughout the brain, including neuropeptide signaling genes that are abundantly expressed in the forebrains of tetrapods, including the human [62, 115]. Like dopamine, their cellular actions are exerted by the activation of G protein-coupled receptors (GPCRs), which can persistently modulate Hebbian synaptic plasticity [86–88]. This suggests that they too, could play a role in shaping cortical learning and synaptic credit assignment. Moreover, nearly all neurons express one or more neuropeptide signaling gene [62], which suggests a dense interplay between synaptic and peptidergic modulatory networks to shape synaptic credit assignment [3].

Approximate gradient descent learning: Standard algorithms for gradient descent learning in recurrent neural networks (RNNs), real time recurrent learning (RTRL) and backpropagation through time (BPTT), are not biologically-plausible [78] and have vast computational and storage demands [70]. However, multiple studies have shown that learning algorithms that approximate the gradient, while mitigating some of the problems of exact gradient computation, can lead to satisfactory learning outcomes [12, 78, 97]. In feedforward networks, plenty of biologically-plausible learning rules have been proposed and demonstrated impressive performance that rival backpropagation on many different tasks [12, 22, 59, 78–84].

For efficient online learning in RNNs, approximations to RTRL have been proposed [9, 71–76]. For instance, a recent influential study [4] conceived how the three-factor learning framework could approximate the gradient. Among the biologically-plausible proposals [4, 9, 11], approximations have mainly been based on temporal truncations of the gradient computation graph and because of that, their ability to learn dependencies across arbitrary recurrent steps has been limited. Lastly, non-truncation-based approximations have been proposed outside of bio-plausible research [72, 73]. For further discussions on related algorithms, please refer to Appendix S3.3.

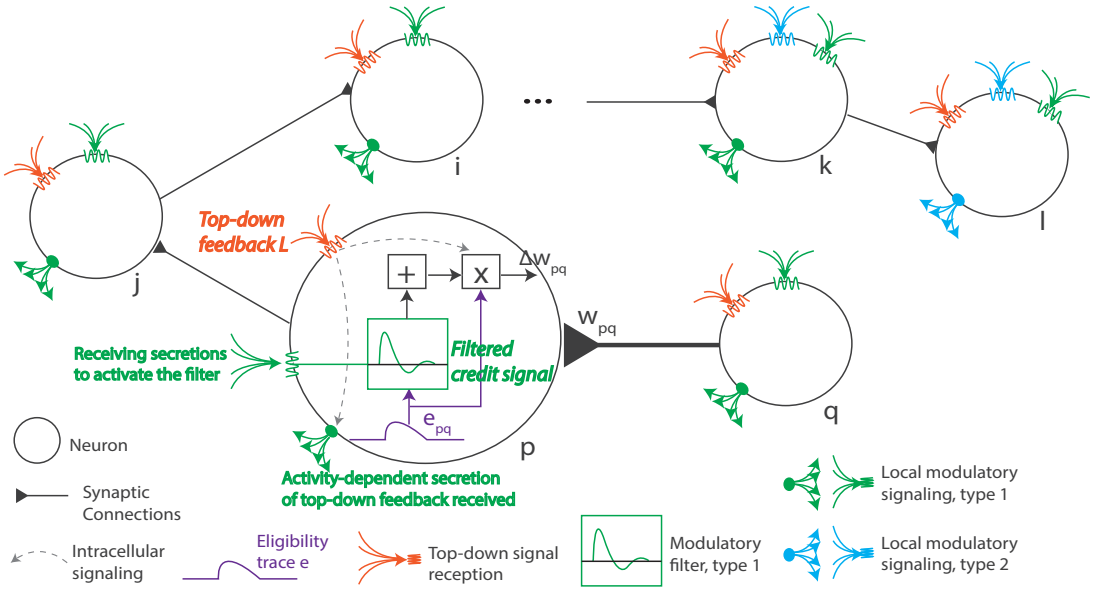

Figure 3.1: Biologically-plausible temporal credit assignment via modulatory and synaptic message passing. In addition to established biological learning ingredients (eligibility traces and a top-down learning signal [7, 8]), synapse-type-specific local modulatory networks may also be involved in weight updates [3]. Our learning rule, ModProp, conceives the action of participating modulators on receiving cells as a convolution of the eligibility trace with causal, time-invariant, cell-type-specific filters. Each circle represents a neuron and the synaptic weight of interest is W_{pq}; we illustrate the cellular processes of postsynaptic neuron p. Our derivation (Supp. Section S3.1 and Theorem 1) predicts that the modulatory signal each neuron receives can represent a filtered credit signal regarding how its past firings (arbitrary steps back) contribute to the task outcome.

3.3 Results

3.3.1 Learning rule overview

Biological plausibility is a guiding principle in developing our model. Hence, the model choices are based either on established constraints of neurobiology such as the locality of synaptic transmission, causality, and Dale's Law, or on emerging evidence from large-scale datasets, such as the cell-type-specificity of certain neuromodulators [62] or the hierarchical organization of cell types [93]. We consider a discrete-time rate-based RNN similar to the form in [124] with observable states, i.e. firing rates, as z_t at time t, and the corresponding internal states as s_t. W denotes the recurrent weight matrix with $(pq)^{\text{th}}$ entry W_{pq} representing the synaptic connection strength between presynaptic neuron q and postsynaptic neuron p. See Supp. Section S3.1 for the neuron model.

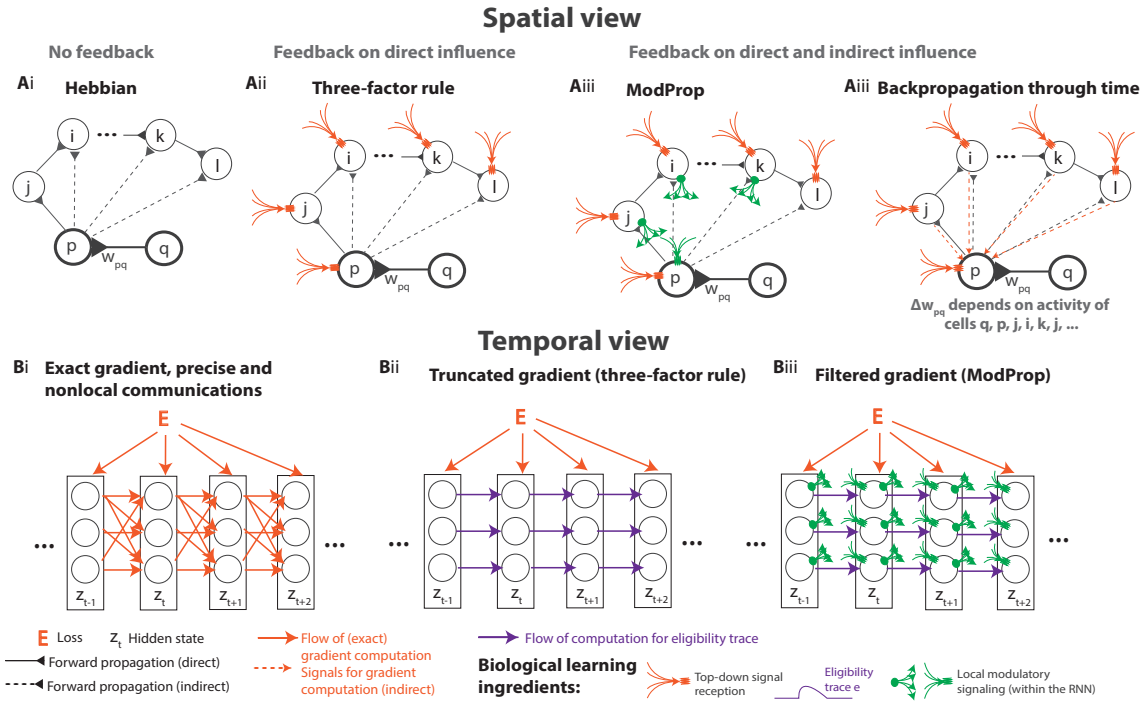

Figure 3.2: Local modulatory signaling for gradient estimation. A) A spatial view of learning rules for updating weight W_{pq}. (i) Hebbian learning, where weight update depends only on pre-/post-synaptic activities. (ii) Three-factor learning [4, 7, 9], which updates weights using additional top-down learning signals, severely truncates the exact gradient. (iii) ModProp also accounts for (filtered) distant feedback information delivered through synapse-type-specific neuromodulation; (iv) BPTT computes the exact gradient: weight update involves nonlocal information, i.e. activities of indirectly connected units. B) A temporal view. Bi) BPTT propagates the precise intercellular dependencies in an acausal manner. Bii) Three-factor learning rule neglects all the intercellular dependencies in the temporal propagation of the credit signal. Biii) ModProp **approximates** such spatiotemporal dependencies through local neuromodulatory signals (Eq. 3.6). ModProp approximates the exact gradient by assuming similar connectivity among cells of the same type, and filtering the indirect effects on loss from neurons that are potentially many synapses away. Figure 3.6 provides a summary of approximations made by ModProp.

Gradient descent learning in RNNs: RNNs are typically trained by gradient descent learning on the task error (or negative reward) E. However, the two equivalent factorizations of error gradient in RNNs, BPTT and RTRL, both involve nonlocal information that is inaccessible to neural circuits. This is due to recurrent connectivity: a synaptic weight W_{pq} can affect loss E through many other neurons in the other network in addition to its pre- and postsynaptic neurons. To see this more concretely, the following is the exact gradient at

time t using RTRL factorization, on which we base our approximation for online learning:

$$\frac{\mathrm{d}\,E}{\mathrm{d}\,W_{pq}}|_t = \sum_j \frac{\partial E}{\partial s_{j,t}} \frac{\mathrm{d}\,s_{j,t}}{\mathrm{d}\,W_{pq}} \tag{3.1}$$

$$\frac{\mathrm{d}\,s_{j,t}}{\mathrm{d}\,W_{pq}} = \frac{\partial s_{j,t}}{\partial W_{pq}} + \sum_l \frac{\partial s_{j,t}}{\partial s_{l,t-1}} \frac{\mathrm{d}\,s_{l,t-1}}{\mathrm{d}\,W_{pq}} = \frac{\partial s_{j,t}}{\partial W_{pq}} + \frac{\partial s_{j,t}}{\partial s_{j,t-1}} \frac{\mathrm{d}\,s_{j,t-1}}{\mathrm{d}\,W_{pq}} + \underbrace{\textstyle\sum_{l \neq j} W_{jl} \frac{\partial z_{l,t-1}}{\partial s_{l,t-1}} \frac{\mathrm{d}\,s_{l,t-1}}{\mathrm{d}\,W_{pq}}}_{\text{depends on all weights } W_{jl}}, \tag{3.2}$$

where ∂ denotes direct dependency and d accounts for all (direct and indirect) dependencies, following the notation in [4]. (See Appendix S3.1.2 for further details on the notation.) While $\frac{\partial E}{\partial s_{j,t}}$, which considers only the direct contribution of the internal state of neuron j at time t to the loss, is easy to compute, the factor $\frac{\mathrm{d}\,s_{j,t}}{\mathrm{d}\,W_{pq}}$ is a memory trace of all inter-cellular dependencies and requires $O(N^3)$ memory and $O(N^4)$ computations. This makes RTRL expensive to implement for large networks. Moreover, the last factor $\frac{\mathrm{d}\,s_{j,t}}{\mathrm{d}\,W_{pq}}$ poses a serious problem for biological plausibility: the nonlocal terms in Eq. 3.2 requires knowledge of all other weights in the network to update the weight W_{pq}. Existing biologically-plausible solutions to this problem apply severe truncations: references [9] and [4] completely ignore the third nonlocal term (Figure 3.2), whereas reference [11] restores terms within one recurrent step through local modulatory signaling but truncates further terms.

ModProp approximation overview: Derivation of gradient descent-based weight updates involving neuromodulatory signaling from Eq. 3.1 suggests two approximations to Eq. S3.4 and 3.2 to move beyond severe truncations of the exact gradient while retaining biological plausibility. **Approximation 1** replaces the activation derivative with a constant:

$$\frac{\partial z_{j,t}}{\partial s_{j,t}} \approx \mu \quad \text{(approximation 1)}, \tag{3.3}$$

where μ represents the average activity of neurons in a ReLU network. This approximation assumes stationarity in neuron activity and uncorrelatedness of such activity with a small subset of synaptic weights, as explained in derivations leading up to Appendix Eq. S3.14 and Eq. S3.21 (Methods in Appendix S3.1). While neuronal activity and synaptic weights are not necessarily uncorrelated, considering that a single neuron may have thousands of synaptic partners, the activity of the neuron or its time derivative is typically weakly correlated to

any one synaptic weight. This also indicates that such approximation might work better for large networks with many neurons and synaptic partners for each neuron, as is the case for biological neural networks. We take advantage of this phenomenon in our model and ignore these weak correlations. This approximation enables the filter taps (Eq. 3.6) to be **time-invariant**, a property likely required for biological plausibility. We also define S, the arbitrary number of credit propagation steps, and it will become clear later that this corresponds to the number of modulatory filter taps. With this, the estimated gradient becomes:

$$\frac{\mathrm{d}\,E}{\mathrm{d}\,W_{pq}}|_t \approx \frac{\partial E}{\partial z_{p,t}} e_{pq,t} + \sum_j \frac{\partial E}{\partial s_{j,t}} \sum_{s=1}^{S} (W^s)_{jp}\, \mu^{s-1}\, e_{pq,t-s} \tag{3.4}$$

where $e_{pq,t} \approx \frac{dz_{p,t}}{dW_{pq}}$ ($e_{pq,t}$ is defined precisely in Appendix Eq. 2.25) can be interpreted as a persistent Hebbian "eligibility trace" [8, 38, 108] that keeps a fading memory of past coincidental pre- and postsynaptic activity [4]. Here, $(W^s)_{jp}$ represents the $(jp)^{\text{th}}$ entry of W^s, W raised to the power of s, and $(W^1)_{jp} = W_{jp}$. Details of the derivation can be found in Appendix S3.1.

It is important to note that **Approximation 1** is applied only to the nonlocal gradient terms, i.e. exact activation derivative is used during the computation of the eligibility trace that is local to the pre- and postsynaptic neurons. Also, we treat μ as a hyperparameter in our simulations and explain how this is tuned in Appendix S3.6. One should note that tuning the hyperparameter μ properly is important, because the estimated gradient can explode when μ is too large and ModProp approaches the three-factor rule when μ is too small. Future work involves improving ModProp with an adaptive μ for better numerical stability and accuracy.

Approximation 2 replaces **synapse-specific** feedback weights with **type-specific** weights:

$$(W^s)_{jp} \approx (W^s)_{\alpha\beta} \quad \text{(approximation 2)}. \tag{3.5}$$

Here, cell j belongs to type α, cell p is of type β and C denotes the set of cell types. (e.g., $W_{\alpha\beta} = \mathbb{E}_{j\in\alpha, p\in\beta}[W_{jp}]$, $\alpha, \beta \in C$.) This approximation is due to the type-specific nature of modulatory channels [62]. We call these modulatory weights synapse-type-specific (as

opposed to cell-type-specific) to emphasize the connectivity-based grouping. Details of how these modulatory weights were obtained can be found in Appendix S3.1.

ModProp: filtering credit signals via local neuromodulation: Substituting **Approximation 1** and **Approximation 2** into Eq. S3.4 and 3.2 leads to the ModProp update:

$$\Delta W_{pq}|_{ModProp} \propto L_p \times e_{pq} + \left(\sum_{\alpha \in C} \left(\sum_{j \in \alpha} \mathrm{L}_j \text{ activity}_j \right) \times F_{\alpha\beta} \right) * \mathrm{e}_{pq},$$
$$F_{\alpha\beta,s} = \mu^{s-1} (W^s)_{\alpha\beta}, \tag{3.6}$$

where L and e denote top-down learning signal and eligibility trace, respectively. We postulate that $F_{\alpha\beta}$ represents type-specific filter taps of GPCRs expressed by cells of type β to precursors secreted by cells of type α; $*$ is the convolution operation with S as the number of filter taps. Note that the matrix powers $(W^s)_{\alpha\beta}$ appearing in the values of different filter taps $F_{\alpha\beta,s}$ could be genetically pre-determined as part of the cell identity and optimized over evolutionary time scales. (Appendix Figure S3.1 shows successful learning using fixed modulatory weights.) Details of a biological interpretation of Eq. 3.6 can be found in Appendix S3.4.1. Briefly, the secretion of top-down (TD) learning signals can selectively activate a biochemical process at the post-synaptic neuron, which can then act as a temporal filter on the eligibility trace.

Observing that neurons of the same type demonstrate consistent properties across a range of features (e.g. connectivity, physiology, gene expression) [62, 99], we use two cell types with consistent wiring, neuromodulation, and type of synaptic action (excitatory/inhibitory) in our relatively simple models.

In summary, we propose a synaptic weight update rule, where the eligibility trace is compounded not only with top-down learning signals – as in modern biologically-plausible learning rules [8, 42] – but also with local modulatory pathways through convolution (Figure 3.1). This modulatory mechanism allows the propagation of credit signals through an arbitrary number of recurrent steps.

3.3.2 Properties of ModProp

Remark. *The biologically plausible implementation (Eq. 3.6) of ModProp complexity scales as $O(SN^2)$ per time step t.*

Here, N and S denote the number of recurrent units and the number of filter taps. As seen in Eq. 3.4, the number of filter taps corresponds to the number of recurrent steps for which the credit information is propagated. We also present an alternative implementation with potentially lower cost later in Proposition 1. Details for cost analysis and biological interpretation can be found in Appendix S3.4.1.

We next show through Theorem 1 that learning with synapse-type-specific weights leads to loss reduction at every step on average. The mechanistic intuition behind Theorem 1 is that, in the presence of a statistical connection between synaptic weights (forward path) and modulatory weights (feedback path), the modulatory feedback signal each neuron receives can be a good estimate of how its activity contributes to the overall task error, which can be used for effective tuning of incoming synaptic weights to reduce the error. Figure S3.5 compares the angle with the exact gradient across different learning algorithms to demonstrate that the direction of the approximate gradient computed by ModProp is similar to (aligned with) the direction of the exact gradient, thereby reducing the loss.

In the following, we define the "residual weights" away from the cell-type averages as $\epsilon_{ij} := W_{ij} - W_{\alpha\beta}$ and $(\epsilon_s)_{ij} := (W^s)_{ij} - (W^s)_{\alpha\beta}$, where $i \in \alpha, j \in \beta$, $\alpha, \beta \in C$. We consider these terms as stochastic so that the circuit output and the eligibility traces are also stochastic, as functions of ϵ_s. We consider the cell type averages and the output connection strengths as deterministic. (Merely modifying the uncorrelatedness assumption below would extend our results to stochastic output connection strengths.) Below, $\mathbb{E}$ is short for $\mathbb{E}_\epsilon$, $\Delta E|_t$ denotes the change in the task error at time t and $\Delta E|_{pq,t}$ denotes the contribution of the synapse (pq) to it.

Theorem 1. *Consider linear RNNs $\mathcal{N}$, $\widehat{\mathcal{N}}$ with weight matrices W, $\widehat{W}$, respectively and identical architectures. Let $\widehat{W}_{ij} = W_{\alpha\beta}$, $\forall i \in \alpha, j \in \beta$. Assume a small enough learning rate such that the remainder of the first-order Taylor expansion of loss is negligible. For network $\mathcal{N}$, if $\mathbb{E}[\epsilon_{ij}] = 0$, $(\epsilon_s)_{ik}$ and ϵ_{kj} are uncorrelated for $i \neq j$, and ϵ_s and $(y_t - y_t^*)^2 e_{pq,t'} e_{pq,t-s}$ are*

uncorrelated for any s, t, t', then $\mathbb{E}[\Delta E|_{pq,t}] \leq 0$ and $\mathbb{E}[\Delta E|_t] \leq 0$. Moreover, $\mathbb{E}[\Delta E|_t] < 0$ if gradient descent is possible for network $\widehat{\mathcal{N}}$. (Proof is in Appendix S3.5.)

Note that the uncorrelatedness assumption above is relatively mild because any single connection strength is typically a very poor predictor of network activity, especially as the network size grows. Note also that gradient update can introduce a drift in residual weights, requiring a similar drift in cell-type averages for the strict inequality to be applicable over multiple update steps.

3.3.3 Simulation results

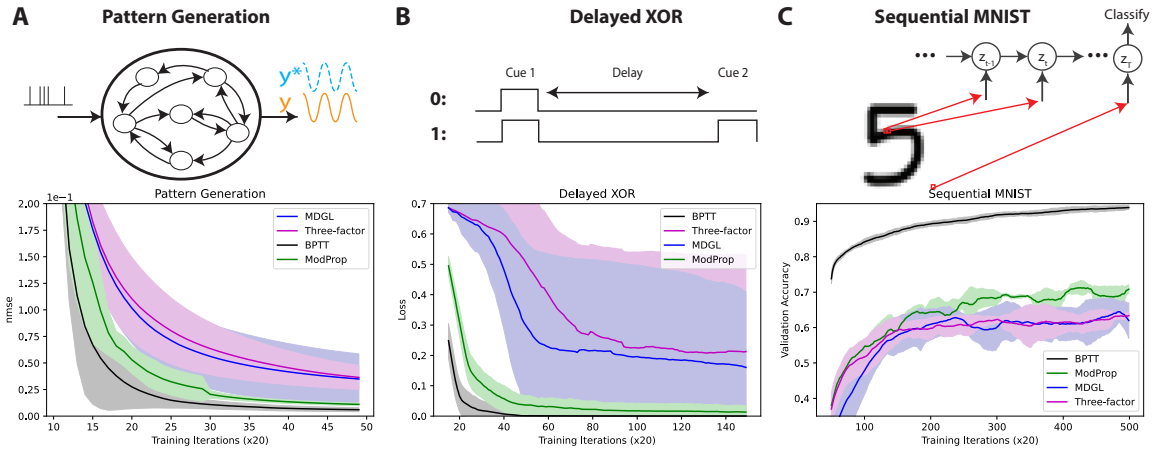

Figure 3.3: Modulatory signaling of credit information on long-term recurrent interactions can improve learning outcomes. ModProp improves the learning performance over existing bio-plausible rules. This experiment examines the performance due to Approximation 1 (Eq. 3.3) before any cell-type approximation of modulatory weights (Eq. 3.5). A) Learning to produce a time-resolved target output pattern. B) A delayed XOR task, where the network determines if two cue alternatives — the presence or absence of input represented by 1 or 0 — match or mismatch after a delay, requiring memory via recurrent activity. c) Pixel-by-pixel MNIST task [10]. Note that this task is unlikely to be solved effectively by humans. (See text.) Consistent with the original MDGL paper [11], we also find that MDGL confers little advantage over the three-factor rule (e-prop) under dense connectivity. Solid lines/shaded regions: mean/standard deviation of loss curves across five runs.

To test the ModProp formulation, we study its performance in well-known tasks involving temporal processing: pattern generation, delayed XOR, and sequential MNIST. We compare the learning performance of ModProp with the state-of-the-art biologically-plausible learning rules (MDGL [11] and e-prop [4] (labeled as "three-factor"), as well as BPTT to provide a lower bound on task error. Note, BPTT and RTRL both compute the exact gradient, so they

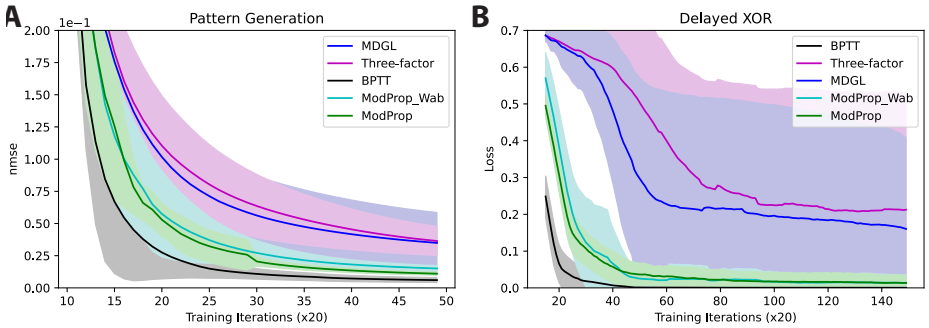

Figure 3.4: Efficient learning with type-specific modulatory weights. In addition to Approximation 1 studied in Figure 3.3, this figure investigates the effect of Approximation 2 (labeled as ModProp_Wab), which uses type-specific, rather than synapse-specific feedback weights for signaling credit information (Eq. 3.5). Here, ModProp_Wab uses only two modulatory types mapped to the two main cell classes. The cell type approximation does not result in any significant performance degradation in A) the pattern generation task and B) the delayed XOR task. This analysis is not done for the sequential MNIST task, where neurons were not divided into E and I types. Solid lines/shaded regions: mean/standard deviation of loss curves across five runs.

should be identical in terms of performance. Here, we chose BPTT due to computational efficiency. Consistent with [11], we also found MDGL to confer little advantage over e-prop when all neurons are connected to the readout (Figure 3.3), so we focus our analysis on that case.

We first study pattern generation with RNNs, where the aim is to produce a one-dimensional target output, generated from the sum of five sinusoids, given a fixed Gaussian input realization. We change the target output and the random input along with initial weights for different training runs, and show the learning curve in Figure 3.3A across seven such runs. By using a densely connected network, we observe that MDGL confers little advantage over the three-factor rule (e-prop), as reported in the original paper [11]. Moreover, communicating longer indirect effects, despite being filtered, leads to improved learning outcomes, as demonstrated by the superior performance of ModProp over MDGL.

Next, to study how RNNs learn to process discrete cues that impact delayed outcomes, we consider a delayed XOR task: two cue alternatives, 1 or 0, are encoded by the presence/absence of input. The network is trained to remember the first cue and learn to compare it with the second cue delivered at a later time to determine if the two cues match. Figure 3.3B illustrates the learning curves for this task. We observe the same general conclusion as the

previous task. Some learning curves have a standard deviation, which indicates that the network struggled to learn the task with these rules for some seeds; based on the standard deviation of the curves, it seems possible for ModProp to perform similarly as three-factor for some seeds, but the focus here is to examine performance across many runs. In Appendix Figure S3.3, we also examined the learning performance of the same set of learning rules for a longer delay period. Interestingly, the performance of ModProp approaches that of BPTT for this task and the previous one, further closing the gap between artificial network training and biological learning mechanisms in performing credit assignment over a long period.

Finally, we study the pixel-by-pixel MNIST [10] task, which is a popular machine learning benchmark. Although it is not a task that the brain would solve well (i.e., humans would struggle to predict the digit with only one pixel presented at a time), we investigate it to test the limits of spatiotemporally filtered credit signals for tasks that demand temporally precise input integration. Figure 3.3C illustrates the learning curves for this task. While the performance ordering of learning rules is still the same as in previous tasks, we observe a wider gap between ModProp and BPTT. Since the time-invariant filter approximation (Eq 3.3) restricts the spatiotemporal resolution of the credit signal, this is in line with our expectation that ModProp will struggle with tasks that demand highly precise spatiotemporal integration, such as the pixel-by-pixel MNIST task that even the brain would struggle to solve.

As a proof-of-concept study, we initially focused our analysis on the approximation performance by imposing time-invariant filter taps (Eq. 3.3). Next, in Figure 3.4, we investigate the learning performance using type-specific rather than synapse-specific feedback modulatory weights (Eq. 3.5). These type-specific weights were calculated using weight averages as in [11]. Later, we repeat these simulations using fixed random modulatory weights in Appendix Figure S3.1; we also demonstrate the superior performance of ModProp with fixed random modulatory weights for the "copy task" [13] in Appendix Figure S3.4. Little performance degradation is observed for only two modulatory types mapped onto the two main cell classes, indicating the effectiveness of cell-type discretization. Other than the sequential MNIST task, cells are divided into two main cell classes — with 80% of the cells being excitatory (E) and 20% being inhibitory (I) — and obeyed connection sign constraints.

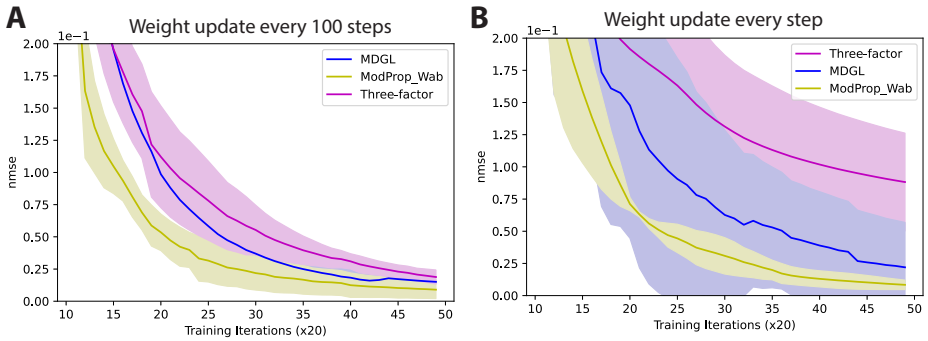

Figure 3.5: Superior performance of ModProp in an online learning setting. We investigate an online learning version of the pattern generation task, where weights are updated either A) every 100 time steps or B) every single step. Here, ModProp uses the efficient online learning implementation derived in Proposition 1. Plotting conventions follow those of previous figures.

Lastly, we demonstrate the advantage of our weight update rule in an online learning setting of the pattern generation task, where weights are updated in real time (Figure 3.5). For that, we also derive a cost and storage efficient in silico implementation of ModProp in Proposition 1.

Proposition 1. *ModProp has an online in-silico (not necessarily biologically-plausible) implementation with $O(CN^2)$ storage and $O(C^2N^2)$ computational complexity, where C is the number of cell types. (Proof is in Appendix S3.4.2.)*

3.4 Discussion

A central question in the study of biological and artificial intelligence is how the temporal credit assignment problem is solved in neuronal circuits [57]. Motivated by recent genomic evidence on the widespread presence of local modulatory networks [62, 115], we demonstrated how such signaling could complement Hebbian learning and global neuromodulation (e.g., via dopamine) to achieve biologically plausible propagation of the credit signal through arbitrary time steps. Going beyond the scalar feedback provided by global top-down modulation, our study proposes how detailed vector feedback information [56] can be delivered in neuronal circuits. Instead of the severe temporal truncations of the gradient information proposed by the state-of-the-art [4, 9, 11], ModProp offers a framework where the full temporal feedback signal can be received, albeit via low-pass filtering at the post-synaptic neuron due to

specificity at the level of neuronal types, and not individual neurons. Moreover, predictions generated by ModProp on the role of a family of signaling molecules (e.g. neuropeptides) could potentially be tested experimentally: physiology of multiple individual cells can be monitored in modern neurobiology experiments [3, 103]. Blocking certain peptidergic receptors of the neurons that are involved with learning a task and comparing the performance to that without blocking can provide a test for the role of peptidergic communication.

While feedback alignment [12] addresses the weight transport problem in feedforward networks, it is not clear in RNNs which biological pathways would implement **temporal** feedback. Our model suggests that such pathways could come from synapse-type-specific local neuromodulation. In addition to improved performance compared to the state-of-the-art across all of our simulations, our theoretical and experimental results show that ModProp can be implemented efficiently for online learning. Figure 3.6 briefly mentions some of the connections of ModProp to feedback alignment. Together, these findings suggest that synapse-type-specific local modulatory signaling could be a neural mechanism for propagating credit information over more than just a few recurrent steps.

Among the many future directions, a natural extension could be to investigate the performance of ModProp across a broader range of tasks. This could include situations where the assumptions in deriving the rule (e.g., stationarity of activity) are severely violated. This might improve the approximations introduced here. Similarly, we observed a significant gap between ModProp and BPTT for the pixel-by-pixel MNIST task that demands precise temporal integration of input (Figure 3.3C), but we discussed earlier that this is a challenging task for the brain. Additionally, this study focuses on dense networks with ReLU activation; future directions include investigating two biologically relevant paradigms: sparse connectivity and a diverse set of activation functions (spike-based in particular). Although ModProp can be applied in theory to temporal credit assignment over an arbitrary duration, the presented learning rule accounts for dependencies at every single step (as for BPTT). This means that, similar to BPTT, it is ill-suited for very long-term credit assignment [125]. An interesting line of research attends to the issue of extracting relevant, rather than full, memories [125, 126]. Investigating how our approximations could potentially be combined with memory sparsification techniques to perform very long-term biologically-plausible credit

assignment can be fruitful [125].

While our paper advances the basic science of learning and we do not foresee immediate societal impacts of our work, its benefits to both neuroscience and deep learning research could have long-term (positive or negative) societal impacts. It is hard to overstate the philosophical implications of understanding how the brain works (and, in particular, learns). Moreover, such understanding could guide us toward curing certain diseases of the brain [127, 128]. Nevertheless, in the same vein, such future tools could also enable abuse if left unregulated, particularly raising concerns regarding information privacy [129, 130] from potential brain hacking. On the machine learning side, our method can be considered a low-cost alternative to the ubiquitous BPTT algorithm. In this sense, our algorithm or a future method that builds on it can make tasks that are currently approachable by just a few wealthy entities available to many more practitioners. On the flip side, as with many other capable machine learning tools, such entities are also capable of utilizing low-cost learning to conquer even harder tasks, which is potentially a reason for concern. Finally, data-driven tools will ultimately reflect the various biases in their training data and our method is no exception.

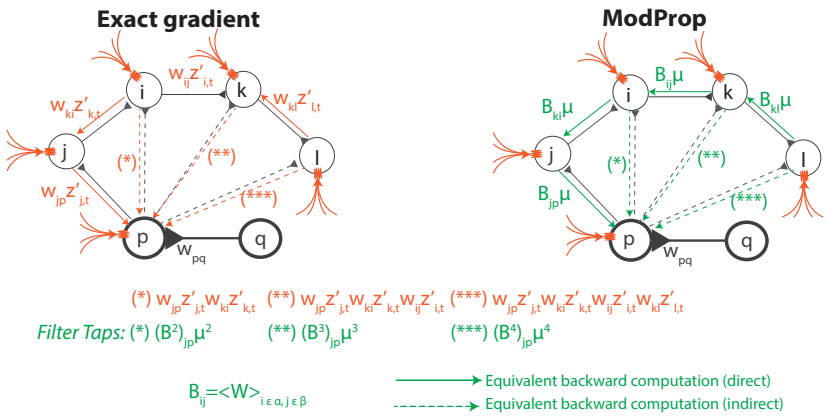

Figure 3.6: ModProp brings two approximations to the nonlocal gradient terms. First, ModProp uses type-specific feedback weights (modulatory weights), rather than cell-specific feedback weights (Eq. 3.3). Second, ModProp approximates the activation derivative (Eq. 3.5). Similar to feedback alignment for feedforward networks [12], different weights are used during the backward pass than the forward pass due to the well-known weight transport problem. This, however, won't be adequate for RNN settings. On top of the type-specific feedback weights approximation, time-invariant activation derivative approximation was also applied for time-invariant filtering (as explained in the texts surrounding Eq. 3.4 and Eq. 3.6). Any unspecified symbols in the illustration were defined in Figure 3.2.

S3.1 Methods

S3.1.1 Network Model

We consider a discrete-time implementation of a rate-based recurrent neural network (RNN) similar to the form in [124]. We denote the observable states, i.e. firing rates, as z_t at time t, and the corresponding internal states as s_t. The dynamics of those states are governed by

$$s_{j,t+1} = \eta\, s_{j,t} + (1-\eta)\left(\sum_{l\neq j} W_{jl}\, z_{l,t} + \sum_{p} W^{\mathrm{IN}}_{jm}\, x_{m,t+1}\right)$$
$$z_{j,t} = ReLU(s_{j,t}), \tag{S3.1}$$

where $\eta = e^{-dt/\tau_m}$ denotes the leak factor for simulation time step dt and membrane time constant τ_m, W_{lj} denotes the weight of the synaptic connection from neuron j to l, W^{IN}_{jm} denotes the strength of the connection between the m^{th} external input and neuron j and x_t denotes the external input at time t. Threshold adaptation is not used here in order to focus on capacity of the temporal credit propagation mechanism. We focused on ReLU activation due to its wide adoption in both deep learning and computational neuroscience communities; as discussed, we leave extension to other activation functions (spike-based in particular) for future work.

The readout y is a linear transformation of the hidden state

$$y_{k,t} = \sum_{j} W^{\mathrm{OUT}}_{kj} z_{j,t} + b^{\mathrm{OUT}}_{k}, \tag{S3.2}$$

where W^{OUT}_{kj} denotes the strength of the connection from neuron j to output neuron k, b^{OUT}_{k} denotes the bias of the k-th output neuron.

We quantify how well the network output matches the desired target using loss function E:

$$E = \begin{cases} \frac{1}{2}\sum_{k,t}(y^*_{k,t} - y_{k,t})^2, \text{ for regression tasks} \\ -\sum_{k,t} \pi^*_{k,t} log \pi_{k,t}, \text{ for classification tasks} \end{cases} \tag{S3.3}$$

where $y^*_{k,t}$ is the time-dependent target, $\pi^*_{k,t}$ is the one-hot encoded target and $\pi_{k,t} = \mathrm{softmax}_k(y_{1,t}, \ldots, y_{N_{OUT},t}) = \exp(y_{k,t}) / \sum_{k'} \exp(y_{k',t})$ is the predicted category probability.

S3.1.2 Gradient descent learning in RNNs

Notation for Derivatives: There are two types of computational dependencies in RNNs; direct and indirect dependencies. We distinguish direct dependencies versus all dependencies (including indirect ones) using partial derivatives (∂) versus total derivatives (d), respectively.

Without loss of generality, consider a function $f(x, y)$, where y itself may depend on x. The partial derivative ∂ of f at x_0 considers y as a constant, and evaluates as $\frac{\partial f(x,y)}{\partial x}|_{x_0,y(x_0)}$; i.e. the derivative calculation only considers how x directly affects f. The total derivative d, on the other hand, may not treat y as a constant and evaluates as $\frac{df(x,y)}{dx} = \frac{\partial f(x,y)}{\partial x}|_{x_0,y(x_0)} + \frac{\partial f(x,y)}{\partial y}|_{x_0,y(x_0)} \frac{\partial y}{\partial x}|_{x_0}$; i.e. the derivative calculation also takes into account how x can indirectly affect f through y.

As an example in our network, variable W_{pq} can impact state $s_{p,t}$ directly through Eq. S3.1, i.e. $\frac{\partial s_{p,t}}{W_{pq}} = (1 - \eta) z_{q,t-1}$. On the other hand, W_{pq} can also impact $s_{p,t}$ indirectly through other cells in the network: i.e. the dependence of $s_{p,t}$ on W_{pq} and all $s_{j,t'}$ ($t' < t$, $j \in \{1, ..., N\}$) affected by W_{pq} are taken into account for the derivative calculation, which leads to the recursive equation in Eq. 2.17.

Exact gradient computation and locality issue: In gradient descent learning, all weight parameters (input weights W^{IN}, recurrent weights W and output weights W^{OUT}) are adjusted iteratively according to the error gradient. This error gradient can be calculated with classical machine learning algorithms, backpropagation through time (BPTT) and real time recurrent learning (RTRL) [70], which uses different factorization but yield equivalent results. However, the BPTT factorization depend on future activity, which poses an obstacle for online learning and biological plausibility. Our learning rule derivation follows the RTRL factorization because it is causal.

RTRL factors the error gradient across time and space as

$$\frac{\mathrm{d}\,E}{\mathrm{d}\,W_{pq}}|_t = \sum_j \frac{\partial E}{\partial s_{j,t}} \frac{\mathrm{d}\,s_{j,t}}{\mathrm{d}\,W_{pq}} \tag{S3.4}$$

$$\frac{\mathrm{d}\,z_{j,t}}{\mathrm{d}\,W_{pq}} = h_{j,t} \frac{\mathrm{d}\,s_{j,t}}{\mathrm{d}\,W_{pq}}, \text{where } h_{j,t} := \frac{\partial z_{j,t}}{\partial s_{j,t}} \tag{S3.5}$$

$$\frac{\partial s_{j,t}}{\partial W_{pq}} = \delta_{jp}(1-\eta) z_{q,t-1} \tag{S3.6}$$

$$\frac{\partial s_{j,t}}{\partial s_{l,t-1}} = \begin{cases} \eta, & j = l \\ \frac{\partial s_{j,t}}{\partial z_{l,t-1}} \frac{\partial z_{l,t-1}}{\partial s_{l,t-1}} = (1-\eta) W_{jl} h_{l,t-1}, & j \neq l \end{cases} \tag{S3.7}$$

$$\begin{aligned} \frac{\mathrm{d}\,s_{j,t}}{\mathrm{d}\,W_{pq}} &= \frac{\partial s_{j,t}}{\partial W_{pq}} + \sum_l \frac{\partial s_{j,t}}{\partial s_{l,t-1}} \frac{\mathrm{d}\,s_{l,t-1}}{\mathrm{d}\,W_{pq}} \\ &= \frac{\partial s_{j,t}}{\partial W_{pq}} + \eta \frac{\mathrm{d}\,s_{j,t-1}}{\mathrm{d}\,W_{pq}} + (1-\eta) \underbrace{\textstyle\sum_{l \neq j} W_{jl} \frac{\partial z_{l,t-1}}{\partial s_{l,t-1}} \frac{\mathrm{d}\,s_{l,t-1}}{\mathrm{d}\,W_{pq}}}_{\text{depends on all weights } W_{jl}}. \end{aligned} \tag{S3.8}$$

following the derivative notation explained above. The factor $\frac{\partial E}{\partial z_{j,t}}$ in Eq. S3.4 can be interpreted as the top-down learning signal, which is defined as $L_{j,t} := \sum_k W_{kj}^{OUT}(y_{k,t} - y_{k,t}^*)$ for regression tasks [4]. It is straightforward to compute. However, the triple tensor $\frac{\mathrm{d}\,s_{j,t}}{\mathrm{d}\,W_{pq}}$ requires $O(N^3)$ memory and $O(N^4)$ computation costs. It keeps track of all the paths that $z_{j,t}$ can affect W_{pq} (for every j, p, q). Moreover, it poses a significant challenge to biological plausibility: updating each weight W_{pq} requires knowing all other weights W_{jl} (for every j and l) in the network, and that information should be inaccessible to neural circuits.

To address this, references [9] and [4] dropped the problematic terms so that the updates to weight W_{pq} would only depend on pre- and post-synaptic activity, and applied this truncation to train rate- and spike-based networks, respectively. However, such truncation results in limited performance.

S3.1.3 Derivation of ModProp

We ask how intercellular neuromodulation might communicate the expensive spatiotemporal dependency in the factor $\frac{\mathrm{d}\,s_{j,t}}{\mathrm{d}\,W_{pq}}$. Along with the interpretation of $L_{j,t} := \frac{\partial E}{\partial z_{j,t}}$ as top-down learning signal, let $e_{pq,t}$ denote the eligibility trace of coincidental activation between presynaptic cell q and postsynaptic cell p [4]. The following derivation leads to our learning

rule. The leak factor is omitted in the derivation below ($\eta = 0$) for readability, and we substitute Eq. 2.17 into Eq. S3.4 and repeatedly expand the expensive $\frac{\mathrm{d}\, s}{\mathrm{d}\, W}$ factor using Eq. 2.17:

$$\frac{\mathrm{d}\, E}{\mathrm{d}\, W_{pq}}|_t = \sum_j \frac{\partial E}{\partial z_{j,t}} h_{j,t} \left(\delta_{jp} z_{q,t-1} + \sum_l W_{jl} h_{l,t-1} \frac{\mathrm{d}\, s_{l,t-1}}{\mathrm{d}\, W_{pq}} \right) \tag{S3.9}$$

$$= \frac{\partial E}{\partial z_{p,t}} h_{p,t} z_{q,t-1} + \sum_j \frac{\partial E}{\partial z_{j,t}} h_{j,t} \sum_l W_{jl} h_{l,t-1} \frac{\mathrm{d}\, s_{l,t-1}}{\mathrm{d}\, W_{pq}} \tag{S3.10}$$

$$= \frac{\partial E}{\partial z_{p,t}} e_{pq,t} + \sum_j \frac{\partial E}{\partial z_{j,t}} h_{j,t} \sum_l W_{jl} h_{l,t-1} \left(\delta_{lp} z_{q,t-2} + \sum_k W_{lk} h_{k,t-2} \frac{\mathrm{d}\, s_{k,t-2}}{\mathrm{d}\, W_{pq}} \right) \tag{S3.11}$$

$$= \dots$$

$$\frac{\mathrm{d}\, E}{\mathrm{d}\, W_{pq}}|_t = \frac{\partial E}{\partial z_{p,t}} e_{pq,t} + \left(\sum_j \frac{\partial E}{\partial z_{j,t}} h_{j,t} W_{jp} \right) e_{pq,t-1} +$$

$$\sum_j \frac{\partial E}{\partial z_{j,t}} h_{j,t} \sum_{s=2}^{S} \sum_{i_1,\dots,i_{s-1}} W_{ji_1} W_{i_1 i_2} \dots W_{i_{s-1} p} h_{i_1,t-1} \dots h_{i_{s-1},t-s+1} e_{pq,t-s} \tag{S3.12}$$

$$\overset{(a)}{\approx} \frac{\partial E}{\partial z_{p,t}} e_{pq,t} + \left(\sum_j \frac{\partial E}{\partial z_{j,t}} h_{j,t} W_{jp} \right) e_{pq,t-1}$$

$$+ \sum_j \frac{\partial E}{\partial z_{j,t}} h_{j,t} \sum_{s=2}^{S} (W^s)_{jp}\, e_{pq,t-s} \frac{1}{N^{s-1}} \sum_{i_1,\dots,i_{s-1}} h_{i_1,t-1} h_{i_2,t-2} \dots h_{i_{s-1},t-s+1} \tag{S3.13}$$

$$= \frac{\partial E}{\partial z_{p,t}} e_{pq,t} + \sum_j \frac{\partial E}{\partial z_{j,t}} h_{j,t} \sum_{s=1}^{S} (W^s)_{jp}\, H(t,s)\, e_{pq,t-s}, \tag{S3.14}$$

where $H(t,1) = 1$, $H(t,s) = \frac{1}{N^{s-1}} \sum_{i_1,\dots,i_{s-1}} h_{i_1,t-1} h_{i_2,t-2} \dots h_{i_{s-1},t-s+1}$ for $s = 1, \dots, S$ and S, as explained later, is the number of filter taps. Again, we neglected the leak factor in the derivation for readability but included in the actual simulations. The only approximation step above, (a), is made by using a point estimate assuming that the W and h chains are uncorrelated and the central limit theorem applies. We note that in a linear network, all the activation derivatives h would be 1, making the approximation exact.

We expand on our explanation for step (a) approximation. We define a W-chain (of

length l) as

$$\prod_{\phi=1}^{l} W_{i_\phi i_{\phi+1}}, \tag{S3.15}$$

for any indices $i_1, \ldots, i_{l+1} \in \{1, ..., N\}$. Similarly, we define an h-chain (of length l') as

$$\prod_{\theta=1}^{l'} h_{j_\theta, t-\theta}, \tag{S3.16}$$

for any indices $j_1, \ldots, j_{l'} \in \{1, ..., N\}$. With these definitions, we call the W-chain $W_{i_1,...,i_{s-1}} = W_{ji_1} W_{i_1 i_2} \ldots W_{i_{s-1} p}$ and the h-chain $h_{i_1,...,i_{s-1}} = h_{i_1,t-1} \ldots h_{i_{s-1},t-s+1}$ uncorrelated if

$$\mathbb{E}_{i_1,...,i_{s-1}} W_{i_1,...,i_{s-1}} h_{i_1,...,i_{s-1}} = \mathbb{E}_{i_1,...,i_{s-1}} W_{i_1,...,i_{s-1}} \mathbb{E}_{i_1,...,i_{s-1}} h_{i_1,...,i_{s-1}}. \tag{S3.17}$$

Considering $W_{i_1,...,i_{s-1}}$ and $h_{i_1,...,i_{s-1}}$ as random i.i.d. samples indexed by $i_1, \ldots, i_{s-1}$, the central limit theorem states that

$$\sum_{i_1,...,i_{s-1}} W_{i_1,...,i_{s-1}} h_{i_1,...,i_{s-1}} \sim \mathcal{N}(N^{s-1}\mathbb{E}[W_{i_1,...,i_{s-1}} h_{i_1,...,i_{s-1}}], N^{s-1}\mathrm{Var}(W_{i_1,...,i_{s-1}} h_{i_1,...,i_{s-1}})) \tag{S3.18}$$

as the sum tends to infinity. Here, we simply use the i.i.d. assumption even though stronger versions of the Central Limit Theorem need weaker assumptions than i.i.d. When the W- and h-chains are uncorrelated, we take the mean of this distribution as a point estimate (note, however, the growing variance) to arrive at the following approximation:

$$\sum_{i_1,...,i_{s-1}} W_{i_1,...,i_{s-1}} h_{i_1,...,i_{s-1}} \approx N^{s-1}\mathbb{E} W_{i_1,...,i_{s-1}} h_{i_1,...,i_{s-1}} = N^{s-1}\mathbb{E} W_{i_1,...,i_{s-1}} \mathbb{E} h_{i_1,...,i_{s-1}}. \tag{S3.19}$$

Since $\mathbb{E} W_{i_1,...,i_{s-1}} = \frac{1}{N^{s-1}}(W^s)_{jp}$ (by the same application of central limit theorem as above) when $i_1, \ldots, i_{s-1}$ are distributed uniformly over valid index ranges, we conclude that

$$\sum_{i_1,...,i_{s-1}} W_{i_1,...,i_{s-1}} h_{i_1,...,i_{s-1}} \approx (W^s)_{jp} \frac{1}{N^{s-1}} \sum_{i_1,...,i_{s-1}} h_{i_1,...,i_{s-1}}, \tag{S3.20}$$

where the expectation of the h-chain is replaced by its empirical estimate.

To put this derivation in terms of biological components, we make the following further approximations. First, we link modulatory weights to type-specific GPCR efficacies, which

means they are type-specific, i.e. $(W^s)_{jp} \approx (W^s)_{\alpha\beta}$, for type-indices α and β in a set of possible classes C. Second, $H(t,s)$ should be time-invariant, i.e. $H(t,s) = H(s)$, since biological filter properties should not vary rapidly across time.

Interestingly, we observe that $H(s)$ is an average of activation history across time and cells (Eq. S3.14). In particular, when the activation function is ReLU, one can think of $H(s)$ as approximating the number of activation chains with length s (divided by the total number of possible chains). Thus, a crude starting point would be to assume first-order stationarity, i.e., assume the average activity level remains invariant ($\frac{1}{N}\sum_i h_{i,t} := \mu_t \approx \mu, \forall t$). Then

$$\begin{aligned} H(t,s) &= \frac{1}{N^{s-1}} \sum_{i_1,\ldots,i_{s-1}} h_{i_1,t-1} \ldots h_{i_{s-1},t-s+1} \\ &\overset{(a)}{=} \frac{1}{N}\sum_{i_1} h_{i_1,t-1} \frac{1}{N}\sum_{i_2} h_{i_2,t-2} \cdots \frac{1}{N}\sum_{i_{s-1}} h_{i_{s-1},t-s+1} \\ &\approx \mu^{s-1}, \end{aligned} \tag{S3.21}$$

where μ is a scalar constant and represents global average neuron activation. (a) is because in the case of ReLU activation (where activation derivative h is binary), total number of different activation chain combinations, $\sum_{i_1,\ldots,i_{s-1}} h_{i_1,t-1} \ldots h_{i_{s-1},t-s+1}$, would equal to the product of number of activation at each time step, $\sum_{i_1} h_{i_1,t-1} \sum_{i_2} h_{i_2,t-2} \ldots \sum_{i_{s-1}} h_{i_{s-1},t-s+1}$. This is because to choose a chain of activated neurons from all possible combinations, the number of possible indices to choose from for each step is equal to the number of activated neuron at that step. Indeed, intercellular signaling level is activity-dependent [91]. For implementation, μ can be treated as a hyperparameter or adapted on a separate timescale. It is also important to note that this activation derivative approximation is only applied to the term (in the equation below) additional to the e-prop term, $\frac{\partial E}{\partial z_{p,t}} e_{pq,t}$, for which the exact activation derivative is still used.

By substituting these further approximations into Eq. S3.14, the approximated gradient becomes:

$$\frac{\mathrm{d}\,E}{\mathrm{d}\,W_{pq}}|_t \approx \frac{\partial E}{\partial z_{p,t}} e_{pq,t} + \sum_{\alpha \in C} \left(\sum_{j \in \alpha} \frac{\partial E}{\partial z_{j,t}} h_{j,t} \right) \sum_{s=1}^{S} (W^s)_{\alpha\beta}\, \mu^{s-1}\, e_{pq,t-s}, \tag{S3.22}$$

and this leads to the ModProp update:

$$\Delta W_{pq}|_{ModProp} \propto L_p \times e_{pq} + (\sum_{\alpha \in C} \left(\sum_{j \in \alpha} \mathrm{L}_j h_j \right) \times F_{\alpha\beta}) * \mathrm{e}_{pq},$$
$$F_{\alpha\beta,s} = \mu^{s-1} (W^s)_{\alpha\beta}, \tag{S3.23}$$

where cell j is of type α, cell p is of type β and C denotes the set of cell types. L and e denote top-down learning signal and eligibility trace, respectively. Again, activation derivative h_j is closely linked to activity level of neuron j. F represents the modulatory filter, $F_{\alpha\beta} * \mathrm{e}_{pq} = \sum_{s=1}^{S} F_{\alpha\beta,s} \mathrm{e}_{pq,t-s}$ is the convolution operation with S as the number of filter taps, and scaling factor μ is a hyperparameter. For calculating the modulatory weights, the weights were calculated using matrix powers for $s > 1$. (See beginning of Theorem 1 proof.) For $s = 1$, we first examined $W^1_{\alpha\beta} =< W^1_{jp} >_{j \in \alpha, p \in \beta}$ in the main text. This assumes modulatory weights and synaptic weights co-adapt throughout training and to what extent they co-adapt in neural circuits is unclear. Thus, we also set modulatory weights to fixed random type-specific values and demonstrate the resulting learning performance in Appendix Figure S3.1. These fixed random type-specific modulatory weights were generated randomly from the distribution of averages of random initial weights. Thus, these fixed random type-specific modulatory weights would be close to the initial synaptic weight averages and could stay close depending on how much these synaptic weight averages change throughout training.

Eligibility trace implementation: We here explain the implementation of eligibility trace $e_{pq,t}$:

$$e_{pq,t} := \frac{\partial z_{p,t}}{\partial s_{p,t}} \epsilon_{pq,t}, \tag{S3.24}$$
$$\epsilon_{pq,t} = \frac{\partial s_{p,t}}{\partial w_{pq}} + \frac{\partial s_{p,t}}{\partial s_{p,t-1}} \epsilon_{pq,t-1}, \tag{S3.25}$$

which tracks the coincidence of postsynaptic activity $h_{p,t} = \frac{\partial z_{p,t}}{\partial s_{p,t}}$ and a low pass filtering of presynaptic activity stored in $\epsilon_{pq,t}$ ($\frac{\partial s_{p,t}}{\partial w_{pq}} = z_{q,t-1}$ and $\frac{\partial s_{p,t+1}}{\partial s_{p,t}} = \eta$ following Eq. S3.1). Reference [4] provides a comprehensive discussion on how eligibility traces can be interpreted as derivatives.

S3.2 Additional simulations

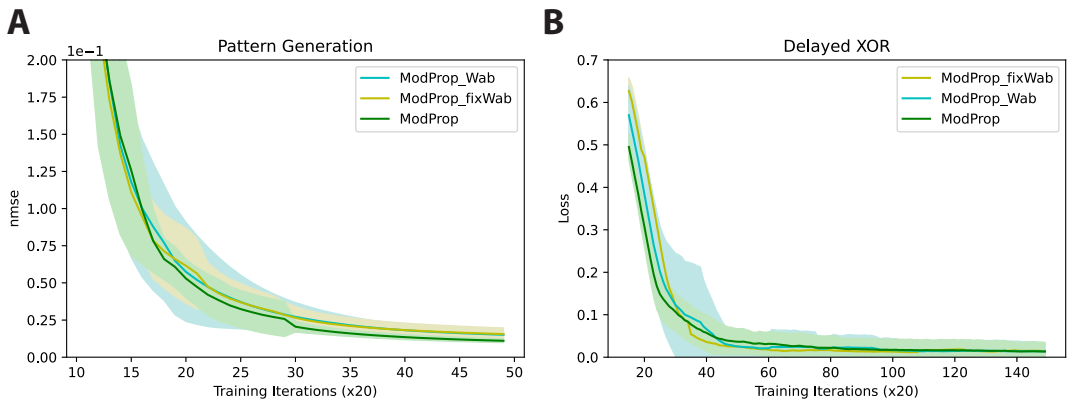

Figure S3.1: Effective learning is achievable with fixed random synapse-type-specific modulatory weights. Figure 3.4 computes type-specific modulatory weights by averaging forward weight entries in the corresponding pre- and postsynaptic cell group. This assumes that modulatory weights co-adapt with synaptic weights. To what extent they are linked in the brain is unclear. Thus, to test the generality of our learning rule, we re-train using fixed random type-specific modulatory weights and show that leads to negligible performance degradation. Note, sequential MNIST task is not considered in figures that involve synapse-type-specific modulatory weights, as cell types were not considered in that task. Plotting convention follows that of previous figures.

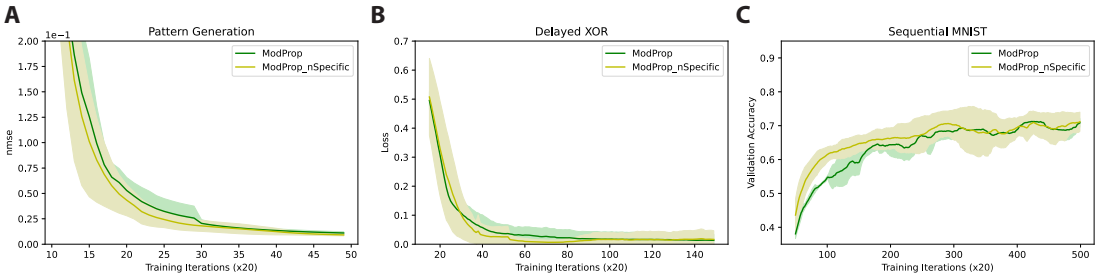

Figure S3.2: Restoring neuron specificity in the activation derivative does not lead to significant improvements. Here, ModProp_global is the basic form of ModProp investigated in the main text, where the activation derivative exhibited no spatiotemporal specificity. ModProp_nSpecific (Eq. S3.26) takes into account the neuron specificity of the activation derivative and only averages across time steps. This comparison is done for the A) pattern generation task, B) delayed XOR task and C) sequential MNIST task. Plotting convention follows that of previous figures.

We discussed how the basic form of ModProp completely neglects any spatiotemporal specificity in the activation derivative. We ask how much performance gain could we get if we

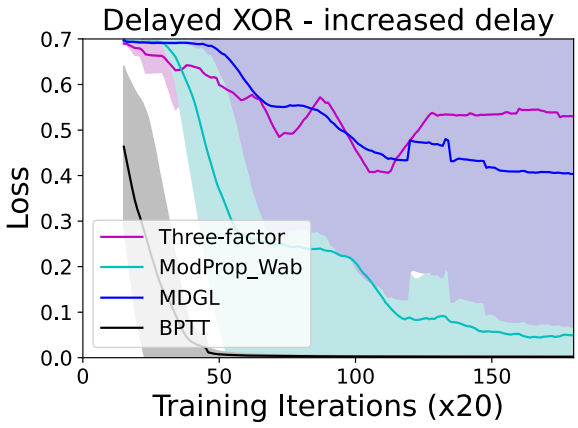

Figure S3.3: Delayed XOR task with a longer delay period. We simulate the delayed XOR task with 1.5 times the delay period used in Figure 3.3B. Although ModProp (with cell-type approximation) still outperforms other bio-plausible learning rules, the performance degrades (compared to ModProp_Wab in Figure 3.4B). Moreover, we found all rules (including BPTT) struggle to learn if we increased the delay period to twice of that in Figure 3.3 without changing other task or network parameters (e.g. cue width and intensity). This connects nicely to our discussion point on the limitation of ModProp in addressing very long temporal credit assignment problems in the absence of a long-term memory mechanism. Solid lines/shaded regions: mean/standard deviation of loss curves across five runs.

lose only temporal specificity, i.e. only average activation derivative across time points. This would see different neurons as having different average activity. To put this more concretely, we approximate the corresponding factor in Eq. S3.12 as

$$\sum_{i_1,\ldots,i_{s-1}} W_{ji_1} h_{i_1,t-1} W_{i_1 i_2} h_{i_2,t-2} \ldots W_{i_{s-1}p} h_{i_{s-1},t-s+1}$$
$$\approx W_{ji_1} \overline{h_{i_1}} W_{i_1 i_2} \overline{h_{i_2}} \ldots W_{i_{s-1}p} \overline{h_{i_{s-1}}} = (\overline{W})^s_{jp}, \tag{S3.26}$$

where $\overline{W} := W \odot \overline{h}$ with $\overline{h}$ — a $1 - by - N$ vector with each entry corresponding to a neuron-specific mean activation — broadcasted for the element-wise multiplication with W. In other words, this restores spatial specificity and the only approximation being made here is to remove temporal specificity of activation derivative. As a practical note, by the famous AM-GM inequality, the estimation ($\Pi_s h_s \approx \overline{h}$) would yield an upper bound of the actual. Thus, we multiply a dampening factor μ to every $\overline{W}$ for stability, and treat μ as a hyperparameter. We name this variant of ModProp as "ModProp_nSpecific", and the most basic form we investigated in the main text as "ModProp_global". Figure S3.2 shows that this restoration of neuron-specificity in activation derivative did not lead to significant

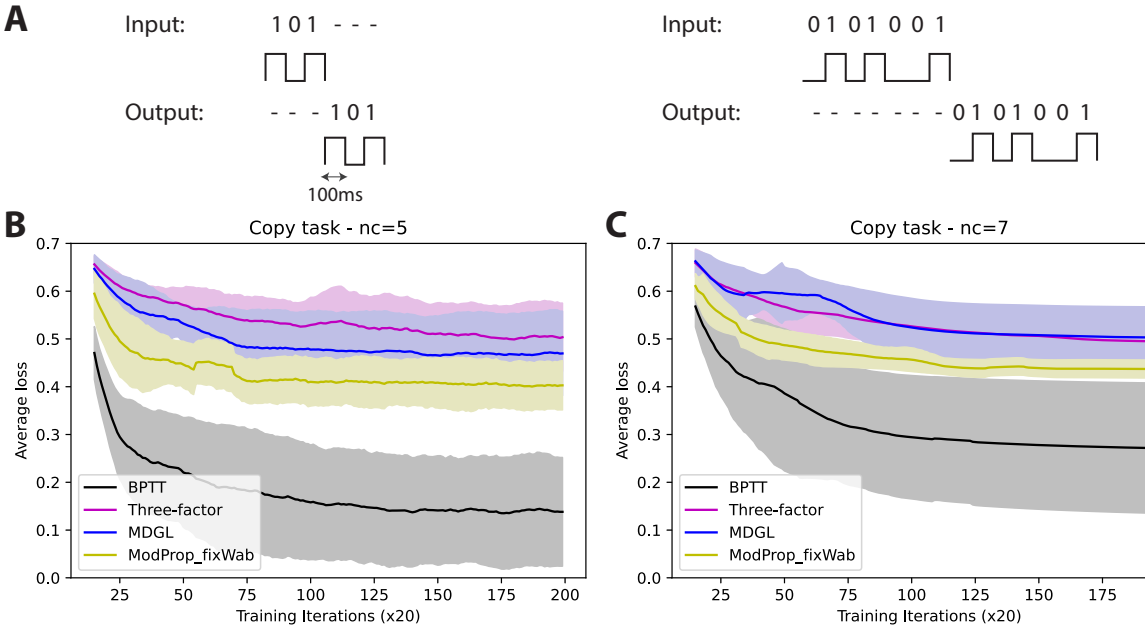

Figure S3.4: Copy task with fixed random synapse-type-specific modulatory weights. Sequences of binary cues are presented to an RNN. For each sequence, once the full sequence has been presented, the network should output the original sequence (with the same value and duration) without any further information [13]. **A)** Examples of input/output pairs at different sequence lengths. Instead of having each cue lasting just 1 step, we have each cue lasting 100 steps (100ms) to mimic the duration of a quick cue flash in biological settings. Superior performance of ModProp even with fixed and random modulatory weights (compared to other biologically plausible rules) is demonstrated for the copy task with a sequence length of **B)** five cues ($nc = 5$) and **C)** seven cues ($nc = 7$). Average loss denotes the binary cross entropy loss computed on target and actual output averaged across time steps. Solid lines/shaded regions: mean/standard deviation of loss curves across five runs.

performance improvement.

S3.3 Further discussion on related algorithms

For efficient online learning in RNNs, approximations to RTRL have been proposed [9, 71–76]. For biological realism (use only local information for local updates) and reduced computational cost, **e-prop** [4] and **RFLO** [9] proposes severe truncation such that the weight update would only depend on pre- and postsynptic neuron activity as well as a top-down error signal that only tells how a neuron directly contributes to the overall network outcome. **MDGL** [11, 131] proposes a less severe truncation than e-prop and RFLO, but it only addresses the contributions to the task error of neurons that are at most 2 synapses away (note, Ref [11] can be considered as a special case of our work, where the filter length is constrained to a single tap). ModProp shows a way of removing this significant limitation and enables the communication and calculation of credit from neurons that can be arbitrarily many synapses away. It also experimentally demonstrates the benefit of this key contribution.

Along the approach of truncations, reference [75] proposed the **SnAp-n** algorithm that allows the user to customize the amount of truncation by deciding on n. SnAP-n stores $\frac{\mathrm{d}\, s_{j,t}}{\mathrm{d}\, w_{pq}}$ (seen in Eq. 2.17) only for j such that parameter w_{pq} influences the activity of unit j within n time steps. SnAp-1 is closely related to e-prop/RFLO (assuming no autapses). However, starting at $n = 2$ (SnAp-2), $\frac{\mathrm{d}\, s_{j,t}}{\mathrm{d}\, w_{pq}}$ will be stored for every $\{j, p, q\}$ such that w_{pq} influences j in two steps. This would require the storage of kN^3 traces, where k is a constant that equals the connection density squared. To our knowledge, there is no evidence on how neural circuits can accommodate such $O(kN^3)$ storage. Therefore, SnAp-n ($n \geq 2$) still poses a significant biological plausibility issue while SnAp-1 reduces to e-prop/RFLO in certain circumstances.

Moving from temporal truncation, **KeRNL** [74] approximates long term dependencies by assuming the dependency to be first order low-pass and learn the parameters using node perturbation. However, the algorithm poses significant implementation and biologically plausibility issues: (1) it uses node perturbation to find the meta-parameters (e.g. the first order decay constant), which is not scalable; (2) meta-parameters are updated per step on the same timescale as synaptic weight update. Their idea of approximating long term dependencies by assuming it follows a certain structure rather than truncating it is what

partially inspired our rule. Unlike KeRNL, our approximation can lead to successful learning with fixed meta parameters that are likely updated on the evolutionary timescale in biology (Figure S3.1).

S3.4 Cost analysis and biological implementation

S3.4.1 Cost and interpretation for biologically-plausible implementation of ModProp in Eq. 3.6

Recall for ModProp, the eligibility trace is combined with total modulatory signals detected:

$$\begin{aligned}
\Delta W_{pq} &\propto \mathrm{ET}_{pq} \times \mathrm{TD}_p + \mathrm{ET}_{pq} * \sum_{\alpha \in C} \mathrm{LM}_{\alpha\beta} \\
\mathrm{ET}_{pq} * \sum_{\alpha \in C} \mathrm{LM}_{\alpha\beta} &= \sum_{s=1}^{S} \mathrm{ET}_{pq,t-s} \times \sum_{\alpha \in C} \mathrm{LM}_{\alpha\beta,s} \\
\mathrm{LM}_{\alpha\beta,s} &= (\text{affinity } W^s_{\alpha\beta}) \times \sum_{j \in \alpha} \underbrace{\mathrm{TD}_j \times (\text{activity } j)}_{\text{modulatory signal } j}.
\end{aligned} \tag{S3.27}$$

We see that the eligibility trace (ET) is brought outside of the double summation of local modulatory (LM) signals. A biological interpretation is that secretion of top-down (TD) learning signals can selectively activate a biochemical process at the post-synaptic neuron, which can then act as a temporal filter on the eligibility trace. The number of filter taps for the underlying biochemical process, S, determines the number of steps for credit information propagation.

Here is the computational cost breakdown for Remark 3.3.2:

- $a_{j,t} = \mathrm{TD}_j \times (\text{activity } j)$ for all $j = 1, .., N$ has $O(N)$ operations.
- $\sum_{j \in \alpha} a_{j,t}$ for $j = 1, ..., N_\alpha$ has $O(N_\alpha)$ operations (assuming $a_{j,t}$ already available, from the last step), where N_α is the number of cells in type α.
- $LM_{\beta,s} := \sum_{\alpha \in C} W^s_{\alpha\beta}(\sum_{j \in \alpha} a_{j,t})$ for all $\alpha, \beta = 1, ...C$ and $s = 1, ..., S$ has $O(SC^2)$ operations. Note, this step is the modulatory communication step, where type-specific-approximation of weights can reduce the cost.

- $\sum_{s=1}^{S} \mathrm{ET}_{pq,t-s} \times \mathrm{LM}_{\beta,s}$ for all $p, q = 1, ..., N$ has N^2 element-wise multiplications per $s = 1, ..., S$, leading to a total of $O(SN^2)$. Since β can be determined from p, there is no need to loop over β in this step.

Since the cost of the last item dominates, **the computational cost scales as** $O(SN^2)$. For storage cost, ModProp stores $e_{pq,t-S}, \dots e_{pq,t}$ for every (pq), leading to **a storage cost of** $O(SN^2)$.

S3.4.2 Cost for biologically-implausible implementation of ModProp

We prove Proposition 1 next, where we discussed a potentially biologically-implausible in silico implementation with lower computational and storage costs than the biologically-plausible version above.

Proof. Let us first introduce the following notations:

- N_α denotes the number of cells in type α
- $[(W^s)_{\alpha\beta}] \in \mathbb{R}^{N_\alpha \times N_\beta}$ is a matrix repeating the value of scalar $(W^s)_{\alpha\beta}$
- Thus, $[(W^1)_{\alpha\gamma}][(W^s)_{\gamma\beta}] = [N_\gamma (W^1)_{\alpha\gamma}(W^s)_{\gamma\beta}]$
- $G^t_{\alpha,pq} := \sum_{s=1}^{t} (W^s)_{\alpha\beta}\, \mu^{s-1}\, e_{pq,t-s}$

By properties of block matrix product:

$$[(W^{s+1})_{\alpha\beta}] = \sum_\gamma [(W^1)_{\alpha\gamma}][(W^s)_{\gamma\beta}] = [\sum_\gamma N_\gamma (W^s)_{\alpha\gamma}(W^s)_{\gamma\beta}]$$
$$\rightarrow (W^{s+1})_{\gamma\beta} = \sum_\gamma N_\gamma (W^1)_{\alpha\gamma}(W^s)_{\gamma\beta}. \tag{S3.28}$$

Now, let's find a recursive expression to calculate $Z^{t+1}_{\alpha,pq}$ online:

$$\begin{aligned}
G^{t+1}_{\alpha,pq} &= \sum_{s=1}^{t+1} (W^s)_{\alpha\beta}\, \mu^{s-1}\, e_{pq,t+1-s} \\
&= \sum_{s=0}^{t} (W^{s+1})_{\alpha\beta}\, \mu^{s}\, e_{pq,t-s} \\
&= \sum_{s=1}^{t} (W^{s+1})_{\alpha\beta}\, \mu^{s}\, e_{pq,t-s} + (W^1)_{\alpha\beta} e_{pq,t} \\
&= \sum_{s=1}^{t} (\sum_{\gamma} N_\gamma (W^1)_{\alpha\gamma} (W^s)_{\gamma\beta})\, \mu^{s}\, e_{pq,t-s} + (W^1)_{\alpha\beta} e_{pq,t} \\
&= \sum_{\gamma} N_\gamma (W^1)_{\alpha\gamma} \mu \sum_{s=1}^{t} (W^s)_{\gamma\beta}\, \mu^{s-1}\, e_{pq,t-s} + (W^1)_{\alpha\beta} e_{pq,t} \\
&= \sum_{\gamma} N_\gamma (W^1)_{\alpha\gamma} \mu G^{t}_{\gamma,pq} + (W^1)_{\alpha\beta} e_{pq,t}
\end{aligned} \tag{S3.29}$$

And the overall update is:

$$\Delta W_{pq}|_{ModProp} = \frac{\partial E}{\partial z_{p,t}} e_{pq,t} + \sum_{\alpha} G^{t}_{\alpha,pq} \sum_{j\in\alpha} \frac{\partial E}{\partial z_{j,t}} h_{j,t} \tag{S3.30}$$

The second term dominates the cost, for which we need to store $G^t_{\alpha,pq}$ for every α, p, q. This amounts to $O(CN^2)$ storage cost. To update and attain $G^{t+1}_{\alpha,pq}$, we need $O(C)$ summations and multplications per $G^{t+1}_{\alpha,pq}$, which amounts to $O(C^2N^2)$ computational cost. The final step of combining G and $\sum_{j\in\alpha} \frac{\partial E}{\partial z_{j,t}} h_{j,t}$, requires $O(CN^2)$ computational cost, which does not dominate the cost.

We note that the specific implementation outlined in the proof of Proposition 1 can significantly reduce the implementation cost, but is likely biologically-implausible, because each synaptic weight update requires the knowledge of all modulatory weights in the network (Appendix S3.4). Moreover, it reduces the cost compared to RTRL ($O(N^3)$ storage and $O(N^4)$ computational complexity) as well as SnAP-2 ($O(d^2N^3)$ storage and $O(d^2N^4)$ computational complexity for connection density d) [75] significantly if only a few cell types are used. In this work, we used only two cell types ($C = 2$) that map onto the two main cell classes: excitatory and inhibitory. However, ModProp is more expensive (by a constant factor) than e-prop, RFLO and MDGL, which all have $O(N^2)$ storage and $O(N^2)$ computational

complexity. However, as mentioned, the performance of these rules are limited due to their severe temporal truncation.

□

S3.5 Unreasonable effectiveness of synapse-type-specific modulatory backpropagation (through time) weights

We provide the proof for Theorem 1 below:

Proof. We first show that $\mathbb{E}[(\epsilon_s)_{ij}] = 0$ for all $s \geq 1$. Note $(W^s)_{\alpha\beta} = \sum_\gamma N_\gamma (W^{s-1})_{\alpha\gamma} W_{\gamma\beta}$ and $(W^s)_{ij} = \sum_k (W^{s-1})_{ik} W_{kj}$ can be calculated recursively.

The base case $s = 1$ is already given in the condition. Suppose $\mathbb{E}[(\epsilon_s)_{ij}] = 0$, for $s + 1$:

$$\begin{aligned}
\mathbb{E}[(\epsilon_{s+1})_{ij}] &= \mathbb{E}[(W^{s+1})_{ij} - (W^{s+1})_{\alpha\beta}] \\
&= \mathbb{E}\left[\sum_k (W^s)_{ik} W_{kj}\right] - \sum_\gamma N_\gamma (W^s)_{\alpha\gamma} W_{\gamma\beta} \\
&= \mathbb{E}\left[\sum_\gamma \sum_{k\in\gamma} ((W^s)_{\alpha\gamma} + (\epsilon_s)_{ik})(W_{\gamma\beta} + \epsilon_{kj})\right] - \sum_\gamma N_\gamma (W^s)_{\alpha\gamma} W_{\gamma\beta} \\
&= \sum_\gamma N_\gamma (W^s)_{\alpha\gamma} W_{\gamma\beta} + \sum_\gamma \sum_{k\in\gamma} \mathbb{E}[(\epsilon_s)_{ik}] W_{\gamma\beta} + \sum_\gamma \sum_{k\in\gamma} (W^s)_{\alpha\gamma} \mathbb{E}[\epsilon_{kj}] \\
&\quad + \sum_\gamma \sum_{k\in\gamma} \mathbb{E}[(\epsilon_s)_{ik}\epsilon_{kj}] - \sum_\gamma N_\gamma (W^s)_{\alpha\gamma} W_{\gamma\beta} \\
&= \sum_\gamma \sum_{k\in\gamma} \mathbb{E}[(\epsilon_s)_{ik}] W_{\gamma\beta} + \sum_\gamma \sum_{k\in\gamma} (W^s)_{\alpha\gamma} \mathbb{E}[\epsilon_{kj}] + \sum_\gamma \sum_{k\in\gamma} \mathbb{E}[(\epsilon_s)_{ik}] \mathbb{E}[\epsilon_{kj}] \\
&= 0. \qquad \text{(S3.31)}
\end{aligned}$$

We now prove the Theorem statement for the scalar output case. The extension to multiple output signals follows identically. Consider the loss decrement after one update,

under the (locally) first order loss assumption:

$$\begin{aligned}
\mathbb{E}\left[\Delta E|_{pq,t}\right] &= -\eta\mathbb{E}\left[\widehat{\frac{dE}{dW_{pq}}}\frac{dE}{dW_{pq}}\right] \\
&= -\eta\mathbb{E}\Bigg[(y_t - y_t^*)^2\Bigg[W_p^{\text{OUT}}e_{pq,t} + \sum_{s,\alpha}\sum_{j\in\alpha}W_j^{\text{OUT}}(W^s)_{\alpha\beta}e_{pq,t-s}\Bigg] \\
&\quad\times\Bigg[W_p^{\text{OUT}}e_{pq,t} + \sum_{u,\alpha'}\sum_{j'\in\alpha'}W_{j'}^{\text{OUT}}[(W^u)_{\alpha'\beta} + (\epsilon_u)_{j'p}]e_{pq,t-u}\Bigg]\Bigg] \\
&= -\mathbb{E}[\Gamma_{pq}^2] - \eta\mathbb{E}\Bigg[(y_t - y_t^*)^2\Bigg[W_p^{\text{OUT}}e_{pq,t} + \sum_{s,\alpha}\sum_{j\in\alpha}W_j^{\text{OUT}}(W^s)_{\alpha\beta}e_{pq,t-s}\Bigg] \\
&\quad\times\Bigg[\sum_{u,\alpha'}\sum_{j'\in\alpha'}W_{j'}^{\text{OUT}}(\epsilon_u)_{j'p}e_{pq,t-u}\Bigg]\Bigg] \\
&= -\mathbb{E}[\Gamma_{pq}^2] - \eta\sum_{u,\alpha'}\sum_{j'\in\alpha'}W_{j'}^{\text{OUT}}W_p^{\text{OUT}}\mathbb{E}\left[(\epsilon_u)_{j'p}(y_t - y_t^*)^2e_{pq,t}e_{pq,t-u}\right] \\
&\quad - \eta\sum_{s,u,\alpha,\alpha'}\sum_{j\in\alpha,j'\in\alpha'}(W^s)_{\alpha\beta}W_j^{\text{OUT}}W_{j'}^{\text{OUT}}\mathbb{E}\left[(\epsilon_u)_{j'p}(y_t - y_t^*)^2e_{pq,t-s}e_{pq,t-u}\right] \\
&\overset{(a)}{=} -\mathbb{E}[\Gamma_{pq}^2] - \eta\sum_{u,\alpha'}\sum_{j'\in\alpha'}W_{j'}^{\text{OUT}}W_p^{\text{OUT}}\mathbb{E}\left[(\epsilon_u)_{j'p}\right]\mathbb{E}\left[(y_t - y_t^*)^2e_{pq,t}e_{pq,t-u}\right] \\
&\quad - \eta\sum_{s,u,\alpha,\alpha'}\sum_{j\in\alpha,j'\in\alpha'}(W^s)_{\alpha\beta}W_j^{\text{OUT}}W_{j'}^{\text{OUT}}\mathbb{E}\left[(\epsilon_u)_{j'p}\right]\mathbb{E}\left[(y_t - y_t^*)^2e_{pq,t-s}e_{pq,t-u}\right] \\
&\overset{(b)}{=} -\mathbb{E}[\Gamma_{pq}^2] \leq 0,
\end{aligned} \tag{S3.32}$$

where (a) follows from the uncorrelatedness condition and (b) follows from the result of (S3.31). Here, we defined $\Gamma_{pq} := \eta(y_t - y_t^*)\left[W_p^{\text{OUT}}e_{pq,t} + \sum_s\sum_\alpha\sum_{j\in\alpha}W_j^{\text{OUT}}(W^s)_{\alpha\beta}e_{pq,t-s}\right]$. Then,

$$\mathbb{E}[\Delta E|_t] = -\eta\mathbb{E}\left[\widehat{\nabla E}^T\nabla E\right] = -\eta\sum_{p,q}\mathbb{E}\left[\widehat{\frac{dE}{dW_{pq}}}\frac{dE}{dW_{pq}}\right] \leq 0. \tag{S3.33}$$

Moreover, if gradient descent is possible for a network $\hat{\mathcal{N}}$ with weight $W_{ij} = W_{\alpha\beta}$, $\forall i \in \alpha, j \in \beta$, then $\mathbb{E}[\sum_{p,q}\Gamma_{pq}] < 0$ by definition and $\mathbb{E}[\Delta E|_t] < 0$.

□

We note that with the linear RNN assumption in Theorem 1, **Approximation 1** becomes exact when $\mu = 1$ because the activation derivative is a constant 1 for linear networks. Thus, the proof only only examines the effect of **Approximation 2** (type-specific feedback weight

approximation). Also, the Theorem assumes uncorrelatedness for residual weights ϵ, which may not be the case for networks that are not Erdős–Rényi [132]. Despite that, ModProp still leads to performance improvement over existing rules for the tasks examined. Nevertheless, it is important to investigate ModProp across a broad range of tasks in the future.

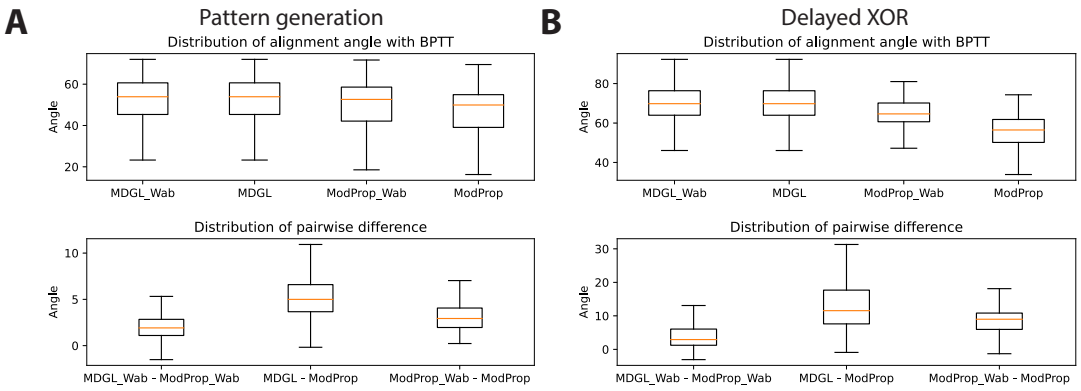

Figure S3.5: Alignment angle comparison shows that gradients approximated by ModProp (with or without type-specific modulatory weights) are more similar (than MDGL) to the exact gradients. We quantify the similarity between approximated and exact gradients via the alignment angle, which describes the similarity in the direction of the two update vectors (Appendix S3.6) for various tasks. In all top-panels, the alignment angles between approximate rules and BPTT are less than 90°, which indicate that the approximated gradients are aligned with the exact gradient, despite the high-dimensionality of the update vectors. All bottom panel plots show that ModProp variants achieve smaller alignment angles (hence better alignment) with BPTT than MDGL does. To ensure a fair comparison, we examine the statistics of pairwise differences, so that the point on the loss landscape — where the comparison is done — is matched. This is achieved by training the network using BPTT across seven different runs and sampling the approximated gradient once every 50 training iterations. Alignment analysis illustrated here is for recurrent weight gradients, and similar trends are observed for the input weights as well.

S3.6 Simulation details

All weight updates were implemented using Adam with default parameters [107]. All Adam learning rates are optimized by picking the best one within $\{5e-5, 1e-4, 2e-4, 5e-4, 1e-3, 2e-3, 5e-3\}$ for every learning rule and task. For ModProp, the best value of hyperparameter μ (Eq. 3.3) was picked within $\{0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5\}$. For every learning rule and task, we removed the worst performing run quantified by area under the learning curve. We note that while input, recurrent and output weights are all being trained, the nonlocality issue (Eq. 2.17) only applies to training input and recurrent weights. Thus, all approaches update output weights using backpropagation, and approximations apply to training input and recurrent weights. As stated, we repeated runs with different random initialization to quantify uncertainty and weights were initialized similarly as in [21].

We used alignment angle to quantify the similarity between two vectors. The alignment angle θ between two vectors, a and b, was computed by $\theta = acos(\|a^T b\|/\|a\|\|b\|)$. The alignment between two 2D matrices was computed by flattening the matrices into vectors.

For the pattern generation task, our network consisted of 400 neurons described in Eq. S3.1. All neurons had a membrane time constant of $\tau_m = 30$ms. Input to this network was provided by 50 units each producing a different random Gaussian input. The fixed target signal had a duration of 2000ms and given by the sum of five sinusoids, with fixed frequencies of 0.5Hz, 1Hz, 2Hz, 3Hz and 4Hz. For learning, we used mean squared loss function and for visualization, we used normalized mean squared error NMSE $= \frac{\sum_{k,t}(y^*_{k,t} - y_{k,t})^2}{\sum_{k,t}(y^*_{k,t})^2}$ for zero-mean target output $y^*_{k,t}$. For the delayed XOR task, our implementation of the task involved the presentation of two sequential cues, each lasting 100ms and separated by a 700ms delay. There was only one input unit involved and two cue alternatives were presented by setting the input unit to 1 or 0, and the unit was set to to 0 during the delay period. In addition, a Gaussian noise with $\sigma = 0.01$ was added to the input. The network was trained to output 1 (resp. 0) at the last time step when the two cues have matching (resp. non-matching) values. Our network consisted of 120 neurons. All neurons had a membrane time constant of $\tau_m = 100$ms. For learning, we used cross-entropy loss function and the target corresponding to the correct output was given at the end of the trial. A batch size of 32 was used and the

gradients were accumulated during those trials additively.

For the copy task, we presented a input sequence of seven binary cues (taking on the value of 0 or 1) on one set of runs and five cues on another. Each cue lasts 100ms to mimic duration of a quick cue flash in biological setting. After the full sequence presentation, the network is tasked to output the same sequence (same value and duration) without further instruction. Our network consisted of 120 neurons. All neurons had a membrane time constant of $\tau_m = 100$ms. For learning, we used cross-entropy loss function and the target corresponding to the correct output was given at the end of the trial. We used full batch training: a batch size of 8 (resp. 128) was used for the three (resp. seven) cue sequence runs due to 8 (resp. 128) possible permutations.

For the pixel-by-pixel MNIST task [10], our network consisted of 200 neurons. All neurons had a membrane time constant of $\tau_m = 20$ms. Input to this network was provided by a single unit that represented the grey-scaled value of a single pixel, with a total of 784 steps and the network prediction was made at the last step. For learning, we used the cross-entropy loss function and the target corresponding to the correct output was given at the end of the trial. A batch size of 256 was used and the gradients were accumulated during those trials additively.

We used TensorFlow [133] version 1.14 and based it on top of [21]. [1] We performed simulations on a computer server with x2 6-core Intel Xeon E5-2640, 2.5GHz, 32 GB RAM. Regardless of the learning rule, our implementation takes approximately one hour to complete one run of pattern generation or delayed XOR task training (for Figures 3.3 and 3.4) on the server.

[1] Our code is available: `https://github.com/Helena-Yuhan-Liu/ModProp`.

Chapter 4

BEYOND ACCURACY: GENERALIZATION PROPERTIES OF BIO-PLAUSIBLE TEMPORAL CREDIT ASSIGNMENT RULES

4.1 Introduction

A longstanding question in neuroscience is how animals excel at learning complex behavior involving temporal dependencies across multiple timescales and thereafter generalize this learned behavior to unseen data. This requires solving the temporal credit assignment problem: how to assign the contribution of past neural states to future outcomes. To address this, neuroscientists are increasingly turning to the mathematical framework provided by recurrent neural networks (RNNs) to model learning mechanisms in the brain [126, 134]. Temporal credit assignment in RNNs is typically achieved by backpropagation through time (BPTT), or other gradient descent-based optimization algorithms, none of which are biologically-plausible (or bio-plausible for short). Therefore, the use of RNNs as a framework to understand the computational principles of learning in the brain has motivated an influx of bio-plausible learning rules that approximate gradient descent [71, 126, 134].

The performance of such rules is typically quantified by accuracy. Although the accuracy achieved by these rules is often comparable to true gradient descent, little is known about the breadth of the emergent solutions, namely how well they generalize. Broadly speaking, generalization refers to a trained model's ability to adapt to previously unseen data, and is typically measured by the so-called **generalization gap**: the difference between training and testing error. This is especially important when learning complex tasks with nonlinear RNNs where the loss landscape is non-convex, and therefore, many solutions with comparable training accuracy can exist. These solutions, characterized as (local) minima in the loss landscape, can nonetheless exhibit drastically different levels of generalization (Figure 4.1). It is not clear if gradient-based methods like BPTT and the existing bio-plausible alternatives have a different tendency to converge to loss minima that provide better or worse generalization.

While the search for better predictors of the generalization performance remains an open issue in deep learning research [135], recent extensive studies identify flatness of the loss landscape at the solution point as a promising predictor for generalization [14, 136–139]. Leveraging these empirical and theoretical findings, **we ask**: how do proposed biologically-motivated gradient approximations affect the flatness of the converged solution on loss landscape, and thereby, generalization?

Our overarching goal is to investigate generalization trends for existing bio-plausible temporal credit assignment rules, and in particular, examine how truncation-based bio-plausible gradient approximations can affect such trends. Specifically, **our contributions** are summarized as follows:

- In numerical experiments, we demonstrate across several well-known neuroscience and machine learning benchmark tasks that state-of-the-art (SoTA) bio-plausible learning rules for training RNNs exhibit worse and more variable generalization gaps, compared to true (stochastic) gradient descent (Figure 4.2A-C).

- Using the same experiments, we show that bio-plausible learning rules tend to approach high-curvature regions in synaptic weight space as measured by the loss' Hessian eigenspectrum (Figure 4.2D-F). Further, we verify that this correlates with worse generalization (Figure 4.2D-F and 4.3), which is consistent with the literature.

- We present a theorem to explain this phenomenon by examining the weight update equation as a discrete dynamical system, which sheds light on a potential connection between gradient alignment and the preference over converging to narrower minima (Theorem 2, Figure 4.4).

Given the core components in designing artificial neural networks: data, objective functions, learning rules and architectures [61], we investigate different learning rules while holding the data, objective function and architecture constant. SoTA RNN learning rules investigated include a three-factor rule with symmetric feedback (symmetric e-prop [4]), a three-factor rule using random feedback weights (RFLO [140]) and a multi-factor rule using local modulatory

signaling (MDGL) [11]. For an in-depth explanation of how these rules are implemented and why they are bio-plausible, please refer to Appendix S4.1.2. We also encourage the reader to visit Appendix B for Theorem 2 proof and discussion on loss landscape geometry. In the last paragraph of the Discussion section, we discuss potential remedies implemented by the brain and provide preliminary results (Appendix Figure S4.1). To our knowledge, our analysis is the first to highlight and quantitatively provide a mechanistic explanation of the reason for this gap in solution quality between artificial and bio-plausible learning rules for RNNs, thereby motivating further investigations into how the brain learns solutions that generalize.

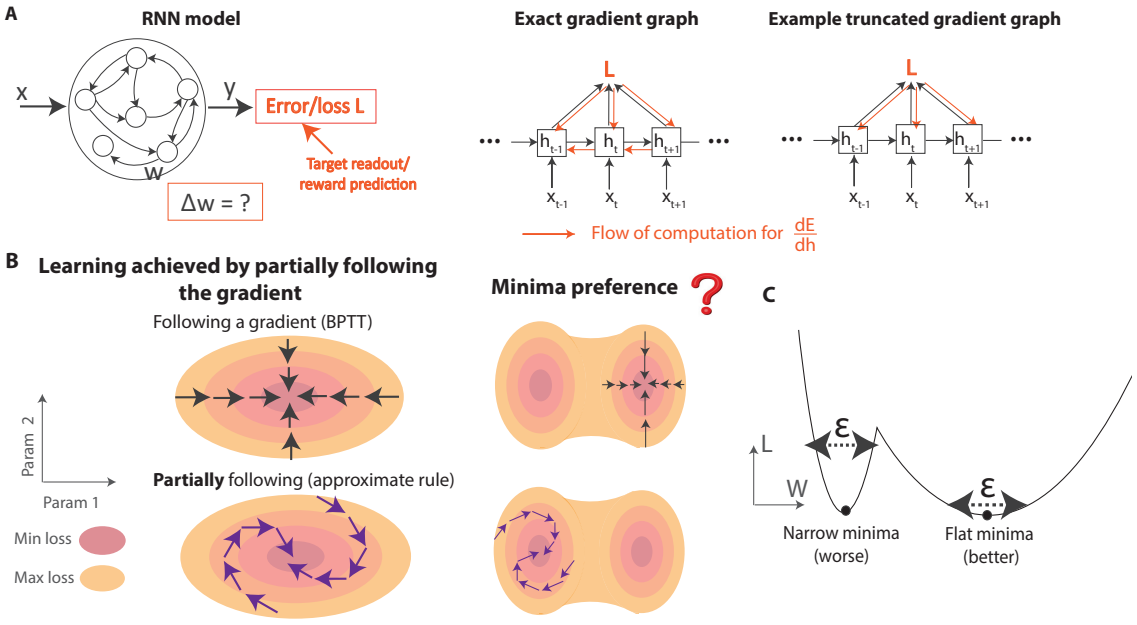

Figure 4.1: Setup. A) Illustration of an RNN trained to minimize error/loss function L (left). Existing bio-plausible proposals for RNNs estimate the gradient by neglecting dependencies that are biologically implausible to compute (right). B) Low training error/loss can be achieved by **partially** following a gradient (right), but the preference for converging to minima with certain generalization properties remains underexamined for these learning rules (left). C) Minima flatness matters: 1-D loss landscape illustration with two solutions that equally minimize loss L, but exhibit drastically different generalization properties: the narrower minima are more sensitive to perturbation.

4.2 Related works

4.2.1 Bio-plausible gradient approximations

Investigating bio-plausible learning rules is of interest both to identify more efficient training strategies for artificial networks and to better understand learning in the brain [71, 134, 141–170]. In order for learning to reach a certain goal quantified by an objective, learning algorithms often minimize a loss function [61]. The error gradient, if available, tells us how each parameter should be adjusted in order to have the steepest local descent. For training RNNs, which are widely used as a model for neural circuits [30, 171–189], standard algorithms that follow this gradient — real time recurrent learning (RTRL) and BPTT — are not bio-plausible and have overwhelming memory storage demands [70, 71]. However, learning rules that only approximate the true gradient can sometimes be as effective as those that follow the gradient exactly [61, 97]. Because of that, bio-plausible learning rules that approximate the gradient using known properties of real neurons have been proposed and led to successful learning outcomes across many different tasks in feedforward networks [22, 56, 59, 79–84, 126, 190], with recent extensions to recurrently connected networks [4, 11, 140]. These existing bio-plausible rules for training RNNs [4, 11, 140] are truncation-based (which is the focus of this study), so that the untruncated gradient terms can be assigned with putative identities to known biological learning ingredients: eligibility traces, which maintain preceding activity on the molecular levels [8, 38, 42, 90, 108, 109], combined with top-down instructive signaling [8, 40, 42, 44, 86, 110, 119, 120, 122] as well as local cell-to-cell modulatory signaling within the network [11, 191, 192]. For efficient online learning in RNNs, other approximations (not necessarily bio-plausible) to RTRL [13, 75, 76, 193–195] have also demonstrated to produce a good performance. Given the impressive accuracy achieved by these approximate rules, several studies began to investigate their convergence properties [196], e.g. for random backpropagation weights in feedforward networks [197, 198]. However, the trend of generalization capabilities of these rules, especially in RNNs, is underexamined.

4.2.2 Loss landscape curvature and generalization performance

Given the central importance of understanding how neural networks perform in situations unseen during training [18, 135, 199–204], the deep learning community has made tremendous efforts to develop tools for understanding generalization that we leverage here. That flat minima could lead to better generalization was observed more than two decades ago [205]. Intuitively, under the same amount of perturbation ϵ in parameter space (e.g. loss landscape changes due to the addition of new data) worse performance degradation will be seen around the narrower minima (see Figure 4.1C). We note that perturbations in parameter space can be linked to that in the input space [14]. Recently, many empirical studies have consistently supported the usefulness of this predictor [206–213]. In particular, the authors of [136] performed an extensive study and found that flatness-based measures have a higher correlation with generalization than other alternatives. Motivated by this, several studies have characterized properties of the loss functions's Hessian — whose eigenspectrum carries information about curvature [214–218]. Connections between flatness and generalization performance have shed light on the reason for greater generalization gaps in large batch training [209, 219–222], and also have inspired optimization methods to favor flatter minima [206, 207, 223–230]. Despite the criticism of scale-dependence of flatness [231], where parameter rescaling can drastically change flatness but not always generalization quality, flatness — with parameter scales taken into account [232, 233] — are connected to PAC-Bayesian generalization error bounds [137–139]. Moreover, a recent theoretical study rigorously connects the flatness of the loss surface to generalization in classification tasks under the assumption that labels are (approximately) locally constant [14]. Leveraging the great progress from the deep learning community, we aim to study the generalization properties of bio-plausible learning rules from a geometric perspective.

4.3 Results

In this section, we first describe the network and learning setup we use (Figure 4.1A). Next, we present a number of numerical experiments where we compute the generalization gap directly on three commonly used ML and neuroscience tasks (Figure 4.2A-C), for truncation-based

bio-plausible gradient approximations. For the same experiments, we also quantify loss landscape curvature along learning trajectories, and connect these quantities to generalization behavior (Figures 4.2D-F and 4.3). Finally, we provide theoretical arguments and a theorem that explains how gradient alignment in bio-plausible gradient approximations can affect curvature preference (Theorem 2, Figure 4.3) and thus, generalization. Through additional experiments, we verify the predictive power of our theory. We conclude with discussions on potential remedies used by the brain (Appendix Figure S4.1) and future directions.

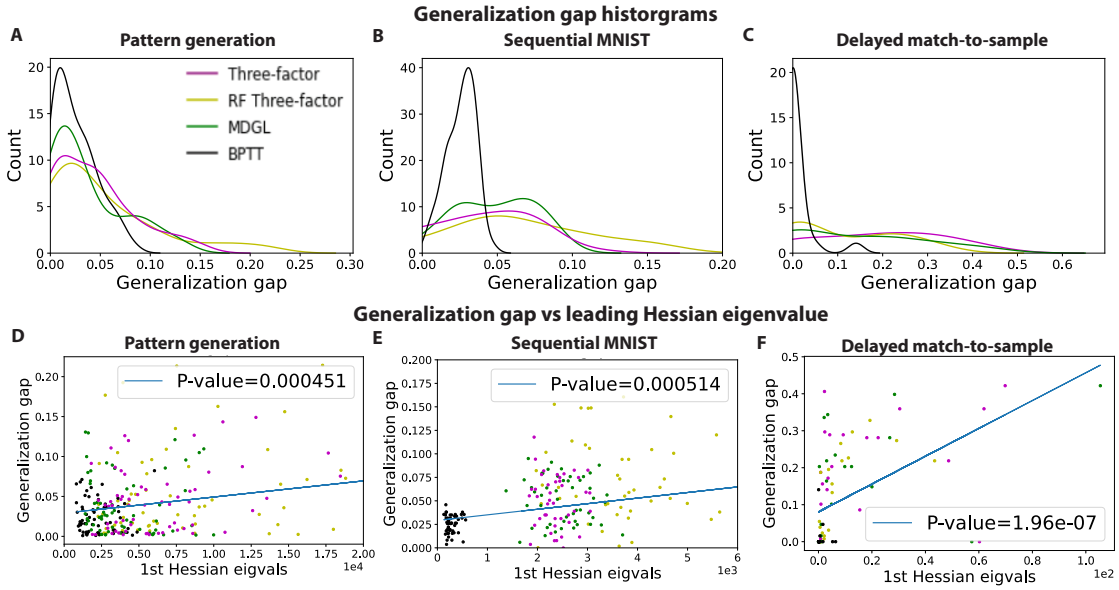

Figure 4.2: Bio-plausible temporal credit assignment rules show worse and more variable generalization gap, which can be informed by loss landscape curvature. A-C) Generalization gap distributions computed at the end of training across different random weight initializations for several well-known neuroscience and machine learning tasks. The higher the generalization gap, the worse the generalization performance. BPTT (black), bio-plausible alternatives (magenta, yellow and green). D-F) Scatter plots showing the trend of generalization gap v.s. leading loss Hessian eigenvalue across many runs; each point corresponds to a single run of the same runs as in A-C.

4.3.1 Network and learning setup

The detailed governing equations of our setup can be found in Methods (Appendix S4.1). We consider a RNN with N_{in} input units, N hidden units and N_{out} readout units (Figure 4.1A). We verified that trends hold for different network sizes and refer the reader to Appendix S4.1.3

for more details. The update formula for $h_t \in \mathbb{R}^N$ (the hidden state at time t) is governed by:

$$h_{t+1} = \phi(W_h f(h_t), W_x x_t), \tag{4.1}$$

where $\phi(\cdot) : \mathbb{R}^N \to \mathbb{R}^N$ is the hidden state update function, $f(\cdot) : \mathbb{R}^N \to \mathbb{R}^N$ is the activation function, $W_h \in \mathbb{R}^{N \times N}$ (resp. $W_x \in \mathbb{R}^{N_{in} \times N}$) is the recurrent (resp. input) weight matrix and $x \in \mathbb{R}^{N_{in}}$ is the input. For ϕ, we consider a discrete-time implementation of a rate-based recurrent neural network (RNN) similar to the form in [124] (details in Appendix S4.1). Readout $\hat{y} \in \mathbb{R}^{N_{out}}$, with readout weights $w \in \mathbb{R}^{N_{out} \times N}$, is defined as

$$\hat{y} = \langle w, f(h_t) \rangle. \tag{4.2}$$

We performed experiments on three tasks: sequential MNIST [10], pattern generation [234] and delayed match-to-sample tasks [235]. The objective is to minimize scalar loss $L \in \mathbb{R}$, which is defined as

$$L(W_h) = \begin{cases} \frac{1}{2TB} \sum_{i=1}^{B} \sum_{t=1}^{T} \sum_{k=1}^{N_{out}} (\hat{y}_{k,t}^{(i)} - y_{k,t}^{(i)})^2, \text{ for regression tasks} \\ \frac{-1}{TB} \sum_{i=1}^{B} \sum_{t=1}^{T} \sum_{k=1}^{N_{out}} \pi_{k,t}^{(i)} log \hat{\pi}_{k,t}^{(i)}, \text{ for classification tasks} \end{cases} \tag{4.3}$$

given target readout $y \in \mathbb{R}^{N_{out}}$, task duration $T \in \mathbb{R}$ and batch size $B \in \mathbb{R}$. $\pi_{k,t} \in \mathbb{R}$ is the one-hot encoded target for readout unit k at time t and $\hat{\pi}_{k,t} = \text{softmax}_k(\hat{y}_{1,t}, \ldots, \hat{y}_{N_{OUT},t}) = \exp(\hat{y}_{k,t}) / \sum_{k'} \exp(\hat{y}_{k',t})$ is the predicted category probability.

Different learning algorithms examined in this work are BPTT (our benchmark), which update weights by computing the exact gradient ($\nabla L(W_h) \in \mathbb{R}^{N \times N}$):

$$\Delta W_h = -\eta \nabla L(W_h), \tag{4.4}$$

and three SoTA bio-plausible learning rules that update weights using approximate gradient:

$$\widehat{\Delta W_h} = -\eta \tilde{\nabla} L(W_h), \tag{4.5}$$

where $\tilde{\nabla} L(W_h) \in \mathbb{R}^{N \times N}$ denotes a gradient approximation and $\eta \in \mathbb{R}$ denotes the learning rate. These three learning rules are explained further in Appendix S4.1 (Methods) but we note that these bio-plausible learning rules are based on truncations of dependency paths — on the computational graph for the exact gradient— that are biologically implausible to

compute (Figure 4.1A). In all figures, learning rules are labeled as "Three-factor" (symmetric e-prop), "RF Three-factor" (RFLO) and "MDGL", respectively. We remark that the focus here is on comparing artificial to bio-plausible learning rules, rather than between biological rules. Finally, we note that tasks were learned with mostly comparable training accuracies for all learning rules, and that generalization gaps reflect a testing departure from these values. We refer the reader to the Appendix for more details (Appendix S4.1.3).

4.3.2 Generalization gap and loss landscape curvature

To study generalization performance, our first step is to compute the generalization gap empirically. Generalization gap is defined as train accuracy minus test accuracy; the larger the generalization gap, the worse the generalization performance. For various learning rules, we plot the generalization gap histogram at the end of training across runs with distinct initializations; different colors represent different learning rules (Figure 4.2A-C). Notice that these bio-plausible rules achieve worse and more variable generalization performance than their machine learning counterpart (BPTT).

We now investigate if the generalization gap behavior described above correlates well with Loss landscape curvature. We use the leading eigenvalue of the loss' Hessian (where derivatives are taken with respect to model parameters) as a measure for curvature following previous practice [219] and note that it is practical for both empirical [216] and theoretical analyses (Theorem 2). There exist other measures for flatness and due to the scale-dependence issue of Hessian spectrum [231], we also test using parameter scale-independent measures (see Appendix Figures S4.3 and S4.4). When we plot generalization gap points from Figure 4.2A-C against the corresponding leading Hessian eigenvalue, a statistically significant correlation is observed (Figure 4.2D-F). We also observed such correlation across runs with the learning rule fixed (Appendix Figure S4.8). We note that we did not expect this relationship to be very tight since in addition to the worse generalization gap on average, bio-plausible learning rules exhibit increased variance. This is an important and consistent trend we observed but might have been overlooked in previous studies.

So far, we investigated the endpoints of optimization trajectories, where training per-

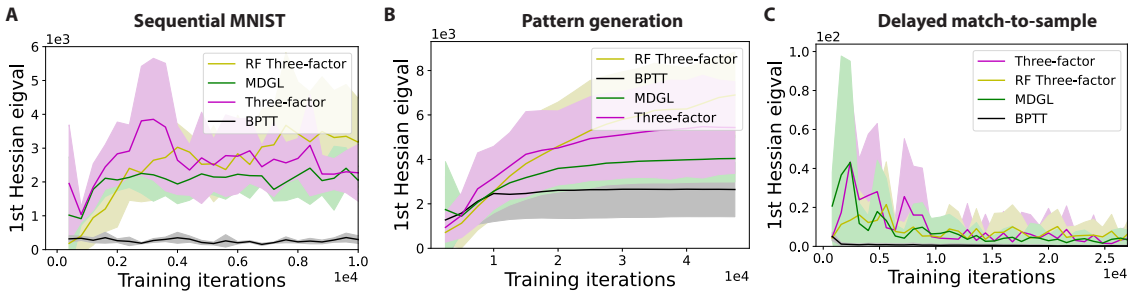

Figure 4.3: Bio-plausible gradient approximations tend to approach high curvature regions in loss landscape. Dominant Hessian eigenvalues are plotted throughout training for bio-plausible learning rules and BPTT. This analysis is done for A) sequential MNIST, B) pattern generation and C) delayed match-to-sample tasks. Solid lines/shaded regions: mean/standard deviation of dominant Hessian eigenvalue curves across five independent runs.

formance has converged. Now, we visualize the whole training trajectory. This done is for two reasons: 1) account for early stopping that can halt training anywhere along the trajectory due to time constraints; 2) flatness during training could give indications of avoiding or escaping high-curvature regions. We observe that the biologically motivated gradient approximations tend to rapidly approach high-curvature regions compared to their machine learning counterpart (Figure 4.3). Together, these results demonstrate a clear trend and a link between the generalization gap and loss landscape curvature: both being increased and more variable for bio-plausible rules. We also stopped BPTT early to match the test accuracy of the three-factor rule, and observed similar trends as in the main text (Appendix Table S4.1). We also note that the curvature convergence behavior seems to be a shared problem of temporal truncations of the gradient (Appendix Figure S4.5), which is what the existing bio-plausible gradient approximations for RNNs are based on. Next, we provide a theoretical argument as to why truncated temporal credit assignment rules favor high curvature regions of the loss landscape.

4.3.3 Theoretical analysis: link between curvature and gradient approximation error

We discussed how generalization can be linked to curvature, and we now examine the link between curvature and gradient alignment during learning dynamics. We represent approximate gradients in a rule-agnostic manner, where an arbitrary approximation is represented in terms of its component along the gradient direction plus an arbitrary orthogonal

vector (Figure 4.4A):

$$\vec{\hat{g}} = \rho\vec{g} + \vec{e}, \tag{4.6}$$

where $\vec{g} := \nabla L(W_h) \in \mathbb{R}^{N^2}$ and $\vec{\hat{g}} \in \mathbb{R}^{N^2}$ are the exact and approximate gradients, respectively (we reshaped $\nabla L(W_h)$ into a vector here); $\vec{e} \in \mathbb{R}^{N^2}$ is an arbitrary orthogonal vector to $\vec{g}$. Here, scalar $\rho \in \mathbb{R}$ represents the relative step length along the gradient direction that the approximate rule is making. As we will see in Theorem 2, ρ is an important quantity in our analysis. One can easily compute ρ from $\vec{g}$ and $\vec{\hat{g}}$ by $\rho = \frac{\vec{\hat{g}}^T\vec{g}}{\vec{g}^T\vec{g}}$.

We express weight updates as discrete dynamical systems (with weights W_h as the state variables):

$$W_h^+ \leftarrow W_h^- + F(W_h^-) = W_h^- - \eta\nabla L(W_h^-), \text{ for BPTT} \tag{4.7}$$

$$W_h^+ \leftarrow W_h^- + \hat{F}(W_h^-) = W_h^- - \eta\tilde{\nabla} L(W_h^-), \text{ for an approximate rule,} \tag{4.8}$$

where η is the learning rate, $\tilde{\nabla}$ is an approximate gradient, and $F : \mathbb{R}^{N^2} \to \mathbb{R}^{N^2}$ (resp. $\hat{F} : \mathbb{R}^{N^2} \to \mathbb{R}^{N^2}$) denotes the map defined by BPTT (resp. an approximate) weight update rule. Notation W^+ and W^- denotes W at the next and current step, respectively. We note in passing that dynamical systems view of weight updates have been used previously [236, 237].

We introduce additional notations before presenting Theorem 2. $J \in \mathbb{R}^{N^2\times N^2}$ (resp. $\widehat{J} \in \mathbb{R}^{N^2\times N^2}$) is the Jacobian of the dynamical system of BPTT (resp. an approximate rule) in Eq. 4.7 (resp. Eq. 4.8). $\lambda_1^J \in \mathbb{R}$ (resp. $\widehat{\lambda_1^J} \in \mathbb{R}$) is the leading eigenvalue for BPTT (resp. an approximate rule) Jacobian. $\lambda_1^H \in \mathbb{R}$ is the leading eigenvalue of the loss' Hessian matrix. $W_B^* \in \mathbb{R}^{N\times N}$ (resp. $W_e^* \in \mathbb{R}^{N\times N}$) is the final fixed point for BPTT (resp. an approximate rule). We now present Theorem 2.

Theorem 2. *Consider an RNN defined in Eq. 4.1 with a single scalar output $\hat{y}$ and least squares loss as in Eq. 4.3 presented only at the last time step T, and weights are updated according to the difference equation for BPTT 4.7 (resp. an approximate rule 4.8) on a single example (stochastic gradient descent) using learning rate η_B (resp. η_e). In the limit of stable fixed point convergence with zero training error, the dominant loss' Hessian eigenvalue attained by BPTT (resp. approximate rule) is bounded by $|\lambda_1^H(W_B^*)| < \frac{1}{\eta_B}$ (resp. $|\lambda_1^H(W_e^*)| < \frac{1}{|\rho|\eta_e}$).*

Proof. Full proof is in Appendix S4.2. Here is a summary of the main steps involved:

1. The Jacobian of BPTT dynamical system (Eq. 4.7) is the loss' Hessian scaled by a constant;

2. Using the above relationship, we can bound $|\lambda_1^H|$ from $|\lambda_1^J| < 1$, which is the condition for a discrete-time dynamical system to converge to a fixed point

3. However, the link between the Jacobian of an approximate update rule and the loss' Hessian is less obvious. Thus, we derive a link between $|\widehat{\lambda_1^J}|$ and $|\lambda_1^J|$, and then apply the step above

□

The consequence of the upper bound derived in Theorem 2 is that truncated gradient rules can converge to minima with a higher dominant Hessian eigenvalue than BPTT, with the leading eigenvalue bound inversely proportional to $|\rho|$ ($|\rho| < 1$ usually). In practice, ρ can vary depending on task settings and in our setup, we observed it to be somewhere between 0.02 and 0.3. We remark that this higher upper bound is consistent with the increased spread of curvature and generalization observed for bio-plausible rules in experiments. Theorem 2 highlights scalar ρ (Eq. 4.6), relative step length along the gradient direction, as an important factor in the curvature bound. To test that, we eliminate the factor of ρ by reducing the learning rule of BPTT such that its step length is matched to that of the three-factor rule. This resulted in the blue curves in Figure 4.4, which is still trained using BPTT but with the update scaled by a factor of ρ. By matching the step length of BPTT and a three-factor rule along the gradient direction, similar convergence behaviors were observed. Similar observations were also made when the matching step experiment was repeated at three times the learning rate for all rules (Appendix Figure S4.8C). **This result then attributes the curvature preference behavior to relative step length along the gradient direction, and thereby indicating a link between curvature and gradient alignment under certain conditions**. Consistent with earlier results, when the step length of BPTT is matched to that of a three-factor rule, its generalization performance also worsened (Appendix Figure S4.2).

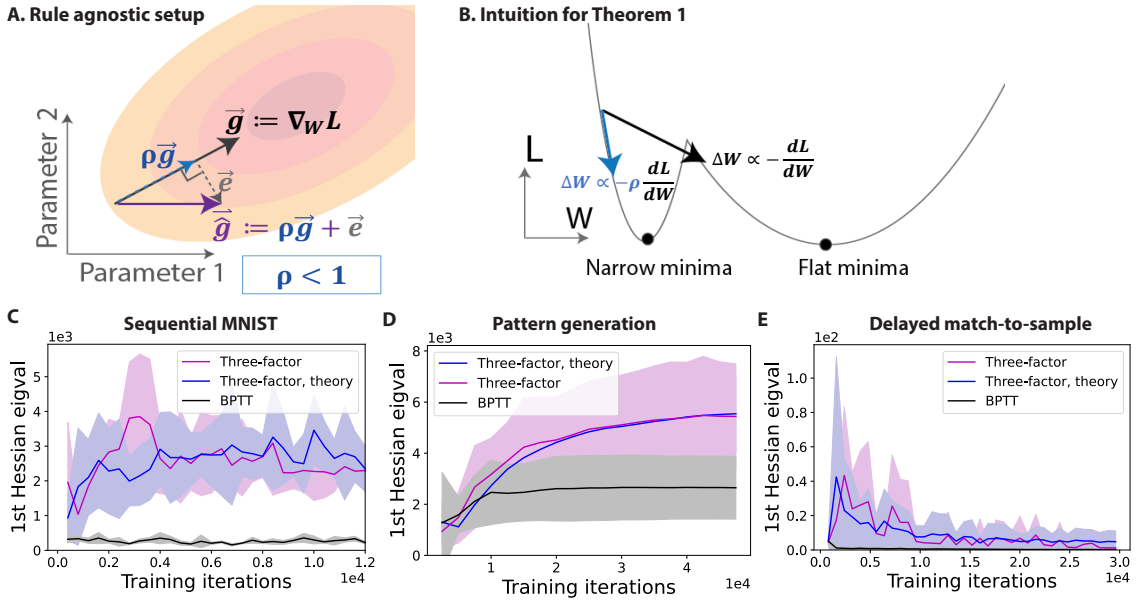

Figure 4.4: Preference for high curvature regions connected to worse gradient alignment under certain conditions. A) An arbitrary gradient approximation, $\tilde{\vec{g}}$, its component along the gradient direction, $\rho\vec{g}$, and orthogonal to the gradient $\vec{e}$ (Eq. 4.6). The scalar ρ represents the relative step length along the gradient direction. B) Illustration for Theorem 2: if gradient $\tilde{\vec{g}}$ is aligned with the sharpest directions but error $\vec{e}$ is not (see Appendix Figure S4.6), smaller step length along the gradient direction can make it harder to step over narrow minima. C-E) Leading loss Hessian eigenvalue v.s. training iteration for the three-factor rule, BPTT, and modified BPTT (three-factor, theory). For the latter, step length along the gradient direction of BPTT was matched to three-factor rule by multiplying BPTT update with a factor of ρ, which recovers curvature trends of the three-factor rule.

We make a few more remarks regarding Theorem 2. ρ is usually less than 1 because otherwise the additional orthogonal component $\vec{e}$ would imply a larger update size for the approximate rule compared to BPTT. This would make the approximate rule more prone to numerical instabilities (Eq. 4.9) and would require a change in solver hyperparameters such as overall learning rate which in turn, influence the update size. In our experiments, when the three-factor rule was scaled up to match the along-gradient update sizes of BPTT, we quickly ran into numerical overflow (values of NaNs in the network), which is expected for a very large learning rate. The exact point for when this numerical instability is reached depends on many factors such as the model, task, numerical precision as well as the consistency of update direction. We have also equated numerical instabilities in simulations as a proxy for situations problematic for the brain in Discussion. Curiously, $\vec{e}$ did not seem to play a helpful role in finding flatter minima; this could be due to that approximation error $\vec{e}$ is not

well aligned with the sharpest directions (Appendix Figure S4.6). In fact, $\vec{e}$ being orthogonal to the leading Hessian eigenvectors is a consequence of the assumptions behind Theorem 2. Despite making assumptions including scalar output, MSE loss and loss available at the last step, our empirical results indicate that our conclusions also extend to other setups: vector output, cross-entropy loss and loss that accumulates over time steps (Figure 4.4). In Appendix S4.2.3, we discuss the generality of Theorem 2 by using quadratic expansion of the loss function and assuming that $\vec{e}$ is orthogonal to the leading Hessian eigenvectors.

One might assume increasing the learning rate of the approximate rules could compensate for the reduced step length due to ρ. However, this can raise other issues. Suppose $\Delta W = -\eta\vec{g}$ for the exact gradient and we increase the learning rate of the approximate update until $\widehat{\Delta W} = -\eta\vec{g} - \eta\vec{e}$ (magnitude of $\vec{e}$ is scaled accordingly), and because $\vec{g} \perp \vec{e}$:

$$\|\widehat{\Delta W}\|^2 = \eta^2\|\vec{g}\|^2 + \eta^2\|\vec{e}\|^2 > \eta^2\|g\|^2 = \|\Delta W\|^2 \tag{4.9}$$

In other words, if approximate updates $\widehat{\Delta W}$ make the same amount of progress as BPTT along the gradient direction, $\widehat{\Delta W}$ would have a larger magnitude due to the orthogonal component $\vec{e}$, and large update magnitude can be very problematic for numerical stability [238]. Thus, the magnitude of $\vec{e}$ limits the learning rate that can be used. **Because of the numerical issues associated with increasing the learning rate for approximate rules (due to $\vec{e}$), the differences in generalization and curvature convergence between rules cannot be reduced by increasing the learning rate for approximate rules.** To balance between this numerical stability issue and the potential benefit of large learning rates, one can consider using a large learning rate early in training to prevent premature stabilization in sharp minima followed by gradual decay to mitigate the stability issue (see Appendix Figure S4.1).

Taken together, Theorem 2 and Eq. 4.9 connect gradient alignment with the curvature of the converged solution under certain conditions. **For an approximation that is aligned poorly with the exact gradient, large orthogonal approximation error vector $\vec{e}$ limit the step length for numerical stability reasons (Eq. 4.9) and small relative step length ρ correspond to a larger curvature bound (Theorem 2).**

4.4 Conclusion

While developing bio-plausible learning rules is of interest for both answering neuroscience questions and searching for more efficient training strategies for artificial networks, the generalization properties of solutions found by these rules are severely underexamined. Through various well-known machine learning and working memory tasks, we first demonstrate empirically that existing bio-plausible temporal credit assignment rules attain worse generalization performance, which is consistent with their tendency to converge to high-curvature regions in loss landscape. Second, our theoretical analysis offers an explanation for this preference for high curvature regions based on worse alignment to the true gradient. This regime corresponds to the situation where the step length along the gradient direction is small and the approximation error vector is large. Finally, we test this theory empirically by matching the relative step length along the gradient direction resulting in similar convergence behavior (Figure 4.4).

4.5 Discussion

Our study — a stepping stone toward understanding biological generalization using deep learning methods — raises many exciting questions for future investigations, both on the front of stronger deep learning theory and more sophisticated biological ingredients.

Deep learning theory and its implications: In this study, we investigate generalization properties using loss landscape flatness, a promising predictor with recent rigorous connection to generalization gap [14]. Yet, to what extent can flatness explain generalization is still an open question in deep learning. For instance, curvature is a local measure, which means that its informativeness of robustness against global perturbation is limited. Moreover, the theoretical association between flatness and generalization is provided in the form of upper bound [14, 139] and the bound may not always be tight, which is consistent with more variability in the generalization gap for bio-plausible learning rules but offers less predictive power. Despite observing a significant correlation between the generalization gap and a curvature-based measure in Figure 4.2, the relationship appears to be messy, suggesting

other factors involved in explaining generalization. Given that developing better predictors of generalization is still a work in progress [135], we anticipate stronger theoretical tools to be applied for studying biological generalization in the future. Our results are also consistent with existing findings linking learning rate to loss landscape curvature in deep networks [221]. In the case of bio-plausible gradient approximations, small step length along the gradient direction cannot be compensated by increasing the learning rate, as that would inadvertently increase the error vector, causing numerical issues (Eq. 4.9). Curiously, the approximation error vector $\vec{e}$ did not seem to play a role other than restricting the learning rate. While it is well-known that stochastic gradient noise (SGN) can help with finding flat minima due to the alignment of SGN covariance and Hessian near minima [220, 239–241], that may not apply to approximation error vector $\vec{e}$ resulting primarily from temporal truncation of the gradient (see Figure 4.1A and Appendix Figure S4.6). This indicates that noise with different properties (e.g. different directions) could affect generalization differently, thereby motivating future investigations into how a broad range of biological noises — which may differ from noise in ML optimization (e.g. SGN) — can impact generalization.

Moreover, our results are closely related to a series of studies that examined the "catapulting" behavior in learning [242–246]. This can happen when the second order Taylor term of the loss function would dominate over the first, which would cause learning to cross the threshold for step size stability and "catapult" into a flatter region that can accommodate the step size. If the truncation noise is only aligned with the eigendirections associated with negligible eigenvalues, then it can only have limited contributions to the second-order Taylor term. On top of that, the orthogonal noise term would require a smaller learning rate to be used to avoid numerical issues, as explained earlier. Overall, these would lead to a weaker second-order Taylor term relative to the first for bio-plausible temporal credit assignment rules, which would then increase the threshold for step size stability. This increased stability is closely tied to the greater dynamical stability (for the weight update difference equation) of approximation rules predicted by Theorem 2, due to the correspondence between loss' Hessian matrix and the Jacobian matrix of the weight update difference equation (the correspondence is explained in Theorem 2 proof in Appendix S4.2).

Toward more detailed biological mechanisms: On the front of more sophisticated biological ingredients that may improve generalization performance, we see two lines of approaches: **1-** develop bio-plausible learning rules that align better with the gradient, as suggested by our rule-agnostic analysis (Theorem 2, Figure 4.4); and **2-** instead of studying learning rules in isolation, consider also other neural circuit components [247–255] that could interact with the learning rule. An important component would be the architecture, including connectivity structure and neuron model, found through evolution [252] (see also [256]). To address our main question, this study varies learning rules while holding data, objective function and architecture constant (see [61]). However, these different components can interact, and more sophisticated architecture can facilitate task learning [125, 256–262]. Given the exploding parameter space resulting from such interactions, we believe it requires careful future analysis and is outside of the scope for this one paper.

Additionally, learning rate modulation [263, 264] could be one of many possible remedies employed by the brain. We conjecture that neuromodulatory mechanisms could be coupled with these learning rules to improve the convergence behavior through our scheduled learning rate experiments (Appendix Figure S4.1), where an initial high learning rate could prevent the learning trajectory from settling in sharp minima prematurely followed by gradual decay to avoid instabilities. One possible way to realize such learning rate modulation could be through serotonin neurons via uncertainty tracking, where the learning rate is high when the reward prediction error is high (this can happen at the beginning of learning) [265]. Since the authors of [265] showed that inhibiting serotonin led to failure in learning rate modulation, we conjecture that such inhibition might have an impact on the generalization performance of learning outcomes. On the topic of balancing numerical instabilities and potential advantages of large learning rates, while the analog nature of biology may seem to avoid finite precision representation in digital computers that give rise to numerical instabilities, the same problems that lead to numerical instabilities in digital computers, such as big ranges between quantities added or multiplied, remains an issue for biology since quantities must be stored in noisy activity patterns of neurotransmitter release. Future investigations could investigate potential homeostatic mechanisms that regulate biological quantities to avoid such instabilities, thereby enabling larger "learning rates" to be used so as to find flatter minima. Other ingredients

for future investigations could include intrinsic noise with certain structures [249, 266, 267] (including directions and bias/variance properties) that would make them more favorable for generalization. Taken together, we hope to see follow-up investigations — riding on the rapid advancements both at the front of deep learning theory and sophisticated biological mechanisms — into how the brain attains solutions that generalize.

S4.1 Methods

S4.1.1 Hessian eigenspectrum analysis

As mentioned, we focus on the leading Hessian eigenvalue because it has been used previously [15, 219] and is very feasible for both empirical [216, 268] and theoretical analyses (Theorem 2). The leading Hessian eigenvalue can be computed by performing power iterations on the Hessian vector product without knowing the full Hessian matrix (Algorithm 2 in [216]). To find multiple top eigenvalues, e.g. top 200 eigenvalues, one can use the generalized power method via QR decomposition [269]. We focused on the loss' Hessian for recurrent weights but observed a similar trend for input weights as well. We set the tolerance for stopping to $1e-6$.

Due to the scale-dependence issue of Hessian spectrum [231], we also used scale-independent measures. For instance, we examined the power-law decay coefficient for the Hessian eigenvalues (Figure S4.3 in Appendix). We also looked at the recently proposed relative flatness measure [14] (Figure S4.4 in Appendix). We used the code in [270] to fit a power-law distribution to the top 200 eigenvalues. We found similar results had we chosen the top 50 or 100 eigenvalues instead, and 200 was chosen mainly due to computational load. We note that the link of power-law decay to generalization has also been examined in some recent studies [218, 271, 272].

S4.1.2 Network setup and learning rule implementations

Neuron Model: We consider a discrete-time implementation of a rate-based recurrent neural network (RNN) similar to the form in [124]. The model denotes the internal hidden state as h_t and the observable states, i.e. firing rates, as $f(h_t)$ at time t, and we use ReLU

activation for f. The dynamics of those states are governed by

$$h_{t+1} = \alpha h_t + (1-\alpha)\left(W_h f(h_t) + W_x x_t\right), \tag{S4.1}$$

where $\alpha = e^{-dt/\tau_m}$ denotes the leak factor for simulation time step dt and membrane time constant τ_m, W_h denotes the weight of the recurrent synaptic connection, W_x denotes the strength of the input synaptic connection and x_t denotes the external input at time t. we use subscripts to represent indices of neurons and time steps. For instance, $h_{i,t}$ represents the hidden activity h of neuron i at time t. $W_{h,ij}$ represents the $(ij)^{th}$ entry of recurrent weight matrix W_h. Model in Eq. S4.1 was used for the sequential MNIST and pattern generation tasks.

We mention in passing that the choice of ReLU activation, which has a discontinuous first derivative, means that the loss Hessian matrix is not guaranteed to be symmetric. A real matrix that is not symmetric can have complex eigenvalues come in conjugate pairs, and if they were amongst the top eigenvalues, power iterations may not converge. However, all iterations have converged in our experiments as mentioned above. Also, because of potential technical issues resulting from non-symmetric Hessian matrices, we foresee challenges in applying our methodology to spiking neural networks (SNNs), which have discontinuous activation functions. Due to the energy efficiency and biological realism of SNNs [76, 273–282], we believe extending to SNNs is an important future direction.

For the delayed match to sample task, which is a working memory task, it was found in [11] and [4] that units with an adaptive threshold as an additional hidden variable can play an important role in the computing capabilities of RNNs. Thus, we implemented adaptive threshold neuron units [260] for that task. In our rate-based implementation, this turns out to be a simple addition of a second hidden variable b_t that represents the dynamic threshold component:

$$\begin{aligned} h_{t+1} &= \alpha h_t + (1-\alpha)\left(W_h f(h_t - b_t) + W_x x_t\right), \\ b_{t+1} &= \beta b_t + (1-\beta) f(h_t - b_t), \end{aligned} \tag{S4.2}$$

where $b_{j,t}$ denotes the dynamic threshold that adapts based on past neuron activity. The decay factor β is given by e^{-dt/τ_b} for simulation time step dt and adaptation time constant

τ_b, which is typically chosen on the behavioral task time scale [11].

Network output and loss function:

Readout $\hat{y}$ is defined as

$$\hat{y} = \langle w, f(h_t) \rangle \tag{S4.3}$$

for readout weights w.

We quantify how well the network output $\hat{y}$ matches the desired target y using loss function L, which is defined as

$$L(W_h) = \begin{cases} \frac{1}{2TB} \sum_{i=1}^{B} \sum_{t=1}^{T} \sum_{k=1}^{N_{out}} (\hat{y}_{k,t}^{(i)} - y_{k,t}^{(i)})^2, \text{ for regression tasks} \\ \frac{-1}{TB} \sum_{i=1}^{B} \sum_{t=1}^{T} \sum_{k=1}^{N_{out}} \pi_{k,t}^{(i)} log \hat{\pi}_{k,t}^{(i)}, \text{ for classification tasks} \end{cases} \tag{S4.4}$$

for target output y, task duration T, N_{out} output neurons and batch size B. $\pi_{k,t}$ is the one-hot encoded target and $\hat{\pi}_{k,t} = \text{softmax}_k(\hat{y}_{1,t}, \ldots, \hat{y}_{N_{OUT},t}) = \exp(\hat{y}_{k,t}) / \sum_{k'} \exp(\hat{y}_{k',t})$ is the predicted category probability.

Biological gradient approximations (truncation-based)

The goal of this subsection is to explain where the approximation happens for each of the bio-plausible learning rules. For full details regarding these rules, we encourage the reader to refer to the respective references. We start by writing down the gradient in terms of real-time recurrent learning (RTRL) factorization:

$$\frac{\partial L}{\partial W_{h,ij}} = \sum_{l,t} \frac{\partial L}{\partial h_{l,t}} \frac{\partial h_{l,t}}{\partial W_{h,ij}}, \tag{S4.5}$$

Key problems that RTRL poses to biological plausibility and computational cost reside in the second factor $\frac{\partial h_{l,t}}{\partial W_{h,ij}}$ that arises during the factorization of the gradient (Eq. S4.5). The factor $\frac{\partial h_{l,t}}{\partial W_{h,ij}}$ keeps track of all recursive dependencies of $h_{l,t}$ on weight $W_{h,ij}$ arising from recurrent connections. These recurrent dependencies can be obtained recursively as follows:

$$\begin{aligned} \frac{\partial h_{l,t}}{\partial W_{h,ij}} &= \frac{\partial h_{j,t}}{\partial W_{h,ij}} + \sum_m \frac{\partial h_{l,t}}{\partial h_{m,t-1}} \frac{\partial h_{m,t-1}}{\partial W_{h,ij}} \\ &= \frac{\partial h_{l,t}}{\partial W_{h,ij}} + \frac{\partial h_{l,t}}{\partial h_{l,t-1}} \frac{\partial h_{l,t-1}}{\partial W_{h,ij}} + \underbrace{\textstyle\sum_{m \neq l} W_{h,lm} f'(h_{m,t-1}) \frac{\partial h_{m,t-1}}{\partial W_{h,ij}}}_{\text{depends on all weights } W_{h,lm}}. \end{aligned} \tag{S4.6}$$

Thus, the factor $\frac{\partial h_{l,t}}{\partial W_{h,ij}}$ poses a serious problem for biological plausibility: it involves **nonlocal** terms that should be inaccessible to neural circuits, i.e. that knowledge of all other weights in the network is required in order to update the weight $W_{h,ij}$.

RFLO [140] (labeled as "RF Three-factor") and **symmetric e-prop** [4] (labeled as "Three-factor") seek to address this by truncating the expensive nonlocal terms in Eq. 2.17 so that the updates to weight $W_{h,ij}$ would only depend on pre- and post-synaptic activity:

$$\widehat{\frac{\partial h_{l,t}}{\partial W_{h,ij}}} = \begin{cases} \frac{\partial h_{i,t}}{\partial W_{h,ij}} + \frac{\partial h_{i,t}}{\partial h_{i,t-1}} \widehat{\frac{\partial h_{i,t-1}}{\partial W_{h,ij}}}, & l = i \\ 0, & l \neq i \end{cases} \tag{S4.7}$$

which results in a much simpler factor than the triple tensor in Eq. 2.17.

After the truncation, RFLO and e-prop implement:

$$\widehat{\frac{\partial L}{\partial W_{h,ij}}} = \sum_t \frac{\partial L}{\partial h_{i,t}} \widehat{\frac{\partial h_{i,t}}{\partial W_{h,ij}}}, \tag{S4.8}$$

$$\widehat{\frac{\partial h_{i,t}}{\partial W_{h,ij}}} = \frac{\partial h_{i,t}}{\partial W_{h,ij}} + \frac{\partial h_{i,t}}{\partial h_{i,t-1}} \widehat{\frac{\partial h_{i,t-1}}{\partial W_{h,ij}}}. \tag{S4.9}$$

The main difference between symmetric e-prop and RFLO implementation is that symmetric feedback is used for symmetric e-prop, i.e. output weight w is used as the feedback weight for the $\frac{\partial E}{\partial h}$, whereas RFLO uses fixed random feedback weights [190] for greater biological plausibility. We note in passing that the authors of e-prop have tested their formulation with fixed random feedback weights as well. **MDGL** [11] also truncates RTRL, but it restores some of the non-local dependencies – those within one connection step — that could potentially be communicated via mechanisms similar to the abundant cell-type-specific local modulatory signaling unveiled by recent transcriptomics data [191, 192]. With that, the expensive memory trace term in Eq. 2.17 becomes

$$\frac{\partial h_{l,t}}{\partial W_{h,ij}} \approx \begin{cases} W_{h,li} f'(h_{i,t-1}) \widehat{\frac{\partial h_{i,t-1}}{\partial W_{h,ij}}}, & i \neq l \\ \frac{\partial h_{i,t}}{\partial W_{h,ij}} + \frac{\partial h_{i,t}}{\partial h_{i,t-1}} \widehat{\frac{\partial h_{i,t-1}}{\partial w_{ij}}}, & i = l \end{cases} \tag{S4.10}$$

MDGL involves one additional approximation: replace $W_{h,li}$ with type-specific weights W_{ab} to mimic the cell-type-specific nature of local modulatory signaling (for cell i in group a and cell j in group b, where $a, b \in C$ for a total of C cell groups). For simplicity, we

just used $W_{ab} = W_{li}$, i.e. without cell-type approximation. This results in overall MDGL implementation as

$$\widehat{\frac{\partial L}{\partial W_{h,ij}}} = \sum_t \frac{\partial L}{\partial h_{i,t}} \widehat{\frac{\partial h_{i,t}}{\partial W_{h,ij}}} + \widehat{\frac{\partial h_{i,t-1}}{\partial W_{h,ij}}} \sum_l W_{h,li} \frac{\partial L}{\partial h_{l,t}}, \tag{S4.11}$$

Interpretation of the above update rule in terms of biological processes can be found in the MDGL paper [11, 131].

We note that input, recurrent and output weights were all being trained. This section illustrates the approximate gradient for updating recurrent weights W_h, and similar expressions were obtained for updating input weights W_x. The approximations, however, did not apply to output weights, as the gradient for that would not violate the aforementioned issue of nonlocality (Eq. 2.17).

S4.1.3 Simulation details

We used TensorFlow [133] version 1.14 and based it on top of [283]. We modified the code for rate-based neurons (Eq. S4.1 and S4.2). [1] We used the code in [270] for the power-law analysis (Figure S4.3 in Appendix). SGD optimizer was used to study the effect of gradient approximation in isolation without the complication of additional factors, as Adam optimizer with adaptive learning rate could convolute our matching step length experiments in Figure 4.4. That said, we verified that the curvature convergence behavior is also observed for Adam optimizer (Figure S4.7 in Appendix. Learning rates were optimized by picking within $\{3e-4, 1e-3, 3e-3, 1e-2, 3e-2, 1e-1\}$ for each algorithm. For the sequential MNIST task, we explored batch sizes within $\{64, 256, 1024\}$. For the sequential MNIST task, these hyperparameters were optimized based on validation performance (the validation set loaded using *tensorflow.examples.tutorials.mnist*). For the two other tasks, these hyperparameters were optimized based on the training performance, but we also tried optimizing on the test set and observed similar trends. Trainings were stopped when both the loss and leading Hessian eigenvalue stabilized. As stated, we repeated runs with different random initialization to quantify uncertainty and weights were initialized similarly as in [140].

[1] Our code is available: `https://github.com/Helena-Yuhan-Liu/BiolHessRNN`.

Simulations were completed on a computer server with x2 20-core Intel(R) Xeon(R) CPU E5-2698 v4 at 2.20GHz. The average time to complete one run of sequential MNIST, pattern generation and delayed match to sample tasks in Figure 4.3 were approximately 2 hours, 1 hour and 1 hour, respectively. Since the computation of second order gradient becomes prohibitively expensive as sequence length T becomes large, all tasks involved no more than 50 time steps. For instance, this was achieved for the sequential MNIST task using the row-by-row implementation. Using fewer time steps, however, should not affect the general trend as the gradient truncation effects were still significant. Because of the use of fewer steps, we dropped the leak factor α in Eq. S4.1 (i.e. set $\alpha = 0$).

For the matching step length experiments (Figure 4.4), we simply obtained $\rho = \frac{\vec{\tilde{g}}^T \vec{g}}{\vec{\tilde{g}}^T \vec{\tilde{g}}}$ for the three-factor learning rule and scaled BPTT updates by that amount. For scheduled learning rate experiments (Figure S4.1), the additional hyperparameters included initial learning rate, decay percentage and decay frequency. We used an initial learning rate that was three times the uniform rate (used in other figures) and decay the learning rate by 80% every X iterations, where X was roughly the total number of training iterations (used in other figures) divided by 30. Since the point of that figure was to show that learning rate scheduling could lead to flatter minima than using a fixed learning rate, we did not search extensively across these additional hyperparameters as the first set of hyperparameters we tried was enough to demonstrate that point.

For the pattern generation task, our network consisted of $N = 30$ neurons described in Eq. S4.1. Input to this network was provided by a random Gaussian input ($N_{in} = 1$). The fixed target signal had a duration of 50 steps and was given by the sum of four sinusoids, with a fixed period of 10, 40, 70 and 100 steps. For learning, we used the mean squared loss function. Training for this task used full batch. For testing, we perturbed the input with additive zero-mean Gaussian noise (with σ picked uniformly between 0 and 0.2 across runs), to mimic the situation where the agent had to faithfully produce the desired pattern even under perturbations. Unlike the other tasks, this task measures accuracy by mean squared error, for which the lower the better. To maintain the convention of a higher generalization gap being worse, the generalization gap for this task was computed by test error minus train error.

For the delayed match to sample task, our network consisted of $N = 100$ neurons, which include 50 neurons with (Eq. S4.2) and 50 neurons without (Eq. S4.1) threshold adaptation. The task involved the presentation of two sequential cues, each taking on a binary value, lasting 2 steps and separated by a delay of 16 steps. Input to this network was provided by $N_{in} = 2$ neurons. The first (resp. second) input neuron sent a value of 1 when the presented cue took on a value of 1 (resp. cue 0), and 0 otherwise. The network was trained to output 1 (resp. 0) when the two cues have matching (resp. non-matching) values. For learning, we used the cross-entropy loss function and the target corresponding to the correct output was given at the end of the trial. Training for this task used full batch. For testing, we tested on increased delay, with the period picked uniformly between the training delay period and twice the training delay period, to mimic situations where the animal has to hold the memory longer than it did during the learning phase.

For the sequential MNIST task [10], our network consisted of $N = 128$ neurons described in Eq. S4.1. Input to this network was provided by $N_{in} = 28$ units that represented the grey-scaled value of a single row, totaling 28 steps and the network prediction was made at the last step. For learning, we used the cross-entropy loss function and the target corresponding to the correct output was given at the end of the trial. For testing, we used the existing MNIST test set [10] with additive zero-mean Gaussian input noise. As mentioned, this task did not train with a full batch but we found the trend to hold across different batch sizes.

Finally, we note that comparisons between BPTT and approximate rules were done at comparable training accuracies for the pattern generation and delayed match to sample tasks. For the sequential MNIST task, the three-factor rule achieved only around 70% training accuracy, but the training accuracy did not explain the curvature convergence behavior. To see this, while three-factor theory (blue in Fig 4), which corresponds to BPTT with reduced step length, achieves an accuracy of $> 95\%$ but still attains similar curvature convergence to that of the three-factor rule.

S4.2 Theorem 2

S4.2.1 Proof of Theorem 2

Proof. First, we note that the Jacobian of the dynamical system for BPTT update (Eq. 4.7) is simply the loss Hessian scaled by $-\eta$. This implies that

$$|\lambda_1^J| = \eta_B |\lambda_1^H|, \tag{S4.12}$$

where we remind the reader that $\lambda_1^J \in \mathbb{R}$ (resp. $\widehat{\lambda_1^J} \in \mathbb{R}$) is the leading eigenvalue for BPTT (resp. an approximate rule) Jacobian; $\lambda_1^H \in \mathbb{R}$ is the leading eigenvalue of the loss' Hessian matrix.

Second, with the assumption of a single output $\hat{y}$, loss presented only at the last time step T and stochastic gradient descent (updates on a single data example as opposed to batch updates), least squares loss in Eq. 4.3 can be simplified to:

$$L(W_h) = \frac{1}{2}(\hat{y} - y)^2, \tag{S4.13}$$

resulting in the following difference equations for discrete dynamical systems defined by BPTT (Eq. 4.7) and an approximate rule (Eq. 4.8):

$$\Delta W_h|_{\text{BPTT}} = F(W_h) = -\eta_B(\hat{y} - y)\nabla\hat{y}, \tag{S4.14}$$

$$\Delta W_h|_{\text{approximate}} = \hat{F}(W_h) = -\eta_e(\hat{y} - y)\tilde{\nabla}\hat{y}, \tag{S4.15}$$

where we remind the reader that $\tilde{\nabla}$ is the notation for an approximate gradient.

We then compute the Jacobian of the difference equations above:

$$J = -\eta_B \left(\nabla\hat{y}\nabla\hat{y}^T + (\hat{y} - y)\nabla^2\hat{y}\right) \quad \text{(for BPTT)} \tag{S4.16}$$

$$\widehat{J} = -\eta_e \left(\nabla\hat{y}\tilde{\nabla}\hat{y}^T + (\hat{y} - y)\nabla\tilde{\nabla}\hat{y}\right) \quad \text{(for an approximate rule).} \tag{S4.17}$$

In the limit of zero error $(y - \hat{y}) = 0$, i.e. close to an optimum, the term involving $(y - \hat{y})$ becomes negligible. That simplifies the Jacobian to

$$J \approx -\eta_B \nabla\hat{y}\nabla\hat{y}^T$$

$$\widehat{J} \approx -\eta_e \nabla\hat{y}\tilde{\nabla}\hat{y}^T. \tag{S4.18}$$

In this case, J and $\widehat{J}$ are **rank-1** matrices. A rank-1 square matrix has only one nonzero eigenvalue, and by inspection, that one eigenvalue is

$$\text{For } J: |\lambda_1^J(W_B^*)| \quad = \eta_B \nabla\hat{y}^T \nabla\hat{y}\big|_{W_B^*} \tag{S4.19}$$

$$\rightarrow |\lambda^H| \overset{\text{(S4.12)}}{=} |\lambda_1^J|/\eta_B = \nabla\hat{y}^T\nabla\hat{y} \tag{S4.20}$$

$$\begin{aligned}\text{For } \widehat{J}: \quad |\widehat{\lambda_1^J}(W_e^*)| &= \eta_e |\tilde{\nabla}\hat{y}^T\nabla\hat{y}|\Big|_{W_e^*} \\ &\overset{(a)}{=} |\eta_e \rho \nabla\hat{y}^T\nabla\hat{y}|\big|_{W_e^*} \\ &\overset{\text{(S4.20)}}{=} |\rho\eta_e \lambda_1^H(W_e^*)|,\end{aligned} \tag{S4.21}$$

where equality (a) is explained as follows. We first remind the reader that ρ is defined such that $\vec{\tilde{g}} = \rho\vec{g} + \vec{e}$ (Eq. 4.6). For the case of a scalar output $\hat{y}$, $\vec{\tilde{g}} = \frac{\partial L}{\partial \hat{y}}\tilde{\nabla}\hat{y}$ and $\vec{g} = \frac{\partial L}{\partial \hat{y}}\nabla\hat{y}$. So if we divide both sides of $\vec{\tilde{g}} = \rho\vec{g} + \vec{e}$ by $\frac{\partial L}{\partial \hat{y}}$ we get $\tilde{\nabla}\hat{y} = \rho\nabla\hat{y} + \vec{e}/\frac{\partial L}{\partial \hat{y}}$. Since we have $\vec{e} \perp \vec{g}$ by definition, then $\vec{e}^\top\nabla\hat{y} = 0$ because $\vec{g}$ is just a scaled $\nabla\hat{y}$ when the output is a scalar. This leads to $\tilde{\nabla}\hat{y}^T\nabla\hat{y} = (\rho\nabla\hat{y} + \vec{e}/\frac{\partial L}{\partial \hat{y}})^\top\nabla\hat{y} = \rho\nabla\hat{y}^T\nabla\hat{y}$.

Since we assume the gradient descent dynamical system converges to an optimum, this corresponds to an asymptotic stable fixed point. Hence, $|\lambda_1^J| < 1$ and $|\widehat{\lambda_1^J}| < 1$, which implies:

$$|\lambda_1^J(W_B^*)| < 1 \overset{\text{(S4.12)}}{\rightarrow} \eta_B|\lambda_1^H(W_B^*)| < 1 \rightarrow |\lambda_1^H(W_B^*)| < \frac{1}{\eta_B} \tag{S4.22}$$

$$|\widehat{\lambda_1^J}(W_e^*)| < 1 \overset{\text{(S4.21)}}{\rightarrow} |\rho\eta_e\lambda_1^H(W_e^*)| < 1 \rightarrow |\lambda_1^H(W_e^*)| < \frac{1}{|\rho|\eta_e}. \tag{S4.23}$$

□

S4.2.2 Discussion on tightness of the bound

Following the derivation, it is clear that the tightness of the bound will depend on how close the magnitude of leading Jacobian eigenvalue is to 1 upon convergence. That is related to the distribution of minima flatness along the loss landscape, which impacts the probability of a rule converging to a minima with flatness in a certain range. Such distribution is likely problem dependent. If the loss were convex, there would just be one minimum and the question of minima preference would become irrelevant.

S4.2.3 Discussion on generality of Theorem 2

The proof above examines a special case where the Jacobian of weight update equations becomes rank-1. We remind the reader that for the case of multivariate loss, higher rank cases or batch updates, we would not have arrived at the rank-1 Jacobian step (Eq. S4.18). For many tasks considered in neuroscience, the rank 1 case can apply, which explains the validity of Theorem 2. We also note in passing that for the case of stochastic gradient descent, the Hessian Jacobian correspondence (Eq. S4.12) would point to loss' Hessian matrix evaluated on a single example, which could reflect the robustness against perturbing that particular example. Moreover, simulation results show our conclusion holds in higher rank cases (Figure 4.4). The challenge of generalizing the proof to higher Jacobian rank case is that we are no longer guaranteed that the leading eigenvectors of BPTT Jacobian coincide with the leading eigenvectors of an approximate rule Jacobian. Thus, it becomes much harder to relate $|\widehat{\lambda_1^J}|$ and $|\lambda_1^J|$. Rather than providing further proof, we provide an intuition for why our conclusion — where the convergence behavior between rules differs by their step length along the gradient direction — can hold in higher rank cases under Assumption 1.

Assumption 1. *Approximation error vector $\vec{e}$ (but not $\vec{g}$) lies orthogonal to the subspace spanned by the leading Hessian eigenvectors. Here, leading Hessian eigenvectors refer to the eigenvectors corresponding to the outlier Hessian eigenvalues in light of the well-known observation that there exists only a few large (outlier) eigenvalues and the rest are near zero [214, 215, 268]).*

The ramification of Assumption 1 is that $\vec{e}$ will lie in the subspace spanned by eigenvectors corresponding to tiny eigenvalues, making $H\vec{e}$ tiny. In the extreme scenario where $\vec{e}$ lies in the null space of H, $H\vec{e}$ would be 0. We verify this assumption numerically in Figure S4.6. We saw from the proof above that this assumption is automatically satisfied in the rank-1 Jacobian case. We remark that this assumption should not hold for stochastic gradient noise (SGN), as the SGN covariance matrix is well aligned with the Hessian matrix near a minima [220]. This could be why $\vec{e}$, unlike stochastic gradient noise, does not seem to be contributing much to escaping narrow minima.

We consider the case of small enough weight updates such that the loss surface can be approximated using second-order Taylor expansion. Thus, the loss change after one update becomes:

$$\begin{aligned}\Delta L &\approx \Delta W^T \vec{g} + \frac{1}{2}\Delta W^T H \Delta W \\ &= -\eta_B \vec{g}^T \vec{g} + \frac{1}{2}\eta_B^2 \vec{g}^T H \vec{g}, \quad \text{(for exact rule)} \qquad \text{(S4.24)} \\ \widehat{\Delta L} &\approx \widehat{\Delta W}^T \vec{g} + \frac{1}{2}\widehat{\Delta W}^T H \widehat{\Delta W} \\ &= -\eta_e \vec{\hat{g}}^T \vec{g} + \frac{1}{2}\eta_e^2 \vec{\hat{g}}^T H \vec{\hat{g}}. \quad \text{(for an approximate rule)} \qquad \text{(S4.25)}\end{aligned}$$

We next focus on the first- and second-order Taylor terms (T_1 and T_2) for the exact rule as well as the terms ($\widehat{T_1}$ and $\widehat{T_2}$) for an approximate rule:

$$T_1 := \eta_B \vec{g}^T \vec{g}, \widehat{T_1} := \eta_e \vec{\hat{g}}^T \vec{g}, T_2 := \frac{1}{2}\eta_B^2 \vec{g}^T H \vec{g}, \widehat{T_2} := \frac{1}{2}\eta_B^2 \vec{\hat{g}}^T H \vec{\hat{g}},$$

and we note that first and second Taylor terms can determine how likely the update will be trapped in a local minimum:

$$\begin{aligned}&\Delta L < 0 \text{ (enables descend)} \quad \rightarrow T_1 > T_2 \\ &\Delta L > 0 \text{ (restricts convergence)} \quad \rightarrow T_1 < T_2 \\ &\widehat{\Delta L} < 0 \text{ (enables descend)} \quad \rightarrow \widehat{T_1} > \widehat{T_2} \\ &\widehat{\Delta L} > 0 \text{ (restricts convergence)} \quad \rightarrow \widehat{T_1} < \widehat{T_2}. \qquad \text{(S4.26)}\end{aligned}$$

Given their central role in determining convergence, we compare these terms between exact gradient descent learning and an approximate rule. For the first Taylor term (T1), it is easy to see that:

$$\vec{\hat{g}}^T \vec{g} = \rho \vec{g}^T \vec{g}.$$

For the second Taylor term (T2) and if H is symmetric:

$$\begin{aligned}\vec{\hat{g}}^T H \vec{\hat{g}} &= \rho^2 \vec{g}^T H \vec{g} + 2\rho \vec{g}^T \underbrace{H\vec{e}}_{\approx 0} + \rho^2 \vec{e}^T \underbrace{H\vec{e}}_{\approx 0, \text{ Assumption 1}} \\ &\approx \rho^2 \vec{g}^T H \vec{g}. \qquad \text{(S4.27)}\end{aligned}$$

To match the convergence behavior between exact gradient descent and an approximate rule (Eq. S4.26) on a (locally) second-order loss surface, we can make (T_1, T_2) approximately equal to $(\widehat{T_1}, \widehat{T_2})$ by setting $\eta_B = \rho\eta_e$, which predicts our numerical results (Figure 4.4). We note that if Assumption 1 were not satisfied, then the above might not hold. We note in passing that if we can satisfy Assumption 1 without being near an optimum, then we may not need the negligible training error assumption.

S4.3 Additional Simulations

In the last paragraph in Discussion, we discussed how learning rate modulation could be one of the potential remedies used by the brain. We also explained how learning rate modulation could serve as a balance between the potential benefits of a large learning rate and the numerical stability issue mentioned shortly after the presentation of Figure 4.4 and Theorem 2 in Results. In Appendix Figure S4.1, we used a large learning rate early in training to prevent premature stabilization in sharp minima followed by gradual decay to mitigate the stability issue. With this remedy, we observed a reduction in the curvature of the converged solution and an improvement in generalization performance. This result also connects with the finding that sensory depletion during critical periods in training deep networks, which can be related to a small learning rate early in training, can impair learning and yield convergence to sharp minima [284]. However, it is important to note that this strategy does not correct the problem; the gap still exists compared to BPTT, suggesting room for further research.

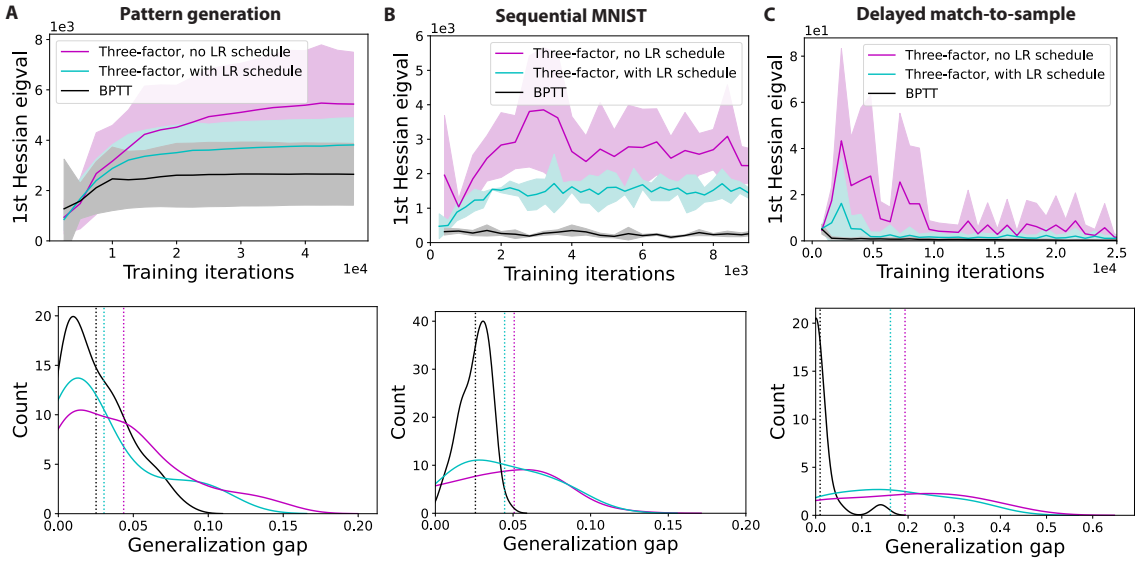

Figure S4.1: Learning rate modulation as a possible remedy of the problem. We increased the learning rate at the beginning of training to prevent the three-factor rule from stabilizing in sharp minima prematurely, followed by a gradual decay to prevent instability. The top panels show this strategy helps to reduce the curvature of the converged solution. The bottom panels show this leads to a slight improvement in the generalization gap (vertical lines denote distribution mean). However, it is important to note that this strategy does not correct the problem; the gap still exists compared to BPTT, suggesting room for further research. Plotting conventions follow that of the previous figures.

We present additional simulations referred to in the main text. In Figure 4.4, we attributed the convergence to high curvature regions to reduced along-gradient step length. In Appendix Figure S4.2, we confirm that such high curvature convergence indeed corresponds to worsened generalization performance, thereby linking reduced along-gradient step length to worsened generalization performance. As mentioned in the main text, due to the scale-dependence issue of Hessian spectrum [231], we also used scale-independent measures. For instance, we examined the power-law decay coefficient for the Hessian eigenvalues (Appendix Figure S4.3). We also looked at the recently proposed relative flatness measure [14] (Appendix Figure S4.4). These additional measures support the trends observed before: BPTT converges to lower curvature regions compared to the three-factor rule. We also observe that the tendency to approach high curvature regions seems to be a shared problem for temporal truncations of the gradient (Appendix Figure S4.5).

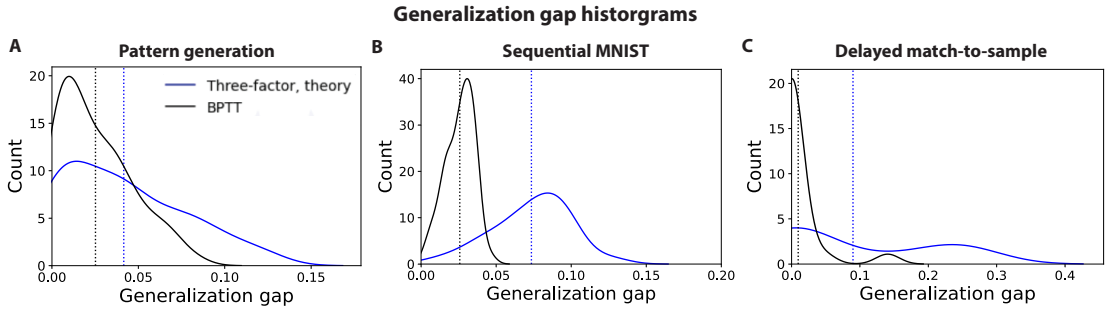

Figure S4.2: Modified BPTT (three-factor, theory) resulted in worse and more variable generalization performance. Here, we follow the convention of previous generalization gap histogram plots and investigated the generalization performance of modified BPTT (three-factor, theory) in Figure 4.4.

In response to our discussion on the potential impact of noise direction (see explanation shortly after the presentation of Theorem 2, Discussion section and Appendix S4.2.3), we confirm that the error vector $\vec{e}$ is significantly less aligned with the leading Hessian eigenvectors relative to the gradient vector $\vec{g}$ (Appendix Figure S4.6). As explained in Methods, we used SGD optimizer due to confounding factors in Adam optimizer that could convolute our matching step length analysis in Figure 4.4. We observe similar curvature convergence trends as in Figure 4.3 when we repeated the experiments with Adam optimizer in Appendix

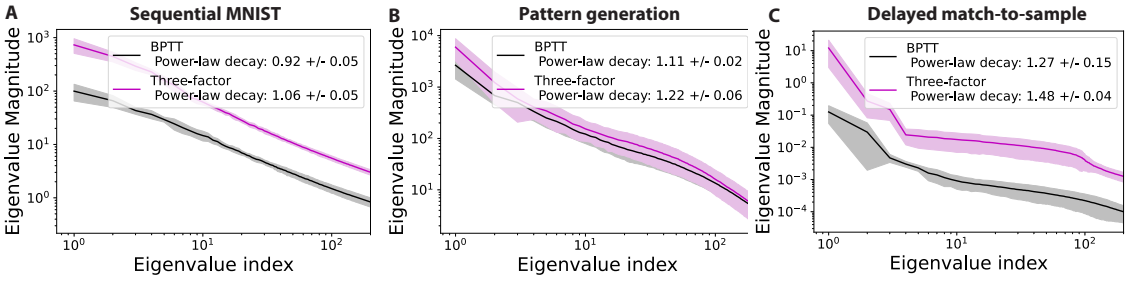

Figure S4.3: Loss' Hessian eigenspectrum for the three-factor rule exhibits significantly steeper power-law decay compared to that of BPTT. We fit a power-law function to the top 200 eigenvalues at the end of training and measure the decay parameter. Fitting to the top 50 or 100 eigenvalues resulted in similar trends. Solid lines/shaded regions: mean/standard deviation of eigenspectrum obtained at the end of training across five independent runs.

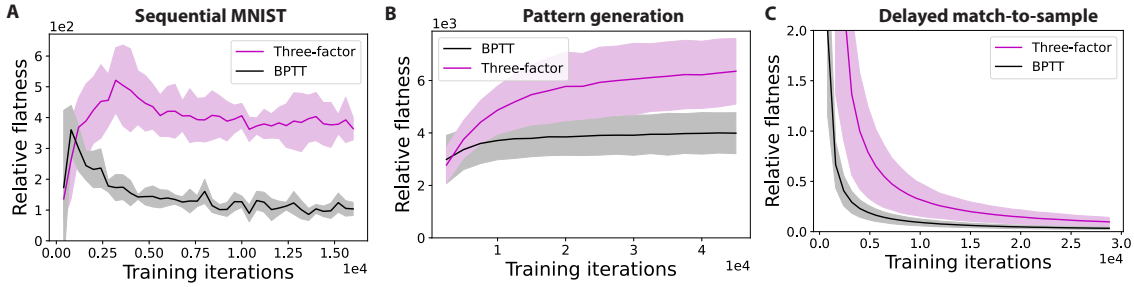

Figure S4.4: Curvature preference behavior corroborated using relative flatness measure [14]. Here, the trend is consistent with that of Figure 4.3. Note that the relative flatness measure can be computationally intensive for recurrent weights, so we computed it for readout weights. Plotting conventions follow that of previous figures.

Figure S4.7.

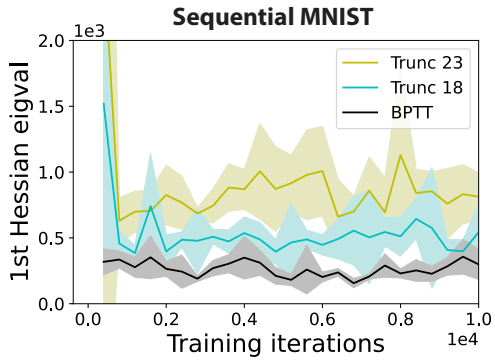

Figure S4.5: Approaching high curvature regions seems to be a shared problem for temporal truncations of the gradient. We repeat the analysis in Figure 4.3 for truncated BPTT (TBPTT). Here, "Trunc X" means X time steps are truncated during the gradient calculation. We observe that TBPTT tends to converge to high curvature regions. Plotting conventions follow that of previous figures.

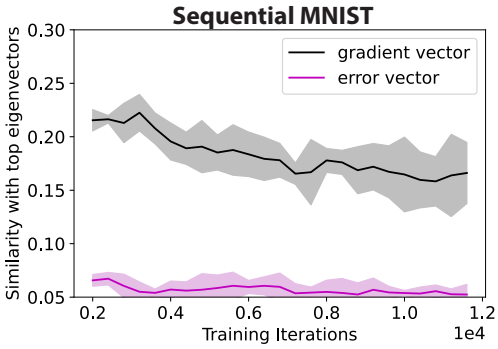

Figure S4.6: Truncation error vector (compared to the gradient) is significantly less aligned with the top Hessian eigenvector subspace. Following [15], we compute the cosine similarity between the error vector (of the three-factor rule) and a top Hessian eigenvector (averaged over the top 5 eigenvectors). The absolute value of the cosine similarity was taken. We observe weak alignment of the approximation error vector $\vec{e}$ with the leading Hessian eigenvectors. Similar trends were attained had we averaged over the leading 10 or 20 eigenvectors. Plotting conventions follow that of previous figures.

Finally, to further examine the correlation between leading Hessian eigenvalue and generalization performance (observed in Figure 4.2), we also observed such correlation correlation for runs with the learning rule fixed (Appendix Figure S4.8). For the matching step experiment in Figure 4.4, similar observations were also made when we repeated the experiment at three times the learning rate (Appendix Figure S4.8C). Moreover, we stopped BPTT early to match the test accuracy of the three-factor rule, and observed similar curvature

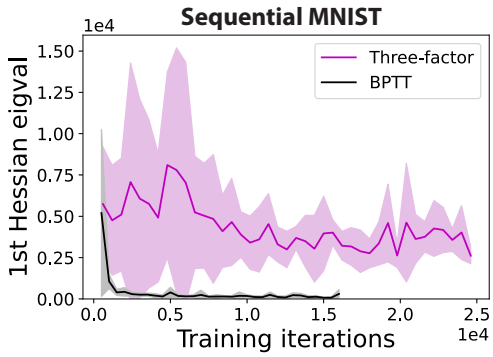

Figure S4.7: Curvature convergence behavior also holds for Adam optimizer. As explained in Methods, we used SGD optimizer due to confounding factors in Adam optimizer that could convolute our matching step length analysis in Figure 4.4. We observe similar trends as in Figure 4.3. Plotting conventions follow that of previous figures.

convergence and generalization performance trends as previously (Appendix Table S4.1).

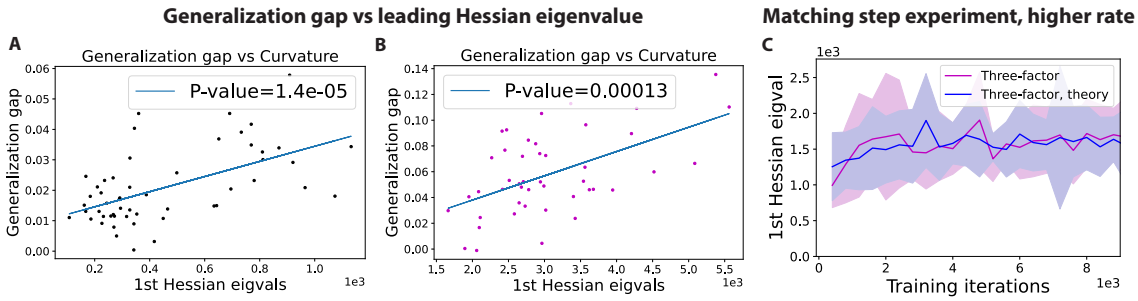

Figure S4.8: We repeat the generalization gap vs leading Hessian eigenvalue scatter plot in Figure 4.2 with the learning rule fixed for A) BPTT and B) the three-factor rule. As expected, a significant correlation between the generalization gap and leading Hessian eigenvalue is observed. Unlike Figure 2, where the hyperparameters were fixed for each rule (tuned using the procedure in Appendix S4.1.3), the learning rate is varied here in order to get a wide enough curvature range to observe the correlation. C) The matching step experiments in Figure 4.4 were repeated here with the learning rate increased by three times for all rules, and the observation agrees with that in Figure 4.4. Plotting convention follows that of previous figures.

Learning	Leading Hessian eigenvalue	Generalization gap
Three-factor	2550 ± 490	0.5 ± 0.3
BPTT, early stopping	316 ± 84	0.2 ± 0.1

Table S4.1: BPTT stopped early to match the test accuracy of the three-factor rule for the sequential MNIST task. Higher generalization gap and leading Hessian eigenvalue is again observed for the three-factor rule, as expected. Each rule is repeated for five runs with different random weight initialization.

Chapter 5

HOW CONNECTIVITY STRUCTURE SHAPES RICH AND LAZY LEARNING IN NEURAL CIRCUITS

5.1 Introduction

Structural variations can significantly impact learning dynamics in theoretical neuroscience studies of animals. For instance, studies have revealed that specific neural connectivity patterns can facilitate faster learning of certain tasks [285–291]. In deep learning, structure, encompassing architecture and initial connectivity, crucially dictates learning speed and effectiveness [61, 252, 285, 292].

A key structural aspect is the initial connectivity prior to training. Specifically, the initial connection weight magnitude can significantly bias learning dynamics, pushing them towards either rich or lazy regimes [293, 294]. Lazy learning often induces minor changes in the network during the learning process. Such minimal adjustments are advantageous given that plasticity is metabolically costly [295, 296], and significant changes in representations might lead to issues like catastrophic forgetting [297, 298]. On the other hand, the rich learning regime can significantly adapt the network's internal representations to task statistics, which can be advantageous for task feature acquisition and has implications for generalization [294, 299]. Most research on initial weight magnitude's role in learning dynamics has focused on random Gaussian or Uniform initializations [285, 294, 300]. These patterns stand in contrast to the connectivity structures observed in biological neural circuits, which could exhibit a more pronounced low-rank eigenstructure [301]. This divergence prompts a pivotal question: how does the initial weight structure, given a fixed initial weight magnitude, bias the learning regime?

This study examines how initial weight structure, particularly the effective rank, modulates the *effective* richness or laziness of task learning within the standard training regime. We note that *rich* and *lazy* learning regimes have well established meanings in deep learning theory. The

latter being defined as a situation where the Neural Tangent Kernel (NTK) stays stationary during training, while the former refers to the case where the NTK changes. In this work, we slightly extend these definitions and introduce **effective learning richness/laziness**. Unlike the traditional definition, which is based upon initialization, effective learning richness/laziness is defined in terms of post-training adjustment measurements. From this perspective, a learning process is deemed effectively "lazy" if the measured NTK movement is small. For example, consider a network whose initialization puts it in standard rich regime, but for a given task, its NTK moves very little during training. We define learning for this specific situation as effectively lazy. In other words, while the standard regime definition informs us (prior to training) whether the network can adapt significantly to task training or not, our "effective" definition lies in the post-training effects.

5.1.1 Contributions

Our main **contributions** and findings can be summarized as follows:

- Through theoretical derivation in two-layer feedforward linear network, we demonstrate that higher-rank initialization results in *effectively* lazier learning **on average** across tasks (Theorem 1). We note that the emphasis of the theorem is on the expectation across tasks.

- We validate our theoretical findings in recurrent neural networks (RNNs) through numerical experiments on well-known neuroscience tasks (Figure 5.1) and demonstrate the applicability to different initial connectivity structures extracted from neuroscience data (Figure 5.2).

- We identify scenarios where certain low-rank initial weights still result in *effectively* lazier learning for specific tasks (Proposition 2 and Figure 5.3). We postulate that such patterns emerge when a neural circuit is predisposed — perhaps due to evolutionary factors or post-development — to certain tasks, ingraining specific inductive biases in neural circuits.

5.1.2 Related works

An extended discussion on related works can also be found in Appendix S5.1.

Theoretical Foundations of Neural Network Regimes and Implications for Neural Circuits: The deep learning community has made tremendous strides in developing theoretical groundings for artificial neural networks [17, 201, 202, 302–305]. A focal point is the 'rich' and 'lazy' learning regimes dichotomy, which have distinct impacts on representation and generalization [293, 294, 299, 300, 306–311]. The 'lazy' regime results in minimal weight changes, while the 'rich' regime fosters task-specific adaptations. The transition between these is influenced by various factors, including initial weight scale and network width [293, 306].

Deep learning theories increasingly inform studies of biological neural network learning dynamics [67, 285, 312–317]. For the rich/lazy regime theory, the existence of diverse learning regimes in neural systems is evident through the resource-intensive plasticity-driven transformations prevalent in developmental phases, followed by more subdued adjustments [318], and previous investigations characterized neural network behaviors under distinct regimes [312, 319] and discerning which mode yields solutions mimicking neural data [294]. Our work extends these studies by examining how initial weight structures affect learning.

Neural circuit initialization, connectivity patterns and learning: Extensive research has explored the influence of various random initializations on deep network learning [320–324]. The literature predominantly focuses on random initialization, but actual neural structures exhibit markedly different connectivity patterns, such as Dale's law and enriched cell-type-specific connectivity motifs [325–329]. Motivated by existing evidence of low-rankedness in the brain [330] and the overrepresentation of local motifs in neural circuits [301], which could be indicative of low-rank structures due to their influence on the eigenspectrum [328, 331], our study explores the impact of connectivity effective rank on learning regimes. This focus is driven by the plausible presence of such low-rank structures in the brain, potentially revealed through these local motifs. With emerging connectivity data [332–336], future work is poised to encompass rich additional features of connectivity.

5.2 Setup and Theoretical Findings

5.2.1 RNN setup

We examine recurrent neural networks (RNNs) because they are commonly adopted for modeling neural circuits [337, 338]. We consider a RNN with N_{in} input units, N hidden units and N_{out} readout units (Figure 5.1A). The update formula for $h_t \in \mathbb{R}^N$ (the hidden state at time t) is governed by [16, 124]:

$$h_{t+1} = \rho h_t + (1-\rho)(W_h f(h_t) + W_x x_t), \tag{5.1}$$

where an exponential Euler approximation is made with $\rho = e^{-dt/\tau_m} \in \mathbb{R}$ denoting the leak factor for simulation time step dt and τ_m denoting the membrane time constant; $f(\cdot) : \mathbb{R}^N \to \mathbb{R}^N$ is the activation function, for which we use $ReLU$; $W_h \in \mathbb{R}^{N\times N}$ (resp. $W_x \in \mathbb{R}^{N\times N_{in}}$) is the recurrent (resp. input) weight matrix and $x_t \in \mathbb{R}^{N_{in}}$ is the input at time step t. Readout $\hat{y}_t \in \mathbb{R}^{N_{out}}$, with readout weights $w \in \mathbb{R}^{N_{out}\times N}$, is defined as

$$\hat{y}_t = \langle w, f(h_t)\rangle. \tag{5.2}$$

The objective is to minimize scalar loss $L \in \mathbb{R}$, for which we use the cross-entropy loss for classification tasks and mean squared error for regression tasks. L is minimized by updating the parameters using variants of gradient descent:

$$\Delta W = -\eta \nabla_W L, \tag{5.3}$$

for learning rate $\eta \in \mathbb{R}$ and $W = [W_h \quad W_x \quad w^T] \in \mathbb{R}^{N\times(N_{in}+N+N_{out})}$ contains all the trainable parameters. Details of parameter settings can be found in Appendix S5.3.

5.2.2 Effective laziness measures

As mentioned above, we introduce *effective* richness and laziness, with effectively lazier (resp. richer) learning corresponding to less (resp. greater) network change over the course of learning. To quantify network change, we adopt the following three measures that have been previously [299]. We note that these measures can be sensitive to other architectural aspects

that bias learning regimes, such as network width, so throughout we hold these variables constant when making the comparisons.

Weight change norm quantifies the vector norm of change in W. Effectively lazier learning should result in a lower weight change norm, and it is quantified as:

$$\|\Delta W\| := \|W^{(f)} - W^{(0)}\|, \tag{5.4}$$

where $\|\cdot\| = \|\cdot\|_F$; $W^{(0)}$ (resp. $W^{(f)}$) are the weights before (resp. after) training.

Representation alignment (RA) quantifies the directional change in a representational similarity matrix (RSM) before and after training. RSM focuses on the similarity between how two pairs of input are represented by computing the Gram matrix R of last step hidden activity. Greater representation alignment indicates effectively lazier learning in the network, and it is obtained by

$$RA(R^{(f)}, R^{(0)}) := \frac{Tr(R^{(f)}R^{(0)})}{\|R^{(f)}\|\|R^{(0)}\|}, \quad \text{where } R := H^T H, \tag{5.5}$$

where $H \in \mathbb{R}^{N\times m}$ is the hidden activity at the last time step; $R^{(0)}$ and $R^{(f)} \in \mathbb{R}^{m\times m}$ are the initial and final RSM, respectively; m is the batch size.

Tangent kernel alignment (KA) quantifies the directional change in the neural tangent kernel (NTK) before and after training; effectively lazier learning should result in higher tangent kernel alignment. The NTK computes the Gram matrix K of the output gradient. Greater tangent kernel alignment points to effectively lazier learning, and it is obtained by

$$KA(K^{(f)}, K^{(0)}) := \frac{Tr(K^{(f)}K^{(0)})}{\|K^{(f)}\|\|K^{(0)}\|}, \quad \text{where } K := \nabla_W \hat{y}^T \nabla_W \hat{y} \tag{5.6}$$

where $K^{(0)}$ and $K^{(f)} \in \mathbb{R}^{m\times m}$ (for the $N_{out} = 1$ case) denote the initial and final NTK, respectively.

5.2.3 Theoretical findings

This subsection derives the theoretical impact of initial weight effective rank on tangent kernel alignment. First, Theorem 1 focuses on **task-agnostic** settings, treating task definition as random variables and computing the **expected** tangent kernel alignment across tasks. With

some assumptions, tangent kernel alignment is maximized when the initial weight singular values are distributed across all dimensions (i.e. high-rank initialization).

In this section, our theoretical results are framed in a simplified feedforward setting, as we use a two-layer network with linear activations. However, we return to RNNs (Eq. 5.1) for the rest of the paper, and verify the generality of our theoretical findings with numerical experiments for both feedforward and recurrent architectures. Our choice is motivated by the need for theoretical tractability. While research on RNN learning in the NTK regime exists [302, 305, 339], we are not aware of any studies featuring the final converged NTK that could serve as a basis for our comparison of the initial and final kernel. Consequently, we have chosen to focus on RNNs for neural circuit modeling and employ linear feedforward networks for theoretical derivations, a strategy also adopted by [315]; numerous other studies, including references [314], [17], [324], and [285], have similarly concentrated on extracting theoretical insights from linear feedforward networks.

For a two-layer linear network with input data $X \in \mathbb{R}^{d\times m}$, $W_1 \in \mathbb{R}^{N\times d}$ and $W_2 \in \mathbb{R}^{1\times N}$ as weights for layers 1 and 2, respectively, the NTK throughout training, K, is:

$$K = X^T(W_1^T W_1 + \|W_2\|^2 I)X. \tag{5.7}$$

Without the loss of generality, suppose the output target $Y \in \mathbb{R}^{1\times m}$ is generated from a linear teacher network as $Y = \beta^T X$, for some Gaussian vector $\beta \in \mathbb{R}^d$, with $\beta_i \sim \mathcal{N}(0, 1/d)$.

Theorem 1. *(Informal) Consider the network above with its corresponding NTK in Eq. 5.7, trained under MSE loss with small initialization and whitened data. The expected kernel alignment across tasks is maximized with high-rank initialization, i.e. the singular values of* $W_1^{(0)}$ *are distributed across all dimensions. (Formal statement and proof are in Appendix S5.2)*

The intuition of Theorem 1 result is that, when two random vectors are drawn in high-dimensional spaces, corresponding to the low-rank initial network and the task, the probability of them being nearly orthogonal is very high; this then necessitates greater movement to eventually learn the task direction. We emphasize again that Theorem 1 is **task-agnostic**, i.e. it focuses on the **expected** tangent kernel alignment across input-output definitions. This is in contrast to **task-specific** settings (e.g. [300]) that focus on a given task. In such

task-specific settings, certain low-rank initializations can in fact lead to lazier learning. The following proposition predicts that if the task structure is known, low-rank initialization that is already aligned with the task statistics (input/output covariance) can lead to kernel alignment. We revisit this proposition again in Figure 5.3. We remark that initializing this way can still have high initial error because of randomized $W_2^{(0)}$.

Proposition 2. *(Informal) Following the setup and assumptions in Theorem 1, rank-1 initializations of the form* $W_1^{(0)} = \sigma[\beta^T/\|\beta\| \quad \vec{0} \quad ... \quad \vec{0}]$ *leads to a high tangent kernel alignment. (Formal statement and proof are in Appendix S5.2)*

Above, we state technical results in terms of one metric of the effective laziness of learning — based on the NTK; our proof in Appendix S5.2 easily extends also to the representation alignment metric. The impact on weight change is also assessed in Appendix Proposition 3. This is in line with our simulations with RNNs, which will show similar trends for all three of the metrics introduced in Section 5.2.2).

5.3 Simulation results

In this section we empirically illustrate and verify our main theoretical results, which are: **(1)** on average, high-rank initialization leads to effectively lazier learning (Theorem 1); **(2)** it is still possible for certain low-rank initializations that are already aligned to the task statistics to achieve effectively lazier learning (Proposition 2).

Impact on effective laziness by low-rank initialization via SVD in RNNs: As a proof-of-concept, we start in Figure 5.1 with low-rank initialization in RNNs by truncating an initial Gaussian random matrix via Singular Value Decomposition (SVD), which enables us to precisely control the rank, and rescale it to ensure that the comparison is across the same weight magnitude [174]. Additionally, all comparisons were made after training was completed, and all these training sessions achieved comparable losses. For our investigations, we applied this initialization scheme across a variety of cognitive tasks — including two-alternative forced choice (2AF), delayed-match-to-sample (DMS), context-dependent decision-making (CXT) tasks — implemented with Neurogym [16] and the well-known machine learning benchmark sequential MNIST (sMNIST). Figure 5.1 indicates that low-rank initial weights

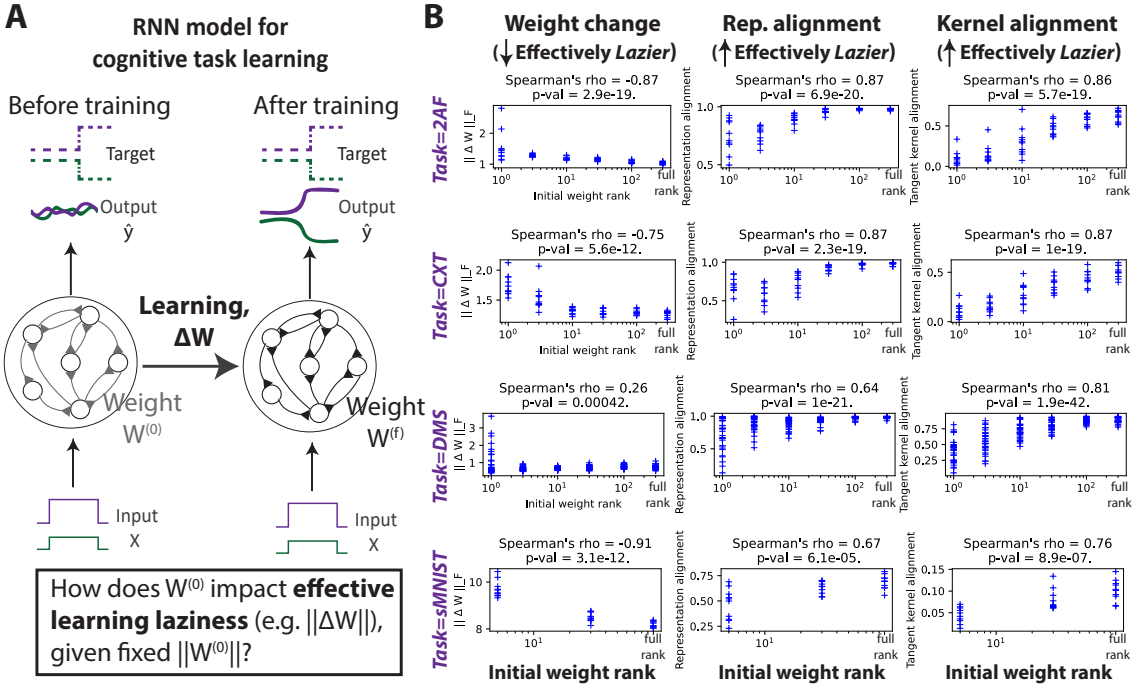

Figure 5.1: Low-rank initial recurrent weights, generated using SVD, lead to greater changes (or effectively richer learning) in the recurrent neural network. A) Schematic of RNN training setup. B) Measurements of effective richness vs laziness of learning (metrics as defined in Section 5.2.2), for RNN trained on several cognitive tasks in Neurogym [16] as well as the sequential MNIST task (sMNIST). For details on SVD weight creation, see Appendix S5.3. Fewer rank points were used for sMNIST due to computational time. Each dot represents a single training run, with each run using a different random initialization (10 runs total for each setting).

result in effectively richer learning and greater network changes.

These numerical trends are in line with Theorem 1, which focused on an idealized setting of a two-layer linear network with numerical results in Appendix Figure S5.1A. We also demonstrated this trend for a non-idealized feedforward setting in Appendix Figure S5.1B, and more explorations in feedforward settings and across a broader range of architecture is left for future exploration due to our focus on RNNs. In the Appendix, we show the main trends observed in Figure 5.1 also hold for Uniform initialization (Figure S5.4), soft initial weight rank (Figure S5.5), various network sizes (Figure S5.6), learning rates (Figure S5.7), gains (Figure S5.8), and finer time step dt (Figure S5.9). We note that, in addition to fixing the weight magnitude across comparisons, the dynamical regime might also confound learning regimes [340]. A common method for controlling the dynamical regime is through

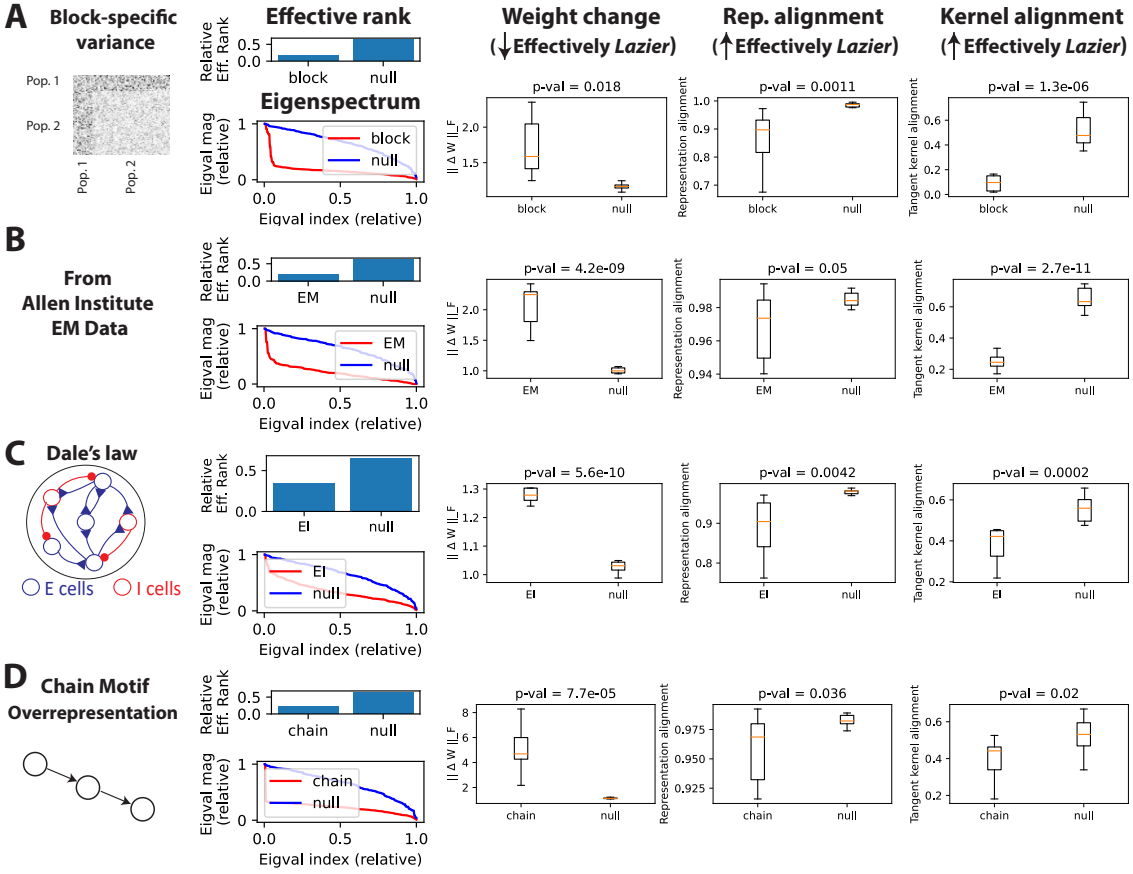

Figure 5.2: Low-rank initial weight structures, inspired by biological examples, lead to effectively richer learning. We present the eigenspectrum and the relative effective rank of connectivity in A) structures with cell-type-specific statistics, B) structures derived from EM data, C) structures obeying Dale's law, and D) structures with an over-representation of chain motifs; we also present the effective learning laziness for networks initialized with these connectivity structures. These structures exhibit a lower effective rank compared to standard random Gaussian initialization (null). We plotted the magnitude of the eigenvalues (Eigval mag) — scaled by the dominant eigenvalue's magnitude — against their indices normalized by the network size N (Eigval index). We apply the effective laziness measures described in Section 5.2.2 to compare the effective laziness of experimentally-driven initial connectivity versus standard random Gaussian initialization (null). See Appendix S5.3 for details on network initialization. The boxplots are generated from 10 independent runs with different initialization seeds. Due to space constraints, we include only the 2AF task here, but Appendix Figures S5.2 and S5.3 show that similar trends hold for the DMS and CXT tasks.

the leading weight eigenvalue, which affects the top Lyapunov exponent. Controlling in this manner led to similar trends (Appendix Figure S5.10). Investigating the relationship between learning regimes and various concepts of dynamical regimes further is a promising direction for future work. Moreover, since our emphasis is on the effective learning regime, which is based on post-training changes, we concentrated on the laziness measures computed from

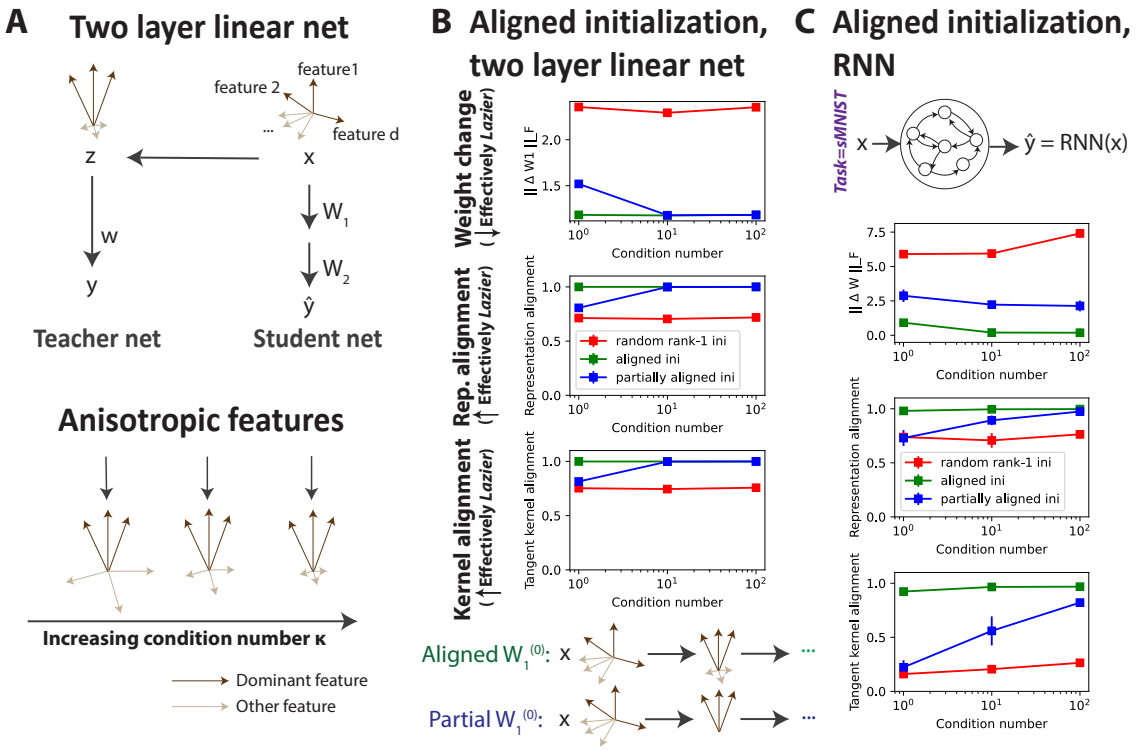

Figure 5.3: Low-rank initializations can still achieve high alignment for specific tasks (see Proposition 2). A) The student-teacher two-layer linear network setup as described in Section 5.2.3, but with feature anisotropy controlled by a feature modulation matrix F, i.e. $z = Fx$. The condition number of F dictates the relative feature strength. We set the top half of the singular values of F are set to κ, while the bottom half are set to 1, where κ represents the condition number of F. B) The aligned initialization (green) is achieved by setting W_1 as described in Proposition 2 (with $\beta = w^T F$, w is as illustrated), so that the initialization aligns with the task statistics. The partial alignment (blue) mirrors the aligned case, but F is substituted with its rank-$(d/2)$ truncation, causing the network to align only with the dominant features. **We observe that a considerably higher alignment can be achieved when the initialization aligns solely with the dominant features, especially when the relative strength of these dominant features is high.** C) The analysis from B) is replicated for RNNs learning the sMNIST task. As the ground truth network function is elusive, we use a teacher network with pre-trained weights. Once again, we replace F with its rank-$(d/2)$ truncation for partial alignment. Details on the input/output definitions and initializations, as well as other simulation specifics, are available in Appendix S5.3. We note that in all scenarios presented here, the initial errors are high since the readout weights are initialized randomly, rendering it a valid learning problem.

networks after training, rather than during the learning process. However, we also tracked the alignment with the initial kernel and task kernel alignment during training (Appendix Figure S5.11). We also examined how the kernel's effective rank evolves throughout the training period (Appendix Figure S5.12).

Low-rank initialization via biologically motivated connectivity in RNNs: To establish a closer connection with biological neural circuits, we have tested our predictions

on low-rank initialization using a variety of biologically motivated structures capable of resulting in low-rank connectivity. Here are some of the examples: (A) connectivity with cell-type-specific statistics [329], where each block in the weight matrix corresponds to the connections between neurons of two distinct cell types, with the variance of these connections differing from one block to another. In terms of block-specific connectivity statistics, there are infinite possibilities for defining the blocks, each resulting in a unique eigenspectrum. For the example provided here, we adopted the setup from Figure S3 in [329], with parameters set as $\alpha = 0.02$, $\gamma = 10$, and $1 - \epsilon = 0.8$; these correspond to the fraction of hyperexcitable neurons, gain of hyperexcitable connections and gain of the rest, respectively. We follow this particular setup because it has been demonstrated to create an outlier leading eigenvalue, thereby reducing the effective rank. We also consider (B) connectivity matrix derived from the electron microscopy (EM) data [341], where the synaptic connections between individual neurons are meticulously mapped to create a detailed and comprehensive representation of neural circuits. Also, we consider (C) connectivity obeying Dale's law, where each neuron is either excitatory or inhibitory, meaning it can only send out one type of signal – either increasing or decreasing the activity of connected neurons – a principle inspired by the way neurons behave in biological systems [301]. Additionally, (D) the over-representation of certain localized connectivity patterns (or network motifs) — such as the chain motif, where two cells are connected via a third intermediary cell — creates outliers in the weight eigenspectrum, subsequently lowering the effective rank [132, 328, 342]. Details of these initial connectivity structures are provided in Appendix S5.3.

As illustrated in Figure 5.2, these connectivity structures, motivated by known features of biological neural networks, exhibit a lower effective rank compared to standard random Gaussian initialization, thereby serving as natural testbeds for our theoretical predictions. To quantify (relative) effective rank, we used $(\sum_i |\lambda_i|)/(|\lambda_1|N)$, which indicates the fraction of eigenvalues on the order of the dominant one and captures the (scaled) area under the curve of the eigenspectrum plots. We also tried effective rank based on singular values, i.e. $(\sum_i |s_i|)/(|s_1|N)$, in Appendix Figure S5.13 and observed similar trends. Importantly, Figure 5.2 show that these different low-rank biologically motivated structures can lead to effectively richer learning compared to the standard random Gaussian initialization. This

finding supports our overarching prediction, that lower rank initial weights leads to effectively richer learning. We note that to test our theoretical predictions based on gradient-descent learning without specific constraints on the solutions, the structures are enforced only at initialization and not constrained during training. In Appendix Figure S5.14, we also constrained Dale's Law throughout training and found similar trends.

Low-rank initialization aligned with task statistics: These simulations may be considered to be within our task-agnostic framework. That is, we have chosen a "random" battery of tasks that is not directly matched to the initial network connectivity structures. Thus, our findings that lower rank initializations lead to richer learning are expected from our theoretical prediction on the task-averaged alignment (Theorem 1), rather than something task-specific. However, Proposition 2 also predicts that low-rank initialization can lead to lazy learning if the initialization is already aligned to the task structure. To test this, we observe in Figure 5.3 that a considerably higher alignment can be achieved when the initialization aligns solely with the dominant task features, especially when the relative strength of these dominant features is high. We postulate that such alignment may occur in biological settings if the circuit has evolved to preferentially learn specific tasks.

5.4 Discussion

Our investigation casts light on the nuanced influence of initial weight effective rank on learning dynamics. Anchored by Theorem 1, our theoretical findings underscore that high-rank random initialization generally facilitates *effectively* lazier learning on average across tasks. This focus on the expectation across tasks can provide insights into the circuit's flexibility in learning across a broad range of tasks as well as predict the effective learning regime when the task structure is uncertain. However, certain low-rank initial weights, when naturally predisposed to specific tasks, may lead to effectively lazier learning, suggesting an interesting interplay between evolutionary or developmental biases and learning dynamics (Proposition 2). Our numerical experiments on RNNs further validate these theoretical findings illustrating the impact of initial rank in diverse settings.

Potential implications to neuroscience: We investigate the impact of effective weight rank on learning regimes due to its relevance in neuroscience. Learning regimes reflect the

extent of change through learning, implicating metabolic costs and catastrophic forgetting [295–297]. The presence of different learning regimes is demonstrated in neural systems, since during developmental phases where neural circuits undergo extensive, plasticity-driven transformations. In contrast, mature neural circuits exhibit more subtle synaptic adjustments [318]. We hypothesize that a circuit's task-specific alignment might be established either evolutionarily or during early development. The specialization of neural circuits, such as ventral versus dorsal [343], may arise from engaging in tasks with similar computational demands. Conversely, circuits with high-rank structures may be less specialized, handling a wider array of tasks. Our framework could be used to compare connectivities across brain regions and species in order to predict their function and flexibility, assessing their functional specialization based on effective connectivity rank. Additionally, our framework predicts that connectivity rank will affect the degree of change in neural activity during the learning of new tasks. This hypothesis could be tested through BCI experiments, as shown in [344] and [345], to explore how learning dynamics vary with connectivity rank.

Regarding deep learning, low-rank initialization is not a common practice, yet adaptations like low-rank updates have gained popularity in training large models [346]. LoRA, the study cited, concentrates on parameter updates rather than initializations, but understanding how update rank affects learning regimes is crucial. Our results offer a starting point for further exploration in this area. Although different rank initializations are less explored, with some exceptions like [347], our findings suggest that this area should receive more attention due to its potential effects on learning regimes and, consequently, on generalization [299].

Limitations and future directions: Our study predominantly focused on the weight (effective) rank, leaving the exploration of other facets of weight on the effective learning regime as an open avenue. Also, the ramifications of effective learning regimes on learning speed — given the known results on kernel alignment and ease of learning [348] and present mixed findings in the existing literature [294, 299] — warrant further exploration.

Expanding the scope of our study calls for examining a wider variety of tasks, neural network architectures, and learning rules. Although our work is based on the backpropagation learning rule, its implications for biologically plausible learning rules remain unexplored. Our primary criterion for selecting tasks was their relevance to neuroscience, aligning with our

main objectives. However, given the diverse range of tasks performed by various species, future research could benefit from exploring a more extensive array of tasks. Exploring more complex neuroscience tasks, such as those in Mod-Cog [349], could provide valuable insights. On that note, we tested the pattern generation task from [19], a neuroscience task differing in structure from the Neurogym tasks, and observed similar trends (refer to Appendix Figure S5.15).

Additionally, we ensured the consistency of outcomes against factors like width, learning rate, and initial gain (see Appendix S5.4), but other factors such as dynamical regime and noise [310] remain underexamined. On that note, the study's focus on RNNs with finite task duration prompts further investigation into the implications for tasks with extended time steps and how conclusions for feedforward network depth [350, 351] translate to RNN sequence length. Examining several mechanisms at once is beyond the scope of one paper, but our theoretical work constitutes the foundation for future investigations.

Moreover, it is crucial to further explore the neuroscientific implications of effective learning regimes, as well as their diverse impacts on aspects such as representation, including kernel-task alignment (see Appendix Figure S5.11), and generalization capabilities [294, 299, 319]. Our current study did not delve into how initial weight rank affects these facets of learning, representing an essential future direction in connecting weight rank to these theoretical implications in both biological and artificial neural networks.

Furthermore, while there exists evidence for low-rankedness in the brain [330], the extent to which the brain uses low-rank structures remains an open question, especially as neural circuit structures can vary across regions and species. While local connectivity statistics [301] can offer some predictive insight into the global low-rank structure, this relationship is not always immediately apparent [331]. Our theoretical results contribute to understanding the role of connectivity rank in the brain by linking effective connectivity rank with learning dynamics.

Lastly, we have primarily examined low-rank tasks and there remains unexplored terrain regarding the interplay between the number of task classes and weight rank, which is pivotal to uncovering a more precise relationship between the effective learning regime and the initial weight rank [352, 353]. Overall, this dynamic area of learning regimes is ripe for many

explorations, integrating numerous factors; our work contributes to this exciting area with new tools.

S5.1 Extended discussions on related works

Theoretical Foundations of Neural Network Regimes and Implications for Neural Circuits: The journey of understanding deep learning systems has borne witness to unprecedented progress in the mathematical dissection of neural network functionalities [17, 18, 201, 202, 204, 302–305, 339]. These theoretical findings, until recently confined predominantly to artificial domains, have embarked upon explorations into biological neural networks, elucidating the intricate dynamics of learning and computational properties [67, 285, 312, 313]. Among the vanguard of these theoretical endeavors stands the dichotomy of 'rich' and 'lazy' learning regimes. Both lead to task learning, yet they carry distinct ramifications for representation and generalization [293, 294, 299, 300, 306–311]. In the 'lazy' regime, which is typically associated with large initial weights, learning predominantly centers on adjusting the readout weights. This leads to minimal alterations in the network weights and representation, while capitalizing on the expansive dimensionality provided by the hidden layer's random projections [294]. In contrast, the 'rich' regime, defined by smaller initial weights, fosters the development of highly tailored hidden unit representations specifically aligned with task demands, resulting in considerable adaptations in weights and representation. It's essential to highlight that the transition and dominance between these regimes are influenced by more than just the initial weight scale. Other factors, ranging from network width to the output gain (often referred to as the α parameter), play a pivotal role [293, 306].

A nexus between deep learning theoretical frameworks and neuroscience has unveiled applications of the rich/lazy regimes. Previous investigations characterized neural network behaviors under distinct regimes [312, 319] and discerning which mode yields solutions mimicking empirical data [294]. It is compelling to observe that the existence of multiple learning regimes isn't an isolated phenomenon in artificial systems; analogous learning patterns echo in neural circuits as well. For instance, while plasticity-driven transformations might be resource-intensive, they manifest robustly during such developmental phases, followed by

minor changes afterwards [318]. Building upon these findings, our research delves deeper into the precursors of these regimes. We examine how inherent factors in the brain, especially initial weight configurations, influence the inclination towards either rich or lazy learning. This understanding is crucial for assessing the applicability of regime-specific tools in neural contexts and for shedding light on the potential benefits of having both learning regimes coexist in the brain.

Interplay of Neural Learning and structure: Understanding how the brain learns using its myriad elements is a perennial quest in neuroscience. Addressing this, certain studies have unveiled biologically plausible learning rules [19, 56, 59, 65, 66, 81, 140, 354–361], suggesting potential neural algorithms involving known neural ingredients. Concurrently, given the three primary components of a neural network's design — task, learning rule, and architecture — another avenue of research delves deep into the architectural facet, specifically focusing on how it interacts with the learning rule to enhance learning [61, 252, 292]. Under the structural umbrella, the neural unit's complexity and initial connectivity patterns are two crucial aspects. Complex neuron models, for instance, have shown the potential in boosting learning performance by allowing implicit forms of memory and computations at the single neuron level [362, 363]. Moreover, A large body of work has investigated the effect of different random initializations on learning in deep networks [320–324]. For instance, the variance in random initial weights can induce pronounced shifts in network behavior, ranging from the "lazy" to the "rich" regimes [293, 294]. This introduces unique inductive biases during the learning process, with distinct preferences for learning certain features [299]. Our discourse primarily orbits around connectivity and its implications on learning dynamics in networks with simple rectified units. Our results sit within the purview of these regimes, with a widely adopted assumption of gradient descent via backpropagation as the learning rule, while remaining open to encompassing a wider spectrum of rules in future explorations.

Neural circuit connectivity pattern and eigenspectrum: While the importance of initial weights on function and learning is clear, the impact of specific weight shapes, apart from weight scale, on rich or lazy dynamics remains less explored. The predominant focus in the literature has been on random initialization. Yet, neural circuit structures significantly diverge from this paradigm. Illustratively, one finds connectivity principles

or patterns markedly different from what one observes with a mere random initialization [364], resulting in distinct neural dynamics; these connectivity principles or patterns include Dale's law [325–327], an over-representation of higher-order motifs [328] and cell-type-specific connectivity statistics [329], to name a few. Given the prominence of low-rankedness observed in neural circuits [301], our study centers on the influence of effective rank on the effective learning regime. As the next generation of connectivity data becomes available [332–336], future explorations will broaden the scope to other weight characteristics.

S5.2 Proofs

S5.2.1 Proofs for main text Theorem and Proposition

Notation Let $f(x) = W_2 W_1 x$ denote a two-layer linear network with N hidden units on d-dimensional inputs $x \in \mathbb{R}^d$, with weight matrices $W_1 \in \mathbb{R}^{N \times d}$ and $W_2 \in \mathbb{R}^{1 \times N}$. We consider m training inputs $x_1, \cdots x_m$ and the corresponding data matrix $X = [x_1^T \cdots x_m^T] \in \mathbb{R}^{d \times m}$; the output target is generated from a linear teacher network as $Y = \beta^T X$, where $\beta_i \sim \mathcal{N}(0, 1/d)$.

Since our goal is to investigate how the *shape* of the initial weights impacts network change, we will consider a fixed small (Froebenius) norm for these; i.e.,

$$\|W_1^{(0)}\|_F = \|W_2^{(0)}\|_F := \sigma \ll 1$$

We denote by $s_1, \cdots, s_d$ denote the singular values of $W_1^{(0)}$; they satisfy $\sum_{j=1}^d s_j^2 = \sigma^2$.

In what follows we focus on the whitened setting, where X has all its non zero singular values equal to 1. We also assume $m \geq d$ for simplicity (this assumption can easily be relaxed in our analysis), so that the whitened data assumption translates as $XX^T = I_d$.

Prior results Our analysis builds on prior results [17] on the evolution of the NTK for two-layer linear networks trained by gradient flow of the mean square error. In the above setting, [17] show that the final NTK $K^{(f)}$ (i.e. the asymptotic NTK as the number of iterations goes to infinity) is given by

$$K^{(f)} = ||\beta|| X^T (\hat{\beta}\hat{\beta}^T + I_d) X + O(\sigma^2). \tag{S5.1}$$

where $\hat{\beta} := \beta / \|\beta\|$. We are interested in the expected kernel alignment over the tasks, in the small initialization regime:

$$\mathbb{E}_\beta[KA(K^{(f)}, K^{(0)})] := \mathbb{E}_\beta \left[\frac{\mathrm{Tr}(K^{(f)} K^{(0)})}{\|K^{(f)}\|_F \|K^{(0)}\|_F} \right]. \tag{S5.2}$$

Theorem 1. *In the above setting, when considering all possible initializations $W_1^{(0)}$ with small fixed norm σ, the expected kernel alignment $\mathbb{E}_\beta[KA]$ (defined in Eq. S5.2) is maximized with high-rank isotropic initialization, i.e with $W_1^{(0)}$ that has all its non-zero singular values equal in absolute value.*

Proof. Let us write $K^{(0)} = X^T M_0 X$ with $M_0 := W_1^{(0)T} W_1^{(0)} + \sigma^2 I_d$. Up to $O(\sigma^4)$ terms, the numerator in Eq. S5.2 takes the form

$$\begin{aligned}
\operatorname{Tr}(K^{(f)} K^{(0)}) &= \|\beta\| \operatorname{Tr}(X^T(\hat{\beta}\hat{\beta}^T + I_d) X X^T M_0 X) \\
&\overset{(a)}{=} \|\beta\| \operatorname{Tr}(X^T(\hat{\beta}\hat{\beta}^T + I_d) M_0 X) \\
&\overset{(b)}{=} \|\beta\| \operatorname{Tr}((\hat{\beta}\hat{\beta}^T + I_d) M_0 X X^T) \\
&\overset{(a)}{=} \|\beta\| \operatorname{Tr}((\hat{\beta}\hat{\beta}^T + I_d) M_0) \\
&\overset{(c)}{=} \|\beta\| (\hat{\beta}^T M_0 \hat{\beta} + \operatorname{Tr} M_0)
\end{aligned} \tag{S5.3}$$

where (a) uses $XX^T = I_d$, (b) the cyclicity of the trace, and (c) the fact that $\hat{\beta}^T M_0 \hat{\beta}$ is a scalar.

As for the denominator in Eq. S5.2), we have,

$$\begin{aligned}
\|K^{(0)}\|_F^2 &= \operatorname{Tr}(K^{(0)} K^{(0)}) \\
&= \operatorname{Tr}(X^T M_0 X X^T M_0 X) \\
&\overset{(a)}{=} \operatorname{Tr}(M_0^2)
\end{aligned} \tag{S5.4}$$

and, up to $O(\sigma^4)$ terms,

$$\begin{aligned}
\|K^{(f)}\|_F^2 &= \operatorname{Tr}(K^{(f)} K^{(f)}) \\
&= \|\beta\|^2 \operatorname{Tr}(X^T(\hat{\beta}\hat{\beta}^T + I_d) X X^T (\hat{\beta}\hat{\beta}^T + I_d)) \\
&= \|\beta\|^2 \operatorname{Tr}(X^T(\hat{\beta}\hat{\beta}^T + I_d) X) \\
&\overset{(a)}{=} \|\beta\|^2 \operatorname{Tr}(\hat{\beta}\hat{\beta}^T + I_d)^2 \\
&\overset{(b)}{=} \|\beta\|^2 (d+3)
\end{aligned} \tag{S5.5}$$

where (a) in these two calculations uses $XX^T = I_d$ and the cyclicity of the trace; and (b) notes that the $d \times d$ matrix $\hat{\beta}\hat{\beta}^T + I_d$ has $d-1$ eigenvalues equal to 1 and one equal to 2. Eq. S5.4 and S5.5 yield

$$\|K^{(f)}\|_F \|K^{(0)}\|_F = \|\beta\| \sqrt{(d+3) \operatorname{Tr} M_0^2} \tag{S5.6}$$

Putting together Eq. S5.3, S5.6 , we obtain, up to additive $O(\sigma^2)$ terms,

$$\mathrm{KA}(K^{(f)}, K^{(0)}) = \frac{\hat{\beta}^T M_0 \hat{\beta} + \operatorname{Tr} M_0}{\sqrt{(d+3)\operatorname{Tr} M_0^2}} \tag{S5.7}$$

Next, averaging over the tasks requires computing the Gaussian average

$$A[M_0] := \mathbb{E}_\beta \left[\hat{\beta}^T M_0 \hat{\beta}\right] = \mathbb{E}_\beta \left[\frac{\beta^T M_0 \beta}{\|\beta\|^2}\right].$$

Lemma 1. *The map A is invariant under the action of the orthogonal group, i.e $A[UMU^T] = A[M]$ for all $M \in \mathbb{R}^{d\times d}$ and all orthogonal matrices $U \in \mathbb{R}^{d\times d}$.*

Proof. This is a consequence of the invariance of the Gaussian measure under the action of the orthogonal group. Explicitly, given an orthogonal matrix U,

$$\begin{aligned} A[UMU^T] &= \frac{1}{(2\pi d)^{d/2}} \int \mathrm{d}^d\beta e^{-\|\beta\|^2/d} \left[\frac{\beta^T U M U^\top \beta}{\|\beta\|^2}\right] \\ &\overset{\beta' := U^T\beta}{=} \frac{1}{(2\pi d)^{d/2}} \int \mathrm{d}^d\beta' |\det U| e^{-\|U\beta'\|^2/d} \left[\frac{\beta'^T M \beta'}{\|U\beta'\|^2}\right] \\ &= \frac{1}{(2\pi d)^{d/2}} \int \mathrm{d}^d\beta' e^{-\|\beta'\|^2/d} \left[\frac{\beta'^T M \beta'}{\|\beta'\|^2}\right] \\ &= A[M] \end{aligned} \tag{S5.8}$$

where the third equality follows from $|\det U| = 1$ and $\|U\beta\| = \|\beta\|$. □

Lemma 2. *There is a constant c such that $A[M] = c\mathrm{Tr}(M)$ for any symmetric matrix M.*

Proof. Given a symmetric matrix M, it can be diagonalized as $M = UDU^T$ where $D = \mathrm{Diag}(\mu_1, \cdots \mu_d)$ is diagonal and U is orthogonal. By rotation invariance from Lemma 1, we have $A[M] = A[D]$, and

$$A[D] = \mathbb{E}_\beta \left[\hat{\beta}^T D \hat{\beta}\right] = \mathbb{E}_\beta \left[\sum_{j=1}^d \hat{\beta}_j^2 \mu_j\right] = \sum_{j=1}^d \mathbb{E}_\beta \left[\frac{\beta_j^2}{\|\beta\|^2}\right] \mu_j := \sum_{j=1}^d c_j \mu_j \tag{S5.9}$$

We conclude by noting that, by invariance of the (isotropic) Gaussian measure under permutation of the vector components, the coefficients c_j are independent of j, i.e $c_j \equiv c$ for all j. In sum,

$$A[M] = A[D] = c \operatorname{Tr} D = c \operatorname{Tr} M. \tag{S5.10}$$

□

The expected kernel alignment thus takes the form,

$$\mathbb{E}_\beta[\mathrm{KA}(K^{(f)}, K^{(0)})] = \frac{(1+c)\,\mathrm{Tr}\,M_0}{\sqrt{(d+3)\,\mathrm{Tr}\,M_0^2}} \tag{S5.11}$$

up to additive $O(\sigma^2)$ terms. Finally, we note that

$$\begin{aligned}\mathrm{Tr}\,M_0 &= \mathrm{Tr}(W_1^{(0)T}W_1^{(0)} + \sigma^2 I_d) \\ &= \|W_1^{(0)}\|_F^2 + d\sigma^2 \\ &= (d+1)\sigma^2\end{aligned} \tag{S5.12}$$

and

$$\begin{aligned}\mathrm{Tr}\,M_0^2 &= \mathrm{Tr}(W_1^{(0)T}W_1^{(0)} + \sigma I_d)^2 \\ &= \sum_{j=1}^{d}(s_j^2+\sigma^2)^2 \\ &= \sum_{j=1}^{d} s_j^4 + 2\sigma^2\sum_{j=1}^{d} s_j^2 + d\sigma^4 \\ &= \sum_{j=1}^{d} s_j^4 + (d+2)\sigma^4\end{aligned} \tag{S5.13}$$

Substituting into Eq. S5.11, we have, up to additive $O(\sigma^2)$ terms,

$$\mathbb{E}_\beta[\mathrm{KA}(K^{(f)}, K^{(0)})] = \frac{(1+c)(d+1)}{\sqrt{(d+3)(d+2+\sum_{j=1}^{d}(s_j/\sigma)^4)}} \tag{S5.14}$$

Finally, we see in Eq S5.14 that the maximization of $\mathbb{E}_\beta[KA]$ reduces to the following convex constrained optimization problem:

$$\min_s \sum_j s_j^4, \quad \text{subject to } \sum_j s_j^2 = \sigma^2. \tag{S5.15}$$

The KKT solutions satisfy $s_i^2 = \sigma^2/d$ for all $j = 1 \cdots d$. This implies that the expected tangent kernel alignment is maximized when the initial weight singular values $|s_i|$ are distributed evenly across dimensions, which corresponds to a high-rank initialization. □

Proposition 2. *Following the setup and assumptions in Theorem 1, rank-1 initialization with* $W_1^{(0)} = \sigma[\hat{\beta}^T \quad \vec{0} \quad ... \quad \vec{0}]$ *leads to maximal aligment, i.e,* $\mathrm{KA}(K^{(f)}, K^{(0)}) = 1$ *up to additive* $O(\sigma^2)$ *terms.*

Proof. We indeed have,

$$\begin{aligned} K^{(0)} &= X^T({W_1^{(0)}}^T W_1^{(0)} + \|W_2^{(0)}\|^2 I)X \\ &= \sigma^2 X^T(\hat{\beta}\hat{\beta}^T + I)X \end{aligned} \tag{S5.16}$$

Thus, writing $K := X^T(\hat{\beta}\hat{\beta}^T + I)X$ and using Eq. S5.1, the alignment takes the form

$$\begin{aligned} \mathrm{KA}(K^{(f)}, K^{(0)}) &:= \frac{\mathrm{Tr}(K^{(f)}K^{(0)})}{\|K^{(f)}\|_F \|K^{(0)}\|_F} \\ &= \frac{\mathrm{Tr}(K(K + O(\sigma^2))}{|K\|_F \|K + O(\sigma^2)\|_F} \\ &= \frac{\mathrm{Tr}(K^2)}{\|K\|_F^2} + O(\sigma^2) \\ &= 1 + O(\sigma^2) \end{aligned} \tag{S5.17}$$

□

S5.2.2 Learning requirement based on $W_h^{(0)}$ rank

The focus of this idea is to show that no changes to hidden weights W_h is not possible (e.g. reservoir settings) for zero-error when the initial weight rank falls below a certain threshold. Freezing the hidden weights W_h would be a special case of lazy learning.

Proposition 3. *Consider a linear RNN with input at time t as $X_t \in \mathbb{R}^{N\times d}$ (for $t = 1, ..., T-1$), target output $Y \in \mathbb{R}^{N_{out}\times d}$ only at the last step, recurrent weight matrix $W_h \in \mathbb{R}^{N\times N}$ and readout weight matrix $w \in \mathbb{R}^{N_{out}\times N}$. Here, N, N_{out}, d and T are the number of hidden units, number of classes, number of data points and number of time steps, respectively, and we assume $N, d > N_{out}$. Define initial recurrent weight $W_h^{(0)}$ and final recurrent weight $W_h^{(f)}$ that achieves zero error. Then, for arbitrary input X and target output Y, $W_h^{(f)} = W_h^{(0)}$ is not possible when $rank(W_h^{(0)}) < N_{out}$.*

Proof. We have the following based on the assumption of the RNN structure, if zero-error learning is achieved:

$$Y = w^{(f)} W_h^{(f)} \left(\sum_{t=1}^{T-1} {W_h^{(f)}}^{T-t-1} X_t \right). \tag{S5.18}$$

We can prove by contradiction. Suppose $W_h^{(f)} = W_h^{(0)}$, then

$$Y = w^{(f)} W_h^{(0)} \left(\sum_{t=1}^{T-1} W_h^{(0)T-t-1} X_t \right). \tag{S5.19}$$

Since Y is arbitrary, we can have $rank(Y) = N_{out}$ (by the assumption of $N, d > N_{out}$). Applying $rank(W_h^{(0)}) < N_{out}$ we have

$$\begin{aligned} rank(Y) &= rank(w^{(f)} W_h^{(0)} \left(\sum_{t=1}^{T-1} {W_h^{(0)}}^{T-t-1} X_t \right)) \\ &\overset{(a)}{\leq} min(rank(w^{(f)}), rank(W_h^{(0)}), \left(\sum_{t=1}^{T-1} {W_h^{(0)}}^{T-t-1} X_t \right)) \\ &< N_{out}, \end{aligned} \tag{S5.20}$$

where (a) is because $rank(W_h^{(0)}) < N_{out}$ so the minimum has to be less than N_{out}. This would contradict an arbitrary Y with $rank(Y) = N_{out}$. Thus, $W_h^{(f)} = W_h^{(0)}$ cannot happen and recurrent weights have to be adjusted. □

S5.3 Setup and simulation details

S5.3.1 Initial low-rank weights creation

For the null case, we initialized with random Gaussian distributions where each weight element $W_{ij} \sim \mathcal{N}(0, g^2/N)$, with an initial weight variance of g. Unless otherwise mentioned, we set $g = 1.5$ and network size $N = 300$, though we also validated across other parameter choices (see Appendix S5.4). Input and readout weights were initialized similarly as in [365] (see their $EIRNN.ipynb$ notebook).

To create low-rank weights using SVD, we generated temporary weights $\hat{W}_{ij} \sim \mathcal{N}(0, g^2/N)$. Subsequently, we applied SVD to $\hat{W}$ and retained the top components based on the desired rank. To ensure comparisons are made across constant initial weight magnitudes, the resultant weight matrix was rescaled to match the Frobenius norm of $\hat{W}$.

Furthermore, we present details for experimentally-driven low-rank weights. For block-specific statistics, we followed the setup in Figure S3 of [329], setting parameters as $\alpha = 0.02$, $\gamma = 10$, and $1 - \epsilon = 0.8$. These parameters substantially influence the weight eigenspectrum, as depicted in Figure S3 of [329]; we selected these values specifically to emphasize the outliers and achieve a lower effective rank. These parameters represent the fraction of hyperexcitable neurons (population 1), gain of hyperexcitable connections, and the gain of remaining connections, respectively. For the creation of a chain motif, we employed the procedure described in Section S3.10 of [328], setting $\tau_{chn} = 0.03$ (and $\tau_{chn} = -0.1$ for over-representation or under-representation of the chain motif, respectively). Here, we set $N = 100$. These parameters were chosen to provide enough distinctions from the null case, while still ensuring stability and effective task learning. The electron microscopy (EM) connectivity (of the V1 cortical column model) is obtained from [341], which includes dendritic tree reconstructions and local axonal projections for hundreds of thousands of neurons, detailing their 0.5 billion synaptic connections. From this, we selected 198 cells, focusing on fully proofread neurons closest to the midpoint between layers 2/3 and 4. Connectivity strength for each neuron is determined by summing the volume of each post-synaptic density to target cells, distinguishing between excitatory and inhibitory cell types. For instance, if cell 'a' forms 10 synapses with cell 'b', the connection strength of connection[a,b] represents the

combined volume of synaptic densities at cell 'b'. Inhibitory connections are assigned a sign of -1, while excitatory ones receive +1. For the Dale's law obeying initial connectivity, balanced initialization was done following the process in [365] with 80% excitatory and 20% inhibitory neurons (see the notebook $EIRNN.ipynb$).

It is crucial to highlight that, in testing our Theorem, which examines the effect of the **initial** weight rank, all low-rank modifications are not enforced during training (although the impact of enforcing these structures could be an interesting avenue for future exploration). Weights are adjusted freely based on gradient descent learning.

S5.3.2 Task and training details

Our code is accessible at `https://github.com/Helena-Yuhan-Liu/BioRNN_RichLazy`. We used PyTorch Version 1.10.2 [366]. Simulations were executed on a computer server with x2 20-core Intel(R) Xeon(R) CPU E5-2698 v4 at 2.20GHz, with the average task training duration being around 10 minutes. Following the procedure in [299], which delved deeply into effective laziness metrics, we employed gradient-descent learning with the SGD optimizer. Unless mentioned otherwise, the learning rate was $3e-3$, but we validated that our findings remain consistent across various learning rates (see Appendix S5.4). For stopping, we trained the neurogym tasks for 10000 SGD iterations, which led to comparable terminal losses and accuracies across initializations. For the sMNIST task, we concluded our training upon reaching 97% accuracy, a criterion informed by both published results and our computational resources. We also experimented with halving and doubling the training iterations and observed similar trends. All weights — input, recurrent and readout — were trained. For statistical analysis and significance tests, we used methods in the SciPy Package [367].

For the neuroscience tasks, we adopted the Neurogym framework [16]. Within this paper, these tasks are denoted as "2AF", "DMS", and "CXT", mirroring Neurogym settings: $task =' PerceptualDecisionMaking-v0'$, $task =' DelayMatchSample-v0'$, and $task =' ContextDecisionMaking-v0'$, respectively. To expedite simulations and facilitate numerous runs, we operated with $dt = \tau_m = 100ms$ and abbreviated task durations: for 2AF, settings were $stimulus = 700ms$ and $decision = 100ms$; for DMS, they were $sample = 100ms$,

$delay = 500ms$, $test = 100ms$, and $decision = 100ms$; for CXT, they comprised $stimulus = 200ms$, $delay = 500ms$, and $decision = 100ms$. For these three tasks, we used a batch size of 32 and trained for 10000 iterations.

Regarding the sequential MNIST task [10], we employed a row-by-row format to hasten simulations. Inputs were delivered via $N_{in} = 28$ units, each presenting a row's grey-scaled value, culminating in 28 steps with network predictions rendered at the final step. Training hinged on the cross-entropy loss function; targets were provided throughout training for the neuroscience tasks, as per Neurogym implementation, and targets were provided at only a trial's conclusion for the sequential MNIST task. For this task, we used a batch size of 200 and trained for 10000 iterations.

For the student-teacher two-layer linear network simulations in Figure S5.1A and Figure 5.3, we set $N = 1000$, $d = 2$ (also found similar trends for $d = 20$ and $d = 100$), $z = Fx$, all entries of w (or β when $F = I$) to 1 and entries of X are sampled from a uniform distribution over the interval $[-2, 2]$. We used standard Normal initialization for both W_1 and W_2 with $\sigma = 0.001$. For Figure 5.3, F is constructed from SVD, i.e. $F = USV^T$, with U and V generated from arbitrary orthogonal matrices, and S is a diagonal matrix consisting of the singular values with the top half of the singular values set to κ and bottom half set to 1, where κ is the condition number of F. For the aligned initialization, W_1 is initialized as given in Proposition 2 with $\beta = w^T F$ (w here is illustrated in Figure 5.3), and the F is replaced by its rank-$(d/2)$ truncation for the partially aligned initialization case. For the MNIST task shown in Figure S5.1B, we used a two-layer feedforward network with a ReLU activation function. The architecture consists of an input layer with 784 units corresponding to the image pixels, a hidden layer with 300 units, and a linear readout layer with 10 output units. The weights of the hidden layer and the readout layer were initialized similarly to the input and readout weights, respectively, used in the RNN settings.

S5.4 Additional simulations

We perform additional simulations to show the robustness of our main trends, Low-rank initial recurrent weights lead to greater changes (or effectively richer learning) in RNNs. We show the main trends observed in Figure 5.1 holds also for Uniform initialization (Figure S5.4), soft initial weight rank (Figure S5.5), various network sizes (Figure S5.6), learning rates (Figure S5.7), initial weight gains (Figure S5.8) and Dale's Law constraint throughout training (Figure S5.14), finer simulation time step dt (Figure S5.9) and fixing the leading initial weight eigenvalue (Figure S5.10). The trends in Figure 5.2 also applies to the DMS task (Figure S5.2) and the CXT task (Figure S5.3). Also, without the low-rankedness in the shuffled EM connectivity, the impact on effective laziness also goes away (Figure S5.16). In Figure S5.1 we confirm that the results, shown in Figure 5.1 and predicted by Theorem 1, are also observed in a two-layer linear network setup. Again, we find that in situations where initializations are random, higher rank initialization leads to greater tangent kernel alignment than lower rank cases. We have also tracked the evolution of kernel task alignment (Figure S5.11) and kernel effective rank over the course of training (Figure S5.12).

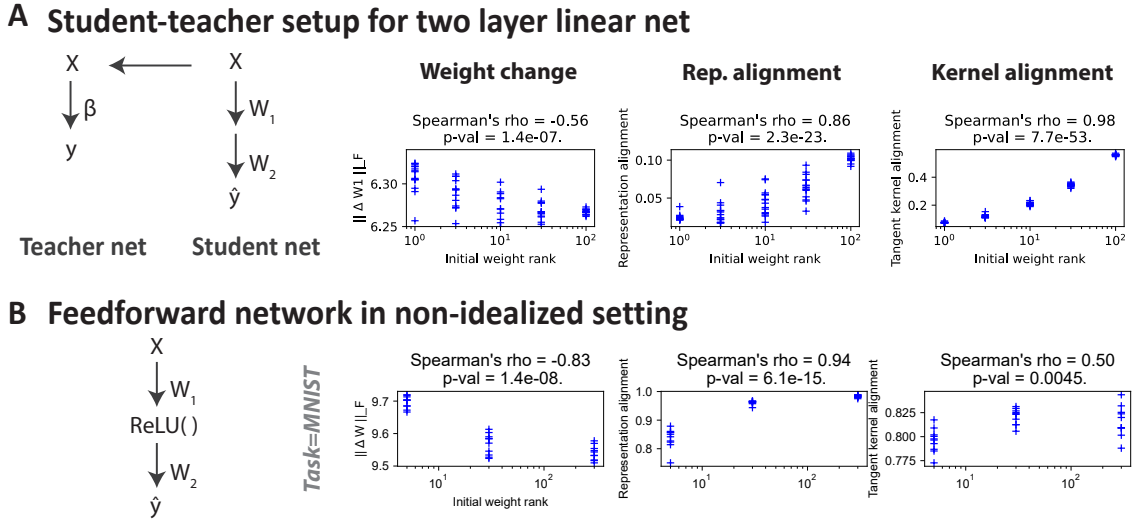

Figure S5.1: As predicted by the theoretical results, higher rank random initialization leads to effectively lazier learning in two-layer linear network. A) We use the student-teacher two-layer linear network setup described in Section 5.2.3. B) a non-idealized setting: two-layer feedforward network with ReLU activation and 300 hidden units trained on the MNIST dataset. Plotting convention follows that of Figure 5.1.

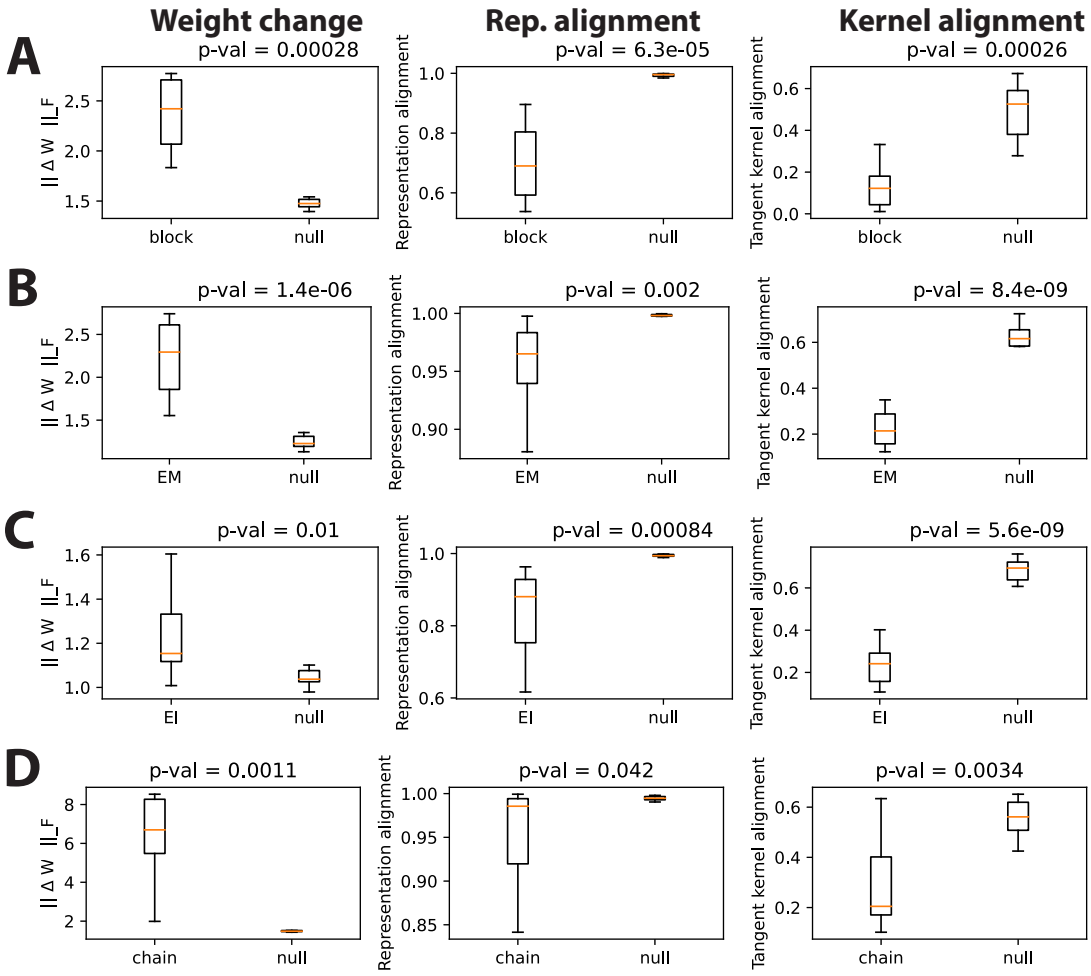

Figure S5.2: We repeated Figure 5.2 for the DMS task and observed similar trends: low-rank initialization, achieved by experimentally-driven initial connectivity in Figure 5.2, leads to effectively richer learning. The plotting conventions used here follow those in Figure 5.2, with panels A-D corresponding to the ones in that figure.

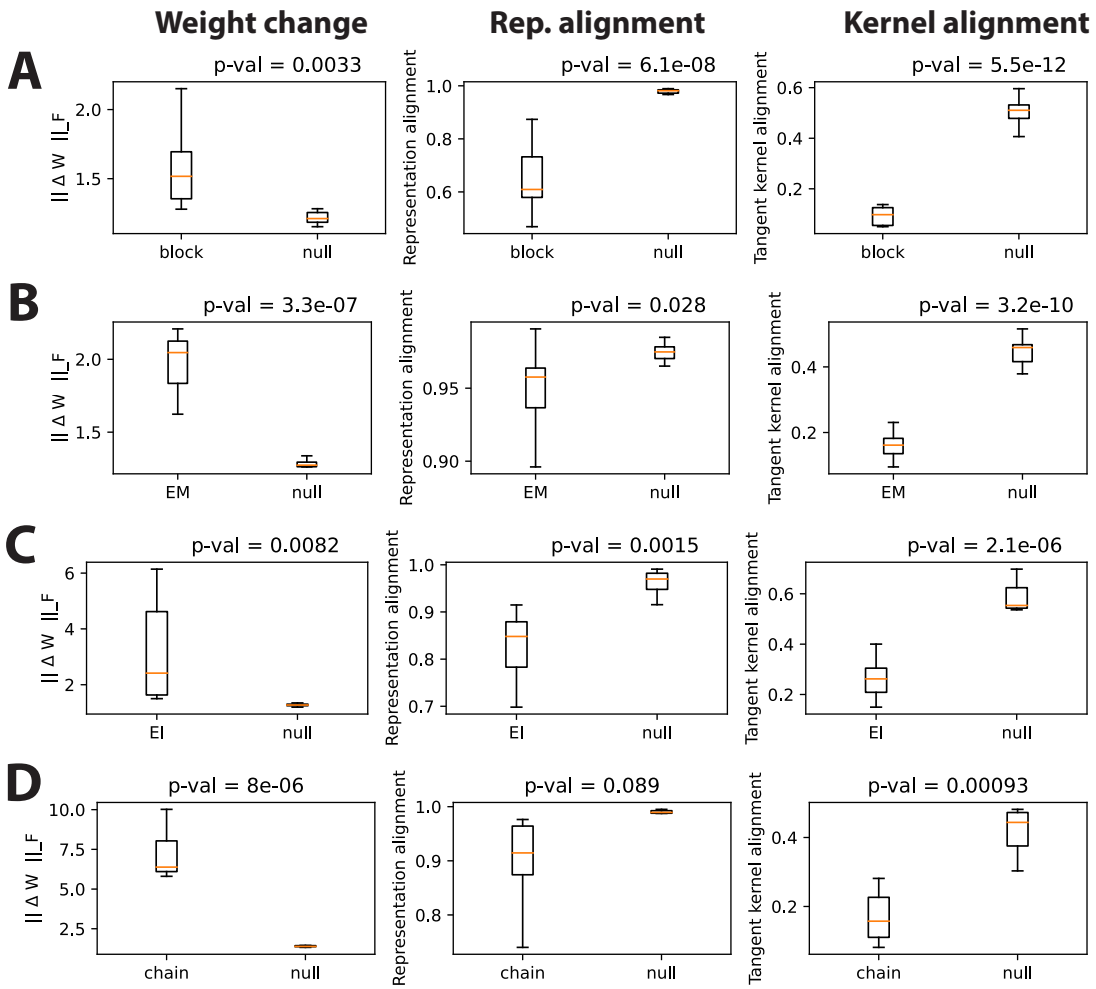

Figure S5.3: We repeated Figure 5.2 for the CXT task and observed similar trends: low-rank initialization, achieved by experimentally-driven initial connectivity in Figure 5.2, leads to effectively richer learning. The plotting conventions used here follow those in Figure 5.2, with panels A-D corresponding to the ones in that figure.

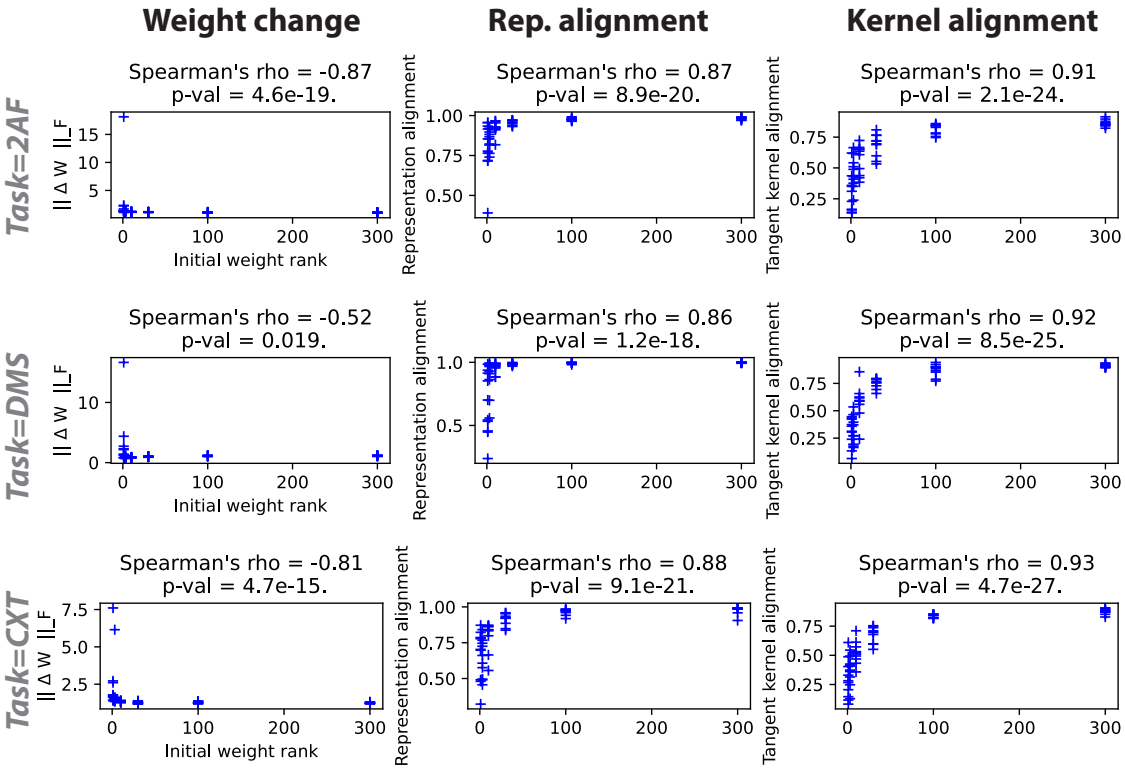

Figure S5.4: Consistent trends observed in Figure 5.1 also for Uniform initialization. We replicated the results of Figure 5.1 — where the initial weights follow a zero-mean Gaussian distribution $W_{ij} \sim \mathcal{N}(0, g^2/N)$ — but now for Uniform initialization $W_{ij} \sim \mathcal{U}\left(-\frac{g}{\sqrt{N}}, \frac{g}{\sqrt{N}}\right)$. Plotting conventions follow that of Figure 5.1.

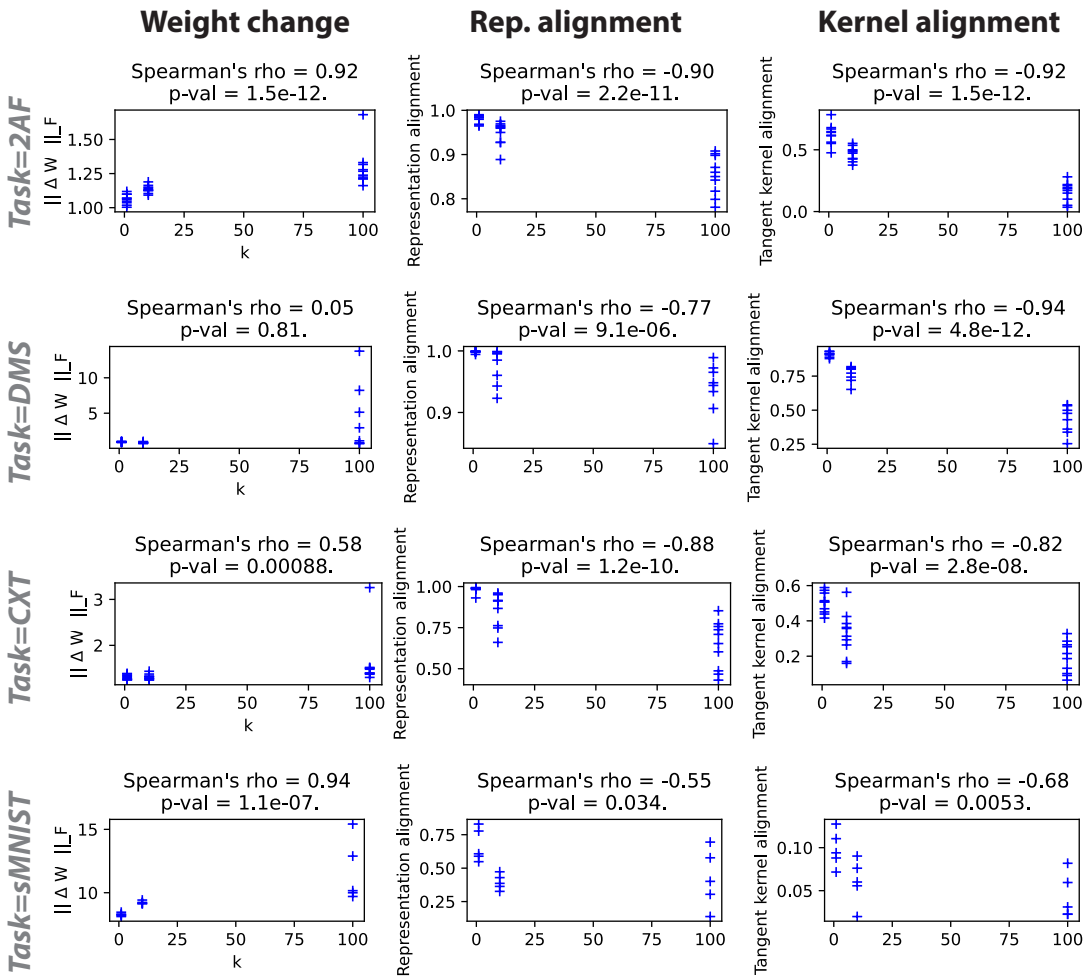

Figure S5.5: Consistent trends observed in Figure 5.1 also for "softer" low-rank weights. Here, instead of the "hard" low-rank weights in Figure 5.1 — where the i^{th} weight singular value s_i is set to 0 if $i > r$ for rank r — we introduce a smoother decay in singular value, where we replace the singular values with $s_i = s_1(1 - i/N)^k$ after performing SVD; this means that greater k leads to lower effective rank. Plotting conventions follow that of Figure 5.1

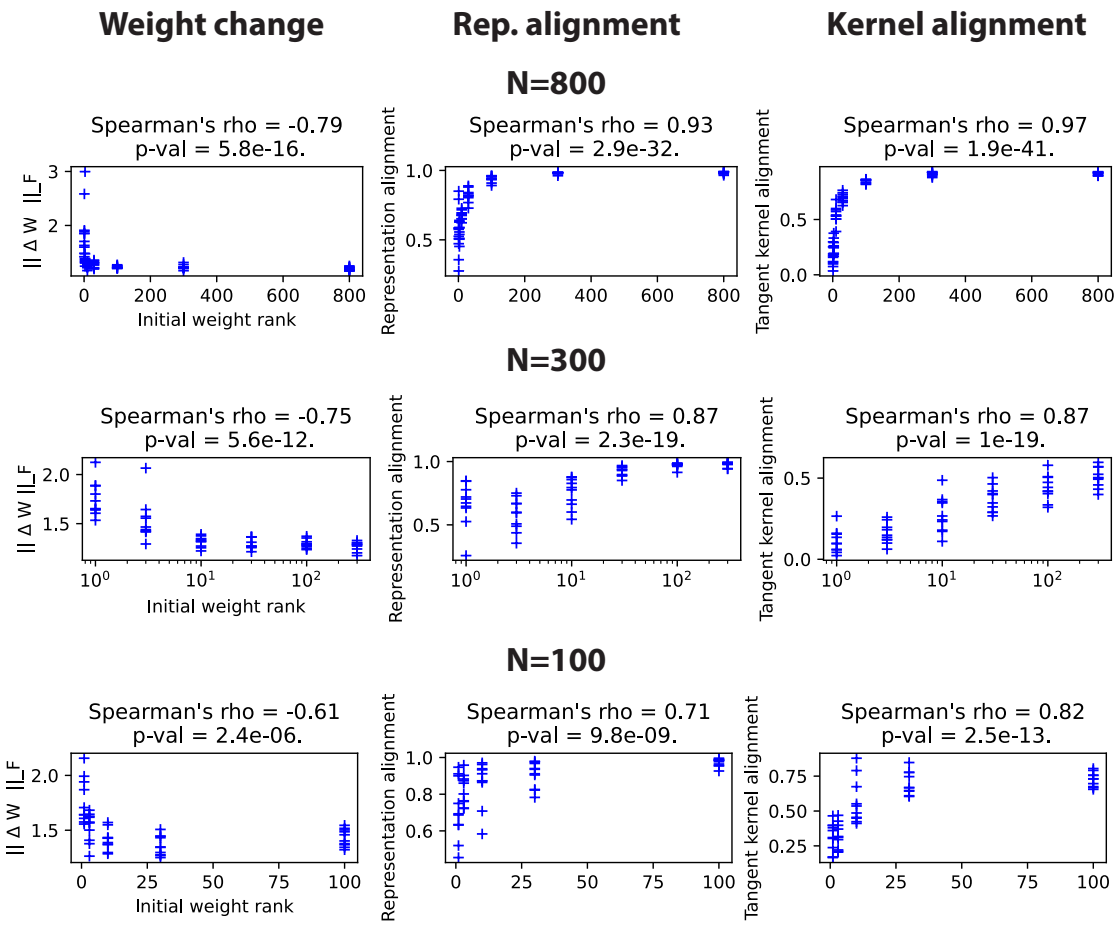

Figure S5.6: Consistent trends observed in Figure 5.1 across various network sizes (N). We replicated the results of Figure 5.1 for different values of N, using the CXT task as an illustrative example. However, the observed trend remains consistent for both the 2AF and DMS tasks. Plotting conventions follow that of Figure 5.1.

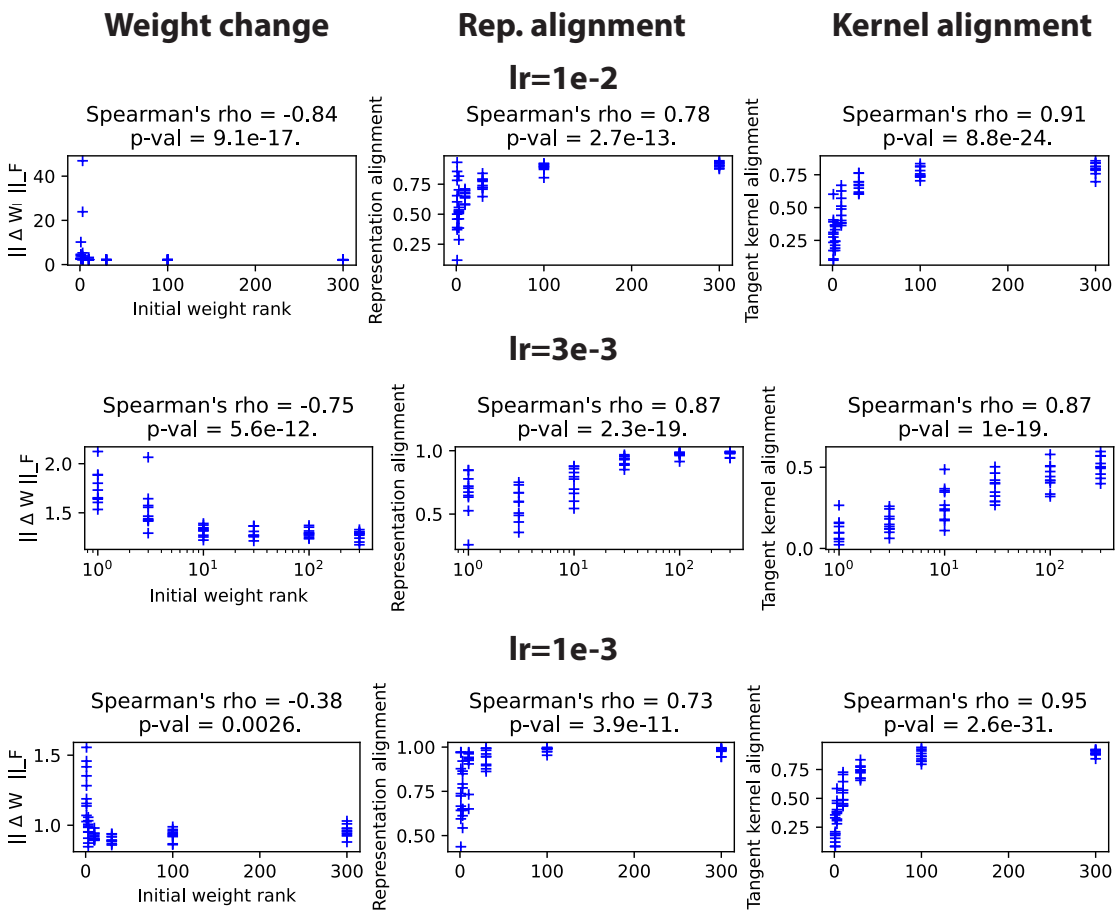

Figure S5.7: Consistent trends observed in Figure 5.1 across various learning rates (lr). We replicated the results of Figure 5.1 for different learning rates, using the CXT task as an illustrative example. However, the observed trend remains consistent for both the 2AF and DMS tasks. Plotting conventions follow that of Figure 5.1.

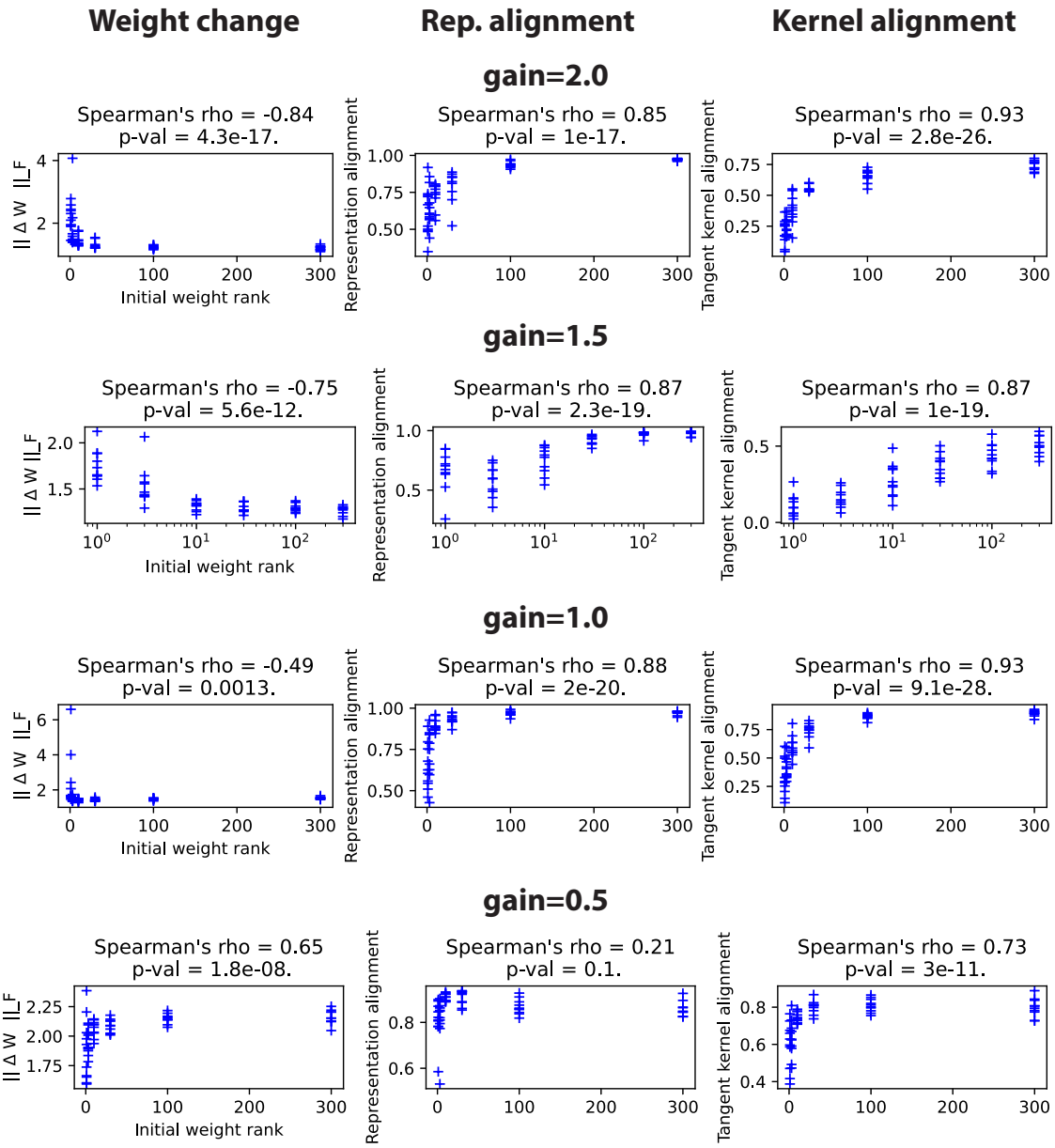

Figure S5.8: Consistent trends observed in Figure 5.1 across various initial gain. Here, the gain refers to g, as weights are initialized as $W_{ij} \sim \mathcal{N}(0, g^2/N)$. The trends hold for most typical range of g from 1.0 to 2.0, but gets weakened for smaller values, $g < 1.0$ (a closer examination of the regime bias in such setting in RNNs is left for future work). We replicated the results of Figure 5.1 for different learning rates, using the CXT task as an illustrative example. However, the observed trend remains consistent for both the 2AF and DMS tasks. Plotting conventions follow that of Figure 5.1.

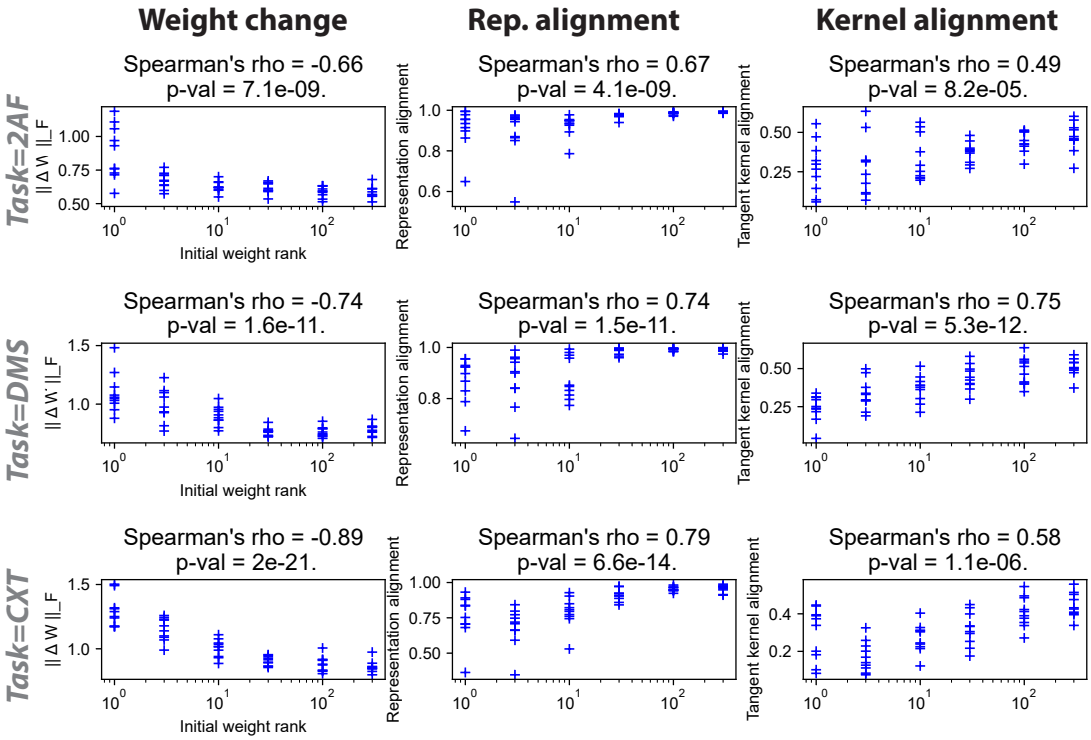

Figure S5.9: Trends in Figure 5.1 are also observed In training RNNs with a fivefold finer time step (dt) and a sequence length extended by five times. As expected, higher rank initializations led to a marked increase in effective laziness. Plotting conventions follows that of Figure 5.1.

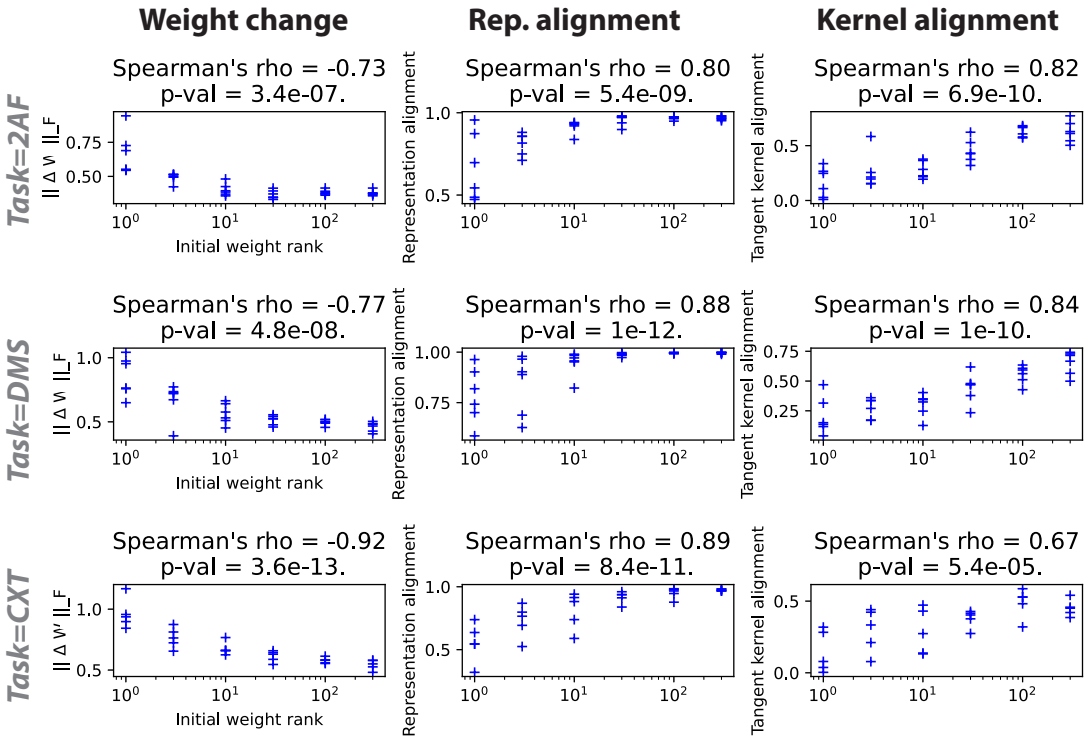

Figure S5.10: Trends in Figure 5.1 are also observed when fixing the leading weight eigenvalue instead of the Frobenius norm across comparisons. As expected, higher rank initializations lead to effectively lazier learning. Plotting conventions follows that of Figure 5.1.

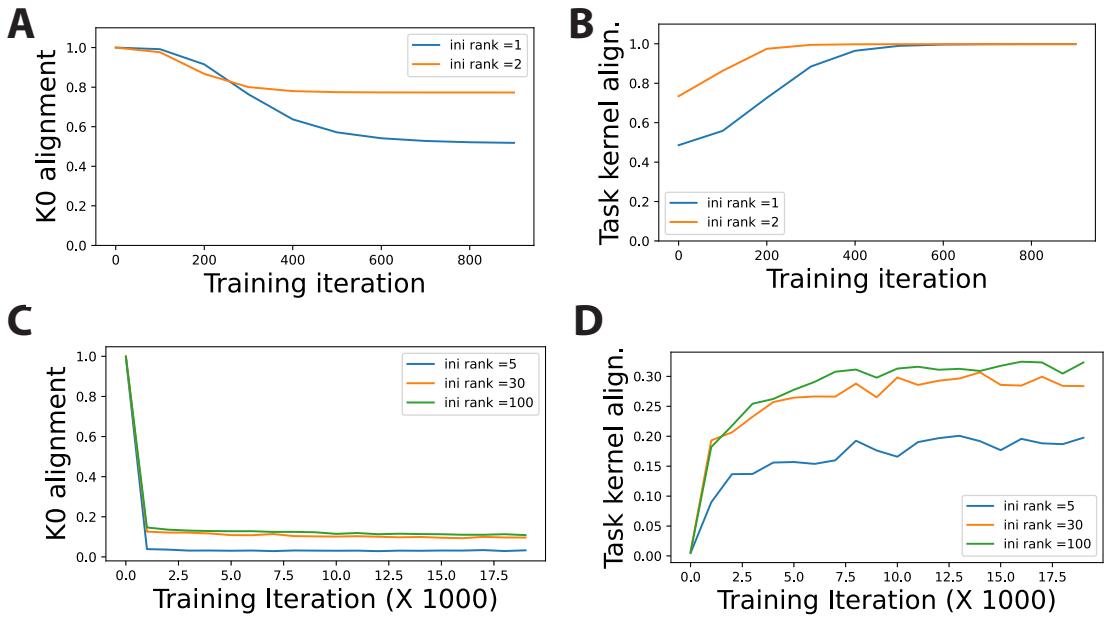

Figure S5.11: [A-B] The idealized two-layer linear network setting from Fig. 2 in [17]. A) Examining the K0 alignment — the alignment between the kernel at various training iterations and the initial kernel — reveals that low-rank random initialization leads to greater changes during training; here, different curves correspond to different initial weight ranks. B) Despite these greater changes, networks with low-rank random initialization take longer to align with the task, as shown by the task kernel alignment metric $y^T Ky/|y|^2 TrK$ throughout training. We remind the reader that y corresponds to the target output and K corresponds to the NTK. **[C-D] A non-idealized setting: the sMNIST task.** C) This panel shows similar trends to A). D) Similar to B), lower-rank random initializations do not achieve as high task kernel alignment within the trained iterations. This is measured by the centered kernel alignment (CKA), which assesses the kernel's alignment with class labels (Eq. 7 in [18]). Although higher CKA values during training could suggest enhanced feature learning (characteristic of the standard rich regime), this aligns with our findings on the effective learning regime, which focuses on changes post-training (see Introduction). Our theory in Section 5.2.3 suggests that lower-rank initializations require greater changes to align with the task, which would typically require more training iterations, as seen in panel B. If training is halted prematurely, perhaps due to resource constraints (as in panel D), these initializations may achieve lower final alignment within the training period. It remains unclear if extended training would lead to similar final alignment across different initializations in a wide range of scenarios. Future research should further investigate the relationship between rankedness of initializations and their impact on the converged solution's representation, including task kernel alignment, across diverse settings.

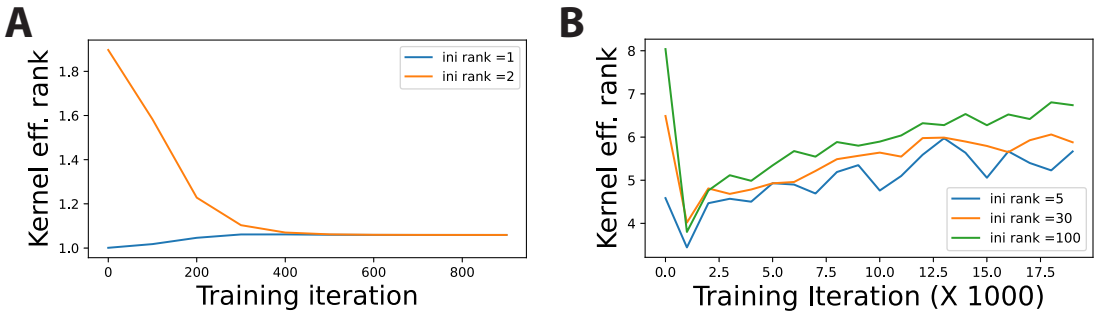

Figure S5.12: The evolution of the leading NTK eigenvalue relative to the rest of the eigenvalues was tracked using an effective rank measure. This measure is based on the ratio of the kernel trace to the kernel dominant eigenvalue, i.e., $\sum_i \lambda_i/\lambda_1$, which indicates the number of eigenvalues on the order of the dominant one. We apply this analysis to A) the idealized setting and B) the sMNIST task, as used in Appendix Figure S5.11 and Figure 5.1, respectively. These results suggest that the kernel effective rank approaches that of the task throughout the training process.

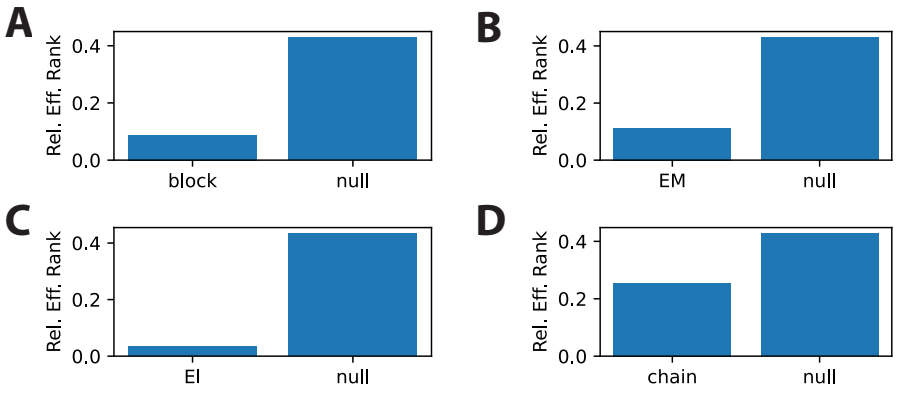

Figure S5.13: Measuring connectivity effective rank based on singular values instead of eigenvalues led to a similar conclusion as Figure 5.2: these experimentally-driven connectivity structures exhibit lower effective rank compared to random Gaussian initialization (null). The plotting conventions used here follow those in Figure 5.2, with panels A-D corresponding to the ones in that figure.

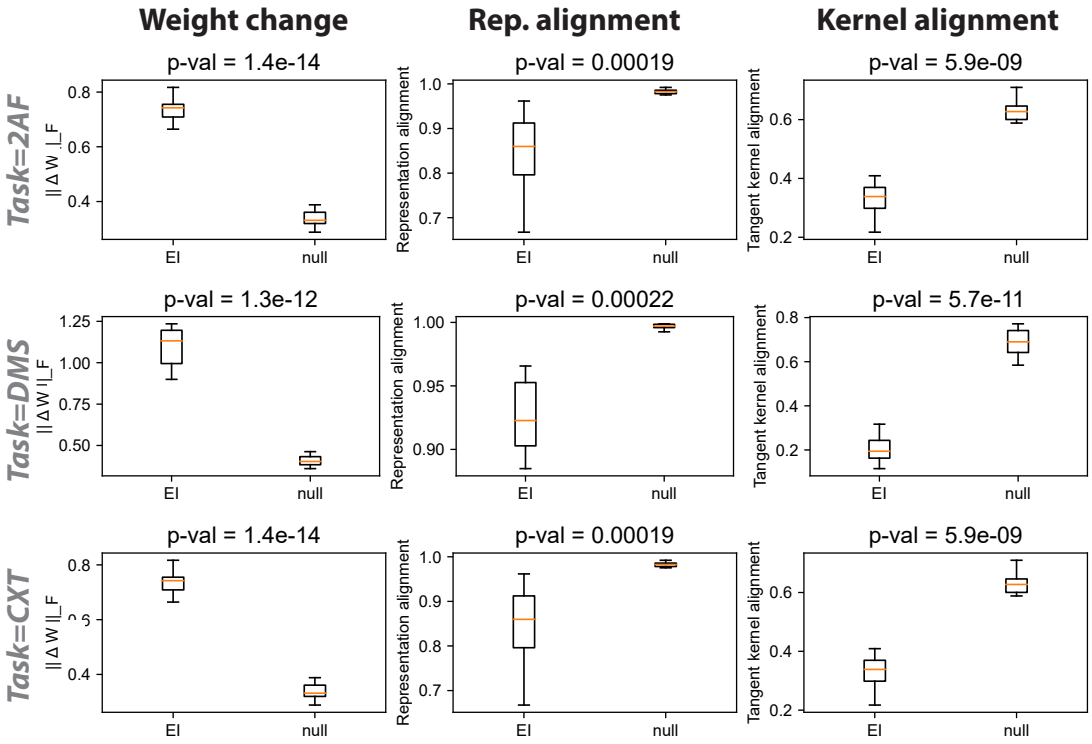

Figure S5.14: Maintaining the constraint of Dale's Law during the entire training process, rather than just at initialization, produced a trend analogous to that observed in Figure 5.2. Plotting conventions follow that of Figure 5.2.

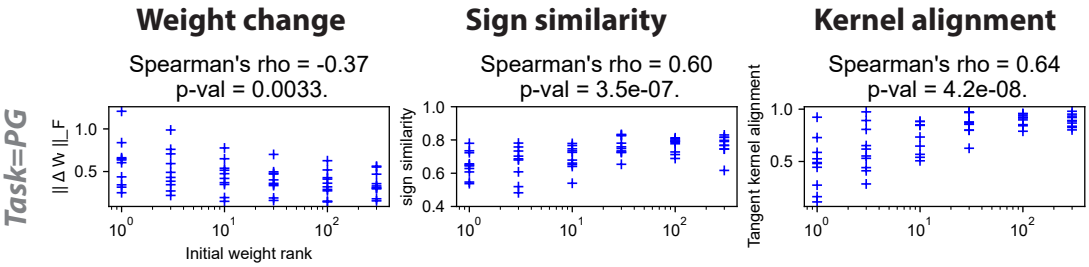

Figure S5.15: Training RNNs on the pattern generation task, as illustrated in Fig. S7 of [19], showed consistent trends with our conclusion: initializations with higher ranks resulted in a more pronounced tendency towards effectively lazier learning. Plotting conventions follows that of Figure 5.1.

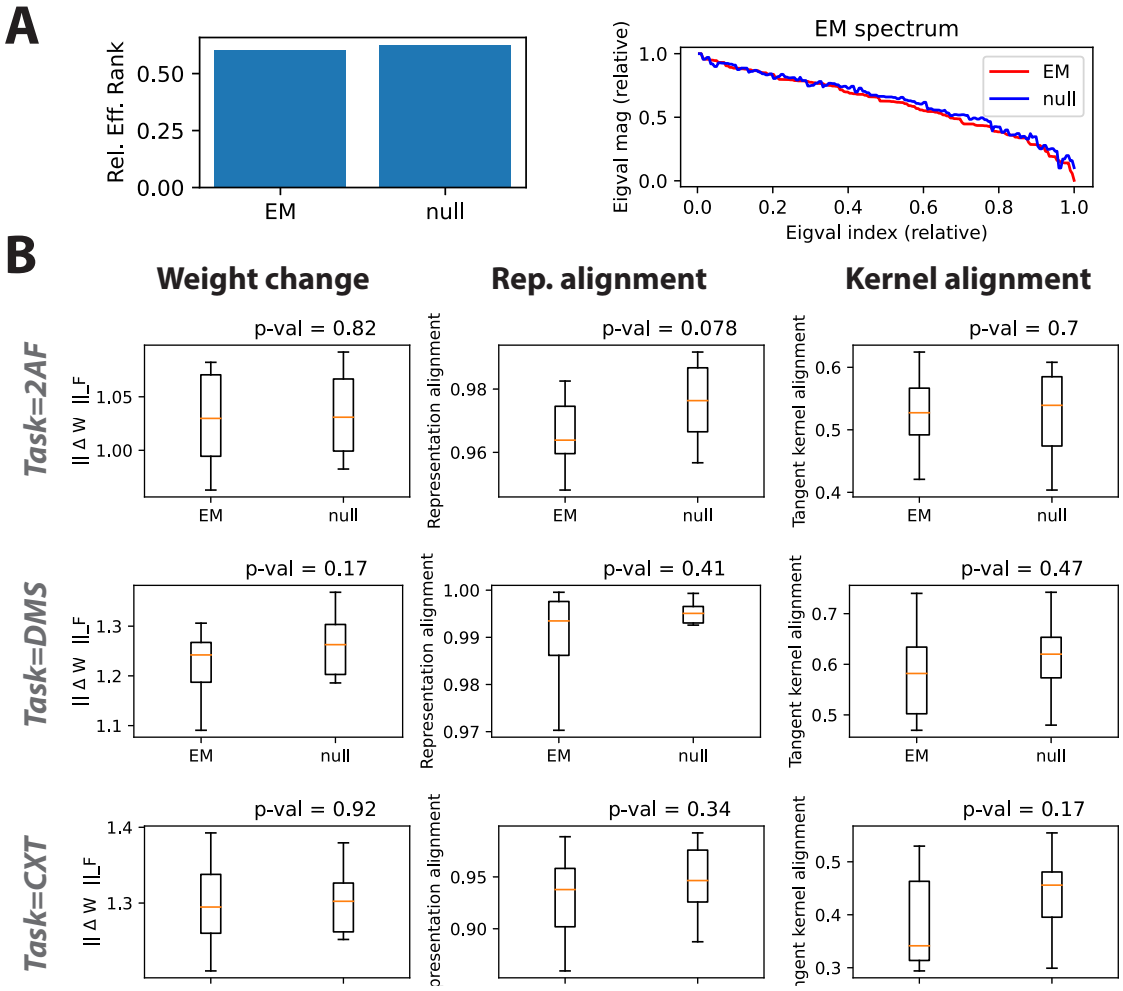

Figure S5.16: Shuffling the EM connectivity, while maintaining the sparsity structure, destroys the low-rankedness and the impact on effective laziness. We repeated the analyses with the EM initial connectivity in Figures 5.2 but performed random shuffling on the EM connectivity, to see if the low-rankedness and the impact on effective laziness is due to the sparsity in the dataset. Performing such shuffling destroys these trends. Plotting conventions follow that of Figure 5.2

Chapter 6

CONCLUSION

This thesis represents a significant stride at the intersection of deep learning and neuroscience, specifically in understanding the principles of learning and generalization within recurrent neural networks (RNNs) used for brain modeling. The research presented here addresses three pivotal directions: the temporal credit assignment problem, the generalization properties of biologically plausible learning rules, and the impact of initial connectivity structures.

First, in tackling the temporal credit assignment problem, this work bridges the gap between artificial neural networks and biological processes. Traditional methods like backpropagation through time (BPTT) have proven effective in artificial RNNs but fall short in biological plausibility. Through a cross-disciplinary collaboration, leveraging transcriptomics data from the Allen Institute, a new bio-plausible learning rule was proposed. This rule predicts the role of local modulatory signals in enhancing the performance of temporal credit assignment, demonstrating improved learning outcomes through extensive numerical experiments and theoretical derivations.

Second, the exploration into the generalization properties of bio-plausible learning rules addresses a fundamental question in neuroscience: how do animals learn complex behaviors with temporal dependencies and generalize this knowledge to novel situations? By collaborating with international interdisciplinary groups and applying deep learning theoretical frameworks, this research uncovered that certain bio-plausible learning rules tend to settle in high-curvature regions of the loss landscape, leading to suboptimal generalization. This finding was backed by a mathematical theorem linking the dynamical stability of weight updates to the loss landscape curvature, and it has opened a constructive discourse on potential remedies, such as modulating the learning rate using neuromodulators.

Third, this thesis shifts the focus to the impact of initial connectivity structures on

learning dynamics. Through empirical and theoretical methodologies, the research highlights the significant influence of initial weight configurations on learning requirements, network changes, metabolic costs, and risks of catastrophic forgetting. This work underscores the importance of initial connectivity in shaping learning trajectories and provides a foundation for optimizing neural network initialization.

In summary, this thesis makes a substantial contribution to the fields of computational neuroscience and deep learning. By investigating how learning rules and initial connectivity structures influence learning and generalization in neural systems, it provides a mechanistic understanding of these processes. Looking forward, much is yet to be done at the intersection of deep learning and neuroscience. One major direction is the experimental validation of the predictions generated by our models; such validation is becoming increasingly feasible due to the recent explosion in neural data availability [368]. I have already initiated several endeavors in this direction. Another promising avenue is the continued transfer of knowledge from deep learning systems to inform the role of neural circuit components in learning and generalization. This area is ripe for exploration, given the wide array of neural elements implicated in the main aspects of neural networks: architecture, learning rules, and tasks [61, 365]. How do these elements impact computation? For instance, as discussed in earlier chapters, how would more realistic neuronal models [260] enhance biologically plausible learning and subsequent generalization performance? Additionally, given the growing popularity of language models and the demonstrated effectiveness of transformers, can such models inspire new insights into biological learning? These models can learn in-context [369], rather than relying solely on learning via parameter updates central to this thesis. Is there a biologically plausible way to achieve effective in-context learning? Overall, the intersection of deep learning and neuroscience offers many exciting opportunities due to recent advancements in deep learning tools and the avalanche of neural data availability. By leveraging the synergy between these two fields, we can gain a deeper understanding of how the brain efficiently learns tasks, with the hope of simultaneously advancing both fields.

BIBLIOGRAPHY

[1] Ben Engelhard, Joel Finkelstein, Julia Cox, Weston Fleming, Hee Jae Jang, Sharon Ornelas, Sue Ann Koay, Stephan Y. Thiberge, Nathaniel D. Daw, David W. Tank, and Ilana B. Witten. Specialized coding of sensory, motor and cognitive variables in VTA dopamine neurons. *Nature*, 570(7762):509–513, jun 2019.

[2] Ari S. Morcos and Christopher D. Harvey. History-dependent variability in population dynamics during evidence accumulation in cortex. *Nature Neuroscience*, 19(12):1672–1681, dec 2016.

[3] Stephen J. Smith, Michael Hawrylycz, Jean Rossier, and Uygar Sümbül. New light on cortical neuropeptides and synaptic network plasticity. *Current Opinion in Neurobiology*, 63:176–188, aug 2020.

[4] Guillaume Bellec, Franz Scherr, Anand Subramoney, Elias Hajek, Darjan Salaj, Robert Legenstein, and Wolfgang Maass. A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature Communications*, 11(1):1–15, dec 2020.

[5] Guillaume Bellec, David Kappel, Wolfgang Maass, and Robert Legenstein. Deep rewiring: Training very sparse deep networks. In *International Conference on Learning Representations*, 2018.

[6] Rebecca Elliott and Raymond J Dolan. Differential neural responses during performance of matching and nonmatching to sample tasks at two delay intervals. *Journal of Neuroscience*, 19(12):5066–5073, 1999.

[7] Nicolas Frémaux and Wulfram Gerstner. Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules. *Frontiers in neural circuits*, 9:85, 2016.

[8] Wulfram Gerstner, Marco Lehmann, Vasiliki Liakoni, Dane Corneil, and Johanni Brea. Eligibility Traces and Plasticity on Behavioral Time Scales: Experimental Support of NeoHebbian Three-Factor Learning Rules. *Frontiers in Neural Circuits*, 12:53, jul 2018.

[9] James M. Murray. Local online learning in recurrent networks with random feedback. *eLife*, 8, may 2019.

[10] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[11] Yuhan Helena Liu, Stephen Smith, Stefan Mihalas, Eric Shea-Brown, and Uygar Sümbül. Cell-type–specific neuromodulation guides synaptic credit assignment in a spiking neural network. *Proceedings of the National Academy of Sciences*, 118(51), 2021.

[12] Timothy P Lillicrap, Daniel Cownden, Douglas B Tweed, and Colin J Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications*, 7(1):1–10, 2016.

[13] Asier Mujika, Florian Meier, and Angelika Steger. Approximating real-time recurrent learning with random kronecker factors. *Advances in Neural Information Processing Systems*, 31, 2018.

[14] Henning Petzka, Michael Kamp, Linara Adilova, Cristian Sminchisescu, and Mario Boley. Relative flatness and generalization. *Advances in Neural Information Processing Systems*, 34, 2021.

[15] Stanisław Jastrzębski, Zachary Kenton, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. On the relation between the sharpest directions of dnn loss and the sgd step length. *arXiv preprint arXiv:1807.05031*, 2018.

[16] Manuel Molano-Mazon, Joao Barbosa, Jordi Pastor-Ciurana, Marta Fradera, Ru-Yuan Zhang, Jeremy Forest, Jorge del Pozo Lerida, Li Ji-An, Christopher J Cueva, Jaime de la Rocha, et al. Neurogym: An open resource for developing and sharing neuroscience tasks. 2022.

[17] Alexander Atanasov, Blake Bordelon, and Cengiz Pehlevan. Neural networks as kernel learners: The silent alignment effect. *arXiv preprint arXiv:2111.00034*, 2021.

[18] Aristide Baratin, Thomas George, César Laurent, R Devon Hjelm, Guillaume Lajoie, Pascal Vincent, and Simon Lacoste-Julien. Implicit regularization via neural feature alignment. In *International Conference on Artificial Intelligence and Statistics*, pages 2269–2277. PMLR, 2021.

[19] Guillaume Bellec, Franz Scherr, Anand Subramoney, Elias Hajek, Darjan Salaj, Robert Legenstein, and Wolfgang Maass. A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature communications*, 11(1):3625, 2020.

[20] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, may 2015.

[21] Guillaume Bellec, Darjan Salaj, Anand Subramoney, Robert Legenstein, and Wolfgang Maass. Long short-term memory and learning-to-learn in networks of spiking neurons. In *32nd Conference on Neural Information Processing Systems*, pages 787–797, 2018.

[22] Pieter R. Roelfsema and Anthony Holtmaat. Control of synaptic plasticity in deep cortical networks. *Nature Reviews Neuroscience*, 19(3):166–180, feb 2018.

[23] Jan Benda and Andreas VM Herz. A universal model for spike-frequency adaptation. *Neural computation*, 15(11):2523–2564, 2003.

[24] Alba Bellot-Saez, Orsolya Kékesi, John W Morley, and Yossi Buskila. Astrocytic modulation of neuronal excitability through k+ spatial buffering. *Neuroscience & Biobehavioral Reviews*, 77:87–97, 2017.

[25] Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500–544, 1952.

[26] Yoshua Bengio, Ian Goodfellow, and Aaron Courville. *Deep learning*, volume 1. MIT press Massachusetts, USA:, 2017.

[27] Hugh R Wilson and Jack D Cowan. Excitatory and inhibitory interactions in localized populations of model neurons. *Biophysical journal*, 12(1):1–24, 1972.

[28] Valerio Mante, David Sussillo, Krishna V. Shenoy, and William T. Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, 503(7474):78–84, nov 2013.

[29] Evan D Remington, Devika Narain, Eghbal A Hosseini, and Mehrdad Jazayeri. Flexible sensorimotor computations through rapid reconfiguration of cortical dynamics. *Neuron*, 98(5):1005–1019, 2018.

[30] Saurabh Vyas, Matthew D Golub, David Sussillo, and Krishna V Shenoy. Computation through neural population dynamics. *Annual Review of Neuroscience*, 43:249–275, 2020.

[31] Donald Olding Hebb. The organization of behavior; a neuropsycholocigal theory. *A Wiley Book in Clinical Psychology*, 62:78, 1949.

[32] Tim VP Bliss and Terje Lømo. Long-lasting potentiation of synaptic transmission in the dentate area of the anaesthetized rabbit following stimulation of the perforant path. *The Journal of physiology*, 232(2):331–356, 1973.

[33] Liqun Luo. *Principles of neurobiology*. Garland Science, 2015.

[34] Guo-qiang Bi and Mu-ming Poo. Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of neuroscience*, 18(24):10464–10472, 1998.

[35] Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15(3):267–273, 1982.

[36] Carlos SN Brito and Wulfram Gerstner. Nonlinear hebbian learning as a unifying principle in receptive field formation. *PLoS computational biology*, 12(9):e1005070, 2016.

[37] Wolfram Schultz. Neuronal reward and decision signals: from theories to data. *Physiological reviews*, 95(3):853–951, 2015.

[38] Sho Yagishita, Akiko Hayashi-Takagi, Graham C.R. Ellis-Davies, Hidetoshi Urakubo, Shin Ishii, and Haruo Kasai. A critical time window for dopamine actions on the structural plasticity of dendritic spines. *Science*, 345(6204):1616–1620, sep 2014.

[39] Wolfram Schultz, Peter Dayan, and P Read Montague. A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599, 1997.

[40] Michael A. Farries and Adrienne L. Fairhall. Reinforcement Learning With Modulated Spike Timing–Dependent Synaptic Plasticity. *Journal of Neurophysiology*, 98(6):3648–3665, dec 2007.

[41] Charles B Delahunt and J Nathan Kutz. Putting a bug in ml: The moth olfactory network learns to read mnist. *Neural Networks*, 118:54–64, 2019.

[42] Jeffrey C. Magee and Christine Grienberger. Synaptic Plasticity Forms and Functions. *Annual Review of Neuroscience*, 43(1):95–117, jul 2020.

[43] Eugene M Izhikevich. *Dynamical systems in neuroscience*. MIT press, 2007.

[44] Robert Legenstein, Dejan Pecevski, and Wolfgang Maass. A learning theory for reward-modulated spike-timing-dependent plasticity with application to biofeedback. *PLoS Comput Biol*, 4(10):e1000180, 2008.

[45] David Zipser and Richard A Andersen. A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons. *Nature*, 331(6158):679–684, 1988.

[46] Timothy P Lillicrap and Stephen H Scott. Preference distributions of primary motor cortex neurons reflect control solutions optimized for limb biomechanics. *Neuron*, 77(1):168–179, 2013.

[47] Li K Wenliang and Aaron R Seitz. Deep neural networks for modeling visual perceptual learning. *Journal of Neuroscience*, 38(27):6028–6044, 2018.

[48] Seyed-Mahdi Khaligh-Razavi and Nikolaus Kriegeskorte. Deep supervised, but not unsupervised, models may explain it cortical representation. *PLoS computational biology*, 10(11):e1003915, 2014.

[49] Nikolaus Kriegeskorte. Deep neural networks: a new framework for modeling biological vision and brain information processing. *Annual review of vision science*, 1:417–446, 2015.

[50] Daniel LK Yamins and James J DiCarlo. Using goal-driven deep learning models to understand sensory cortex. *Nature neuroscience*, 19(3):356–365, 2016.

[51] Dean A Pospisil, Anitha Pasupathy, and Wyeth Bair. 'artiphysiology'reveals v4-like shape tuning in a deep network trained for image classification. *Elife*, 7:e38242, 2018.

[52] Yosef Singer, Yayoi Teramoto, Ben DB Willmore, Jan WH Schnupp, Andrew J King, and Nicol S Harper. Sensory cortex is optimized for prediction of future input. *Elife*, 7:e31557, 2018.

[53] Eiji Watanabe, Akiyoshi Kitaoka, Kiwako Sakamoto, Masaki Yasugi, and Kenta Tanaka. Illusory motion reproduced by deep neural networks trained for prediction. *Frontiers in psychology*, 9:345, 2018.

[54] Jane X Wang, Zeb Kurth-Nelson, Dharshan Kumaran, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Demis Hassabis, and Matthew Botvinick. Prefrontal cortex as a meta-reinforcement learning system. *Nature neuroscience*, 21(6):860–868, 2018.

[55] Charles F Cadieu, Ha Hong, Daniel LK Yamins, Nicolas Pinto, Diego Ardila, Ethan A Solomon, Najib J Majaj, and James J DiCarlo. Deep neural networks rival the representation of primate it cortex for core visual object recognition. *PLoS Comput Biol*, 10(12):e1003963, 2014.

[56] Timothy P Lillicrap, Adam Santoro, Luke Marris, Colin J Akerman, and Geoffrey Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346, 2020.

[57] Timothy P. Lillicrap and Adam Santoro. Backpropagation through time and the brain. *Current Opinion in Neurobiology*, 55:82–89, apr 2019.

[58] Matthew E Larkum, J Julius Zhu, and Bert Sakmann. A new cellular mechanism for coupling inputs arriving at different cortical layers. *Nature*, 398(6725):338–341, 1999.

[59] Alexandre Payeur, Jordan Guerguiev, Friedemann Zenke, Blake Richards, and Richard Naud. Burst-dependent synaptic plasticity can coordinate learning in hierarchical circuits. *bioRxiv*, 2020.

[60] Justin Werfel, Xiaohui Xie, and H Sebastian Seung. Learning curves for stochastic gradient descent in linear feedforward networks. *Neural computation*, 17(12):2699–2718, 2005.

[61] Blake A Richards, Timothy P Lillicrap, Philippe Beaudoin, Yoshua Bengio, Rafal Bogacz, Amelia Christensen, Claudia Clopath, Rui Ponte Costa, Archy de Berker, Surya Ganguli, et al. A deep learning framework for neuroscience. *Nature neuroscience*, 22(11):1761–1770, 2019.

[62] Stephen J. Smith, Uygar Sümbül, Lucas T. Graybuck, Forrest Collman, Sharmishtaa Seshamani, Rohan Gala, Olga Gliko, Leila Elabbady, Jeremy A. Miller, Trygve E. Bakken, Jean Rossier, Zizhen Yao, Ed Lein, Hongkui Zeng, Bosiljka Tasic, and Michael Hawrylycz. Single-cell transcriptomic evidence for dense intracortical neuropeptide networks. *eLife*, 8, nov 2019.

[63] Ronald J. Williams and David Zipser. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation*, 1(2):270–280, jun 1989.

[64] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[65] Yuhan Helena Liu, Stephen Smith, Stefan Mihalas, Eric Shea-Brown, and Uygar Sümbül. Cell-type–specific neuromodulation guides synaptic credit assignment in a spiking neural network. *Proceedings of the National Academy of Sciences*, 118(51):e2111821118, 2021.

[66] Yuhan Helena Liu, Stephen Smith, Stefan Mihalas, Eric Shea-Brown, and Uygar Sümbül. Biologically-plausible backpropagation through arbitrary timespans via local neuromodulators. *arXiv preprint arXiv:2206.01338*, 2022.

[67] Yuhan Helena Liu, Arna Ghosh, Blake Richards, Eric Shea-Brown, and Guillaume Lajoie. Beyond accuracy: generalization properties of bio-plausible temporal credit assignment rules. *Advances in Neural Information Processing Systems*, 35:23077–23097, 2022.

[68] Yuhan Helena Liu, Aristide Baratin, Jonathan Cornford, Stefan Mihalas, Eric Shea-Brown, and Guillaume Lajoie. How connectivity structure shapes rich and lazy learning in neural circuits. *arXiv preprint arXiv:2310.08513*, 2023.

[69] Terrence J Sejnowski. *The deep learning revolution*. Mit Press, 2018.

[70] R. J. Williams and D. Zipser. Gradient-based learning algorithms for recurrent networks and their computational complexity. In Y. Chauvin and D. E. Rumelhart, editors, *Back-propagation: Theory, Architectures and Applications*, chapter 13, pages 433–486. Hillsdale, NJ: Erlbaum, 1995.

[71] Owen Marschall, Kyunghyun Cho, and Cristina Savin. A unified framework of online learning algorithms for training recurrent neural networks. *Journal of Machine Learning Research*, 21(135):1–34, 2020.

[72] Asier Mujika, Florian Meier, and Angelika Steger. Approximating Real-Time Recurrent Learning with Random Kronecker Factors. In *32nd Conference on Neural Information Processing Systems*, pages 6594–6603, 2018.

[73] Corentin Tallec and Yann Ollivier. Unbiased Online Recurrent Optimization. In *ICLR*, feb 2018.

[74] Christopher Roth, Ingmar Kanitscheider, and Ila Fiete. Kernel RNN Learning (KERNL). In *ICLR*, sep 2019.

[75] Jacob Menick, Erich Elsen, Utku Evci, Simon Osindero, Karen Simonyan, and Alex Graves. A practical sparse approximation for real time recurrent learning. *arXiv preprint arXiv:2006.07232*, 2020.

[76] Friedemann Zenke and Emre O Neftci. Brain-inspired learning on neuromorphic substrates. *arXiv preprint arXiv:2010.11931*, 2020.

[77] Emre O. Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, nov 2019.

[78] Blake A. Richards and Timothy P. Lillicrap. Dendritic solutions to the credit assignment problem. *Current Opinion in Neurobiology*, 54:28–36, feb 2019.

[79] Jonathan E Rubin, Catalina Vich, Matthew Clapp, Kendra Noneman, and Timothy Verstynen. The credit assignment problem in cortico-basal ganglia-thalamic networks: A review, a problem and a possible solution. *European Journal of Neuroscience*, 53(7):2234–2253, 2021.

[80] Isabella Pozzi, Sander Bohté, and Pieter Roelfsema. A biologically plausible learning rule for deep learning in the brain. *arXiv preprint arXiv:1811.01768*, 2018.

[81] João Sacramento, Rui Ponte Costa, Yoshua Bengio, and Walter Senn. Dendritic cortical microcircuits approximate the backpropagation algorithm. *arXiv preprint arXiv:1810.11393*, 2018.

[82] Axel Laborieux, Maxence Ernoult, Benjamin Scellier, Yoshua Bengio, Julie Grollier, and Damien Querlioz. Scaling equilibrium propagation to deep convnets by drastically reducing its gradient estimator bias. *Frontiers in neuroscience*, 15:129, 2021.

[83] Yali Amit. Deep learning with asymmetric connections and hebbian updates. *Frontiers in computational neuroscience*, 13:18, 2019.

[84] Beren Millidge, Alexander Tschantz, Anil K Seth, and Christopher L Buckley. Activation

relaxation: A local dynamical approximation to backpropagation in the brain. *arXiv preprint arXiv:2009.05359*, 2020.

[85] Peter Dayan and Laurence F Abbott. *Theoretical neuroscience: computational and mathematical modeling of neural systems*. Computational Neuroscience Series, 2001.

[86] Zuzanna Brzosko, Susanna B Mierau, and Ole Paulsen. Neuromodulation of spike-timing-dependent plasticity: past, present, and future. *Neuron*, 103(4):563–581, 2019.

[87] Nicolas X. Tritsch and Bernardo L. Sabatini. Dopaminergic Modulation of Synaptic Transmission in Cortex and Striatum. *Neuron*, 76(1):33–50, oct 2012.

[88] Eve Marder. Neuromodulation of neuronal circuits: back to the future. *Neuron*, 76(1):1–11, 2012.

[89] Arif A Hamid, Michael J Frank, and Christopher I Moore. Wave-like dopamine dynamics as a mechanism for spatiotemporal credit assignment. *Cell*, 184(10):2733–2749, 2021.

[90] Aparna Suvrathan. Beyond STDP — towards diverse and functionally relevant plasticity rules. *Current Opinion in Neurobiology*, 54:12–19, feb 2019.

[91] Anthony N. van den Pol. Neuropeptide Transmission in Brain Circuits. *Neuron*, 76(1):98–115, oct 2012.

[92] Corinne Teeter, Ramakrishnan Iyer, Vilas Menon, Nathan Gouwens, David Feng, Jim Berg, Aaron Szafer, Nicholas Cain, Hongkui Zeng, Michael Hawrylycz, et al. Generalized leaky integrate-and-fire models classify multiple neuron types. *Nature communications*, 9(1):1–15, 2018.

[93] Bosiljka Tasic, Zizhen Yao, Lucas T. Graybuck, Kimberly A. Smith, Thuc Nghi Nguyen, Darren Bertagnolli, Jeff Goldy, Emma Garren, Michael N. Economo, Sarada Viswanathan, Osnat Penn, Trygve Bakken, Vilas Menon, Jeremy Miller, Olivia Fong, Karla E. Hirokawa, Kanan Lathia, Christine Rimorin, Michael Tieu, Rachael Larsen, Tamara Casper, Eliza Barkan, Matthew Kroll, Sheana Parry, Nadiya V. Shapovalova, Daniel Hirschstein, Julie Pendergraft, Heather A. Sullivan, Tae Kyung Kim, Aaron

Szafer, Nick Dee, Peter Groblewski, Ian Wickersham, Ali Cetin, Julie A. Harris, Boaz P. Levi, Susan M. Sunkin, Linda Madisen, Tanya L. Daigle, Loren Looger, Amy Bernard, John Phillips, Ed Lein, Michael Hawrylycz, Karel Svoboda, Allan R. Jones, Christof Koch, and Hongkui Zeng. Shared and distinct transcriptomic cell types across neocortical areas. *Nature*, 563(7729):72–78, nov 2018.

[94] Wilten Nicola and Claudia Clopath. Supervised learning in spiking neural networks with FORCE training. *Nature Communications*, 8(1):1–15, dec 2017.

[95] Travis Meyer, Xue Lian Qi, Terrence R. Stanford, and Christos Constantinidis. Stimulus selectivity in dorsal and ventral prefrontal cortex after training in working memory tasks. *Journal of Neuroscience*, 31(17):6266–6276, apr 2011.

[96] Gáspár Jékely. The chemical brain hypothesis for the origin of nervous systems. *Philosophical Transactions of the Royal Society B*, 376(1821):20190761, 2021.

[97] Drew Linsley, Alekh Karkada Ashok, Lakshmi Narasimhan Govindarajan, Rex Liu, and Thomas Serre. Stable and expressive recurrent vision models. *arXiv preprint arXiv:2005.11362*, 2020.

[98] Dongsung Huh and Terrence J Sejnowski. Gradient Descent for Spiking Neural Networks. In *32nd Conference on Neural Information Processing Systems*, pages 1433–1443, 2018.

[99] Nathan W Gouwens, Staci A Sorensen, Jim Berg, Changkyu Lee, Tim Jarsky, Jonathan Ting, Susan M Sunkin, David Feng, Costas A Anastassiou, Eliza Barkan, et al. Classification of electrophysiological and morphological neuron types in the mouse visual cortex. *Nature neuroscience*, 22(7):1182–1195, 2019.

[100] Hui Lu, Hyungju Park, and Mu-Ming Poo. Spike-timing-dependent bdnf secretion and synaptic plasticity. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 369(1633):20130132, 2014.

[101] Elissa J Donzis and Natalie C Tronson. Modulation of learning and memory by cytokines: signaling mechanisms and long term consequences. *Neurobiology of learning and memory*, 115:68–77, 2014.

[102] Skylar M Spangler and Michael R Bruchas. Optogenetic approaches for dissecting neuromodulation and gpcr signaling in neural circuits. *Current opinion in pharmacology*, 32:56–70, 2017.

[103] Sarah Melzer, Elena Newmark, Grace Or Mizuno, Minsuk Hyun, Adrienne C Philson, Eleonora Quiroli, Beatrice Righetti, Malika R Gregory, Kee Wui Huang, James Levasseur, et al. Bombesin-like peptide recruits disinhibitory cortical circuits and enhances fear memories. *Available at SSRN 3724673*, 2020.

[104] Amirsaman Sajad, David C. Godlove, and Jeffrey D. Schall. Cortical microcircuitry of performance monitoring. *Nature Neuroscience*, 22(2):265–274, feb 2019.

[105] Steven K. Esser, Paul A. Merolla, John V. Arthur, Andrew S. Cassidy, Rathinakumar Appuswamy, Alexander Andreopoulos, David J. Berg, Jeffrey L. McKinstry, Timothy Melano, Davis R. Barch, Carmelo Di Nolfo, Pallab Datta, Arnon Amir, Brian Taba, Myron D. Flickner, and Dharmendra S. Modha. Convolutional networks for fast, energy-efficient neuromorphic computing. *Proceedings of the National Academy of Sciences of the United States of America*, 113(41):11441–11446, oct 2016.

[106] Peter D. Welch. The Use of Fast Fourier Transform for the Estimation of Power Spectra: A Method Based on Time Averaging Over Short, Modified Periodograms. *IEEE Transactions on Audio and Electroacoustics*, 15(2):70–73, 1967.

[107] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *ICLR*. International Conference on Learning Representations, ICLR, dec 2015.

[108] Stijn Cassenaer and Gilles Laurent. Conditional modulation of spike-timing-dependent plasticity for olfactory learning. *Nature*, 482(7383):47–51, feb 2012.

[109] Magdalena Sanhueza and John Lisman. The CaMKII/NMDAR complex as a molecular memory. *Molecular Brain*, 6(1):10, feb 2013.

[110] Wolfram Schultz. Dopamine reward prediction-error signalling: a two-component response. *Nature Reviews Neuroscience*, 17(3):183, 2016.

[111] Jean-Christophe Cassel. The role of serotonin in learning and memory: a rich pallet of experimental studies. In *Handbook of Behavioral Neuroscience*, volume 31, pages 549–570. Elsevier, 2020.

[112] Amjad H Bazzari and H Rheinallt Parri. Neuromodulators and long-term synaptic plasticity in learning and memory: A steered-glutamatergic perspective. *Brain sciences*, 9(11):300, 2019.

[113] Éva Borbély, Bálint Scheich, and Zsuzsanna Helyes. Neuropeptides in learning and memory. *Neuropeptides*, 47(6):439–450, dec 2013.

[114] Trevor J. Hamilton, Sara Xapelli, Sheldon D. Michaelson, Matthew E. Larkum, and William F. Colmers. Modulation of distal calcium electrogenesis by neuropeptide Y1 receptors inhibits neocortical long-term depression. *Journal of Neuroscience*, 33(27):11184–11193, jul 2013.

[115] Stephen J Smith. Transcriptomic evidence for dense peptidergic networks within forebrains of four widely divergent tetrapods. *Current opinion in neurobiology*, 71:100–109, 2021.

[116] Yang Dan and Mu-Ming Poo. Spike timing-dependent plasticity: from synapse to perception. *Physiological reviews*, 86(3):1033–1048, 2006.

[117] Sen Song, Kenneth D Miller, and Larry F Abbott. Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nature neuroscience*, 3(9):919–926, 2000.

[118] Marco P Lehmann, He A Xu, Vasiliki Liakoni, Michael H Herzog, Wulfram Gerstner, and Kerstin Preuschoff. One-shot learning and behavioral eligibility traces in sequential decision making. *Elife*, 8:e47463, 2019.

[119] Verena Pawlak, Jeffery R Wickens, Alfredo Kirkwood, and Jason ND Kerr. Timing is not everything: neuromodulation opens the stdp gate. *Frontiers in synaptic neuroscience*, 2:146, 2010.

[120] Johnatan Aljadeff, James D'amour, Rachel E Field, Robert C Froemke, and Claudia Clopath. Cortical credit assignment by hebbian, neuromodulatory and inhibitory plasticity. *arXiv preprint arXiv:1911.00307*, 2019.

[121] Nicolas Frémaux and Wulfram Gerstner. Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules. *Frontiers in Neural Circuits*, 9(JAN2016):85, jan 2015.

[122] Răzvan V Florian. Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity. *Neural computation*, 19(6):1468–1502, 2007.

[123] Roman Pogodin and Peter Latham. Kernelized information bottleneck leads to biologically plausible 3-factor hebbian learning in deep networks. *Advances in Neural Information Processing Systems*, 33:7296–7307, 2020.

[124] Daniel B Ehrlich, Jasmine T Stone, David Brandfonbrener, Alexander Atanasov, and John D Murray. Psychrnn: An accessible and flexible python package for training recurrent neural network models on cognitive tasks. *Eneuro*, 8(1), 2021.

[125] Guangyu Robert Yang and Manuel Molano Mazon. Next-generation of recurrent neural network models for cognition. 2021.

[126] Timothy P Lillicrap and Adam Santoro. Backpropagation through time and the brain. *Current opinion in neurobiology*, 55:82–89, 2019.

[127] Daniel Jacobs, Yuhan H Liu, Trevor Hilton, Martin Del Campo, Peter L Carlen, and Berj L Bardakjian. Classification of scalp eeg states prior to clinical seizure onset. *IEEE journal of translational engineering in health and medicine*, 7:1–3, 2019.

[128] Yuhan Liu, Vasily Grigorovsky, and Berj Bardakjian. Excitation and inhibition balance underlying epileptiform activity. *IEEE Transactions on Biomedical Engineering*, 67(9):2473–2481, 2020.

[129] Yu Liu, Ashish Khisti, and Aditya Mahajan. On privacy in smart metering systems

with periodically time-varying input distribution. In *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 513–517. IEEE, 2017.

[130] Yuhan Helena Liu, Si-Hyeon Lee, and Ashish Khisti. Information-theoretic privacy in smart metering systems using cascaded rechargeable batteries. *IEEE Signal Processing Letters*, 24(3):314–318, 2017.

[131] Yuhan Helena Liu, Stephen Smith, Stefan Mihalas, Eric Shea-Brown, and Uygar Sümbül. A solution to temporal credit assignment using cell-type-specific modulatory signals. *bioRxiv*, pages 2020–11, 2021.

[132] Yu Hu, Steven L Brunton, Nicholas Cain, Stefan Mihalas, J Nathan Kutz, and Eric Shea-Brown. Feedback through graph motifs relates structure and function in complex networks. *Physical Review E*, 98(6):062312, 2018.

[133] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. {TensorFlow}: A system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016.

[134] Luke Y Prince, Ellen Boven, Roy Henha Eyono, Arna Ghosh, Joe Pemberton, Franz Scherr, Claudia Clopath, Rui Ponte Costa, Wolfgang Maass, Blake A Richards, et al. Ccn gac workshop: Issues with learning in biological recurrent neural networks. *arXiv preprint arXiv:2105.05382*, 2021.

[135] Yiding Jiang, Pierre Foret, Scott Yak, Daniel M Roy, Hossein Mobahi, Gintare Karolina Dziugaite, Samy Bengio, Suriya Gunasekar, Isabelle Guyon, and Behnam Neyshabur. Neurips 2020 competition: Predicting generalization in deep learning. *arXiv preprint arXiv:2012.07976*, 2020.

[136] Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. *arXiv preprint arXiv:1912.02178*, 2019.

[137] Gintare Karolina Dziugaite and Daniel M Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017.

[138] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. *Advances in neural information processing systems*, 30, 2017.

[139] Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. Normalized flat minima: Exploring scale invariant definition of flat minima for neural networks using pac-bayesian analysis. In *International Conference on Machine Learning*, pages 9636–9647. PMLR, 2020.

[140] James M Murray. Local online learning in recurrent networks with random feedback. *ELife*, 8:e43299, 2019.

[141] Colin Bredenberg, Benjamin Lyo, Eero Simoncelli, and Cristina Savin. Impression learning: Online representation learning with synaptic plasticity. *Advances in Neural Information Processing Systems*, 34, 2021.

[142] Yanis Bahroun, Dmitri Chklovskii, and Anirvan Sengupta. A normative and biologically plausible algorithm for independent component analysis. *Advances in Neural Information Processing Systems*, 34, 2021.

[143] Basile Confavreux, Friedemann Zenke, Everton Agnes, Timothy Lillicrap, and Tim Vogels. A meta-learning approach to (re) discover plasticity rules that carve a desired function into a neural network. *Advances in Neural Information Processing Systems*, 33:16398–16408, 2020.

[144] Gabriel Ocker and Michael Buice. Tensor decompositions of higher-order correlations by nonlinear hebbian plasticity. *Advances in Neural Information Processing Systems*, 34, 2021.

[145] Danil Tyulmankov, Ching Fang, Annapurna Vadaparty, and Guangyu Robert Yang. Biological key-value memory networks. *Advances in Neural Information Processing Systems*, 34, 2021.

[146] Alexander Meulemans, Matilde Tristany Farinha, Javier Garcia Ordonez, Pau Vilimelis Aceituno, João Sacramento, and Benjamin F Grewe. Credit assignment in neural networks through deep feedback control. *Advances in Neural Information Processing Systems*, 34, 2021.

[147] Brandon McMahan, Michael Kleinman, and Jonathan Kao. Learning rule influences recurrent network representations but not attractor structure in decision-making tasks. *Advances in Neural Information Processing Systems*, 34, 2021.

[148] Paul Haider, Benjamin Ellenberger, Laura Kriener, Jakob Jordan, Walter Senn, and Mihai A Petrovici. Latent equilibrium: A unified learning theory for arbitrarily fast computation with arbitrarily slow neurons. *Advances in Neural Information Processing Systems*, 34:17839–17851, 2021.

[149] Roman Pogodin, Yash Mehta, Timothy Lillicrap, and Peter E Latham. Towards biologically plausible convolutional networks. *Advances in Neural Information Processing Systems*, 34:13924–13936, 2021.

[150] Johannes Friedrich, Siavash Golkar, Shiva Farashahi, Alexander Genkin, Anirvan Sengupta, and Dmitri Chklovskii. Neural optimal feedback control with local learning rules. *Advances in Neural Information Processing Systems*, 34, 2021.

[151] Colin Bredenberg, Eero Simoncelli, and Cristina Savin. Learning efficient task-dependent representations with synaptic plasticity. *Advances in Neural Information Processing Systems*, 33:15714–15724, 2020.

[152] Bernd Illing, Jean Ventura, Guillaume Bellec, and Wulfram Gerstner. Local plasticity rules can learn deep representations using self-supervised contrastive predictions. *Advances in Neural Information Processing Systems*, 34, 2021.

[153] Elias Najarro and Sebastian Risi. Meta-learning through hebbian plasticity in random networks. *Advances in Neural Information Processing Systems*, 33:20719–20731, 2020.

[154] Pablo Tano, Peter Dayan, and Alexandre Pouget. A local temporal difference code

for distributional reinforcement learning. *Advances in Neural Information Processing Systems*, 33:13662–13673, 2020.

[155] Lea Duncker, Laura Driscoll, Krishna V Shenoy, Maneesh Sahani, and David Sussillo. Organizing recurrent network dynamics by task-computation to enable continual learning. *Advances in neural information processing systems*, 33:14387–14397, 2020.

[156] Aran Nayebi, Sanjana Srivastava, Surya Ganguli, and Daniel L Yamins. Identifying learning rules from neural network observables. *Advances in Neural Information Processing Systems*, 33:2639–2650, 2020.

[157] Zoe Ashwood, Nicholas A Roy, Ji Hyun Bak, and Jonathan W Pillow. Inferring learning rules from animal decision-making. *Advances in Neural Information Processing Systems*, 33:3442–3453, 2020.

[158] Siavash Golkar, David Lipshutz, Yanis Bahroun, Anirvan Sengupta, and Dmitri Chklovskii. A simple normative network approximates local non-hebbian learning in the cortex. *Advances in neural information processing systems*, 33:7283–7295, 2020.

[159] Daniel R Kepple, Rainer Engelken, and Kanaka Rajan. Curriculum learning as a tool to uncover learning principles in the brain. In *International Conference on Learning Representations*, 2021.

[160] Danil Tyulmankov, Guangyu Robert Yang, and LF Abbott. Meta-learning synaptic plasticity and memory addressing for continual familiarity detection. *Neuron*, 110(3):544–557, 2022.

[161] Eli Moore and Rishidev Chaudhuri. Using noise to probe recurrent neural network structure and prune synapses. *Advances in Neural Information Processing Systems*, 33:14046–14057, 2020.

[162] Grace Wan Yu Ang, Clara S Tang, Y Audrey Hay, Sara Zannone, Ole Paulsen, and Claudia Clopath. The functional role of sequentially neuromodulated synaptic plasticity in behavioural learning. *PLoS Computational Biology*, 17(6):e1009017, 2021.

[163] Eren Sezener, Agnieszka Grabska-Barwińska, Dimitar Kostadinov, Maxime Beau, Sanjukta Krishnagopal, David Budden, Marcus Hutter, Joel Veness, Matthew Botvinick, Claudia Clopath, et al. A rapid and efficient learning rule for biological neural circuits. *BioRxiv*, 2021.

[164] Owen Marschall, Kyunghyun Cho, and Cristina Savin. Using local plasticity rules to train recurrent neural networks. *arXiv preprint arXiv:1905.12100*, 2019.

[165] David Clark, LF Abbott, and SueYeon Chung. Credit assignment through broadcasting a global error vector. *Advances in Neural Information Processing Systems*, 34:10053–10066, 2021.

[166] Maxence Ernoult, Fabrice Normandin, Abhinav Moudgil, Sean Spinney, Eugene Belilovsky, Irina Rish, Blake Richards, and Yoshua Bengio. Towards scaling difference target propagation by learning backprop targets. *arXiv preprint arXiv:2201.13415*, 2022.

[167] Jack Lindsey and Ashok Litwin-Kumar. Learning to learn with feedback and local plasticity. *Advances in Neural Information Processing Systems*, 33:21213–21223, 2020.

[168] Sukbin Lim, Jillian L McKee, Luke Woloszyn, Yali Amit, David J Freedman, David L Sheinberg, and Nicolas Brunel. Inferring learning rules from distributions of firing rates in cortical neurons. *Nature neuroscience*, 18(12):1804–1810, 2015.

[169] Artur Luczak, Bruce L. McNaughton, and Yoshimasa Kubo. Neurons learn by predicting future activity. *Nature Machine Intelligence*, 4(1):62–72, January 2022.

[170] James C.R. Whittington and Rafal Bogacz. Theories of Error Back-Propagation in the Brain. *Trends in Cognitive Sciences*, 2019.

[171] Matthew G Perich, Charlotte Arlt, Sofia Soares, Megan E Young, Clayton P Mosher, Juri Minxha, Eugene Carter, Ueli Rutishauser, Peter H Rudebeck, Christopher D Harvey, et al. Inferring brain-wide interactions using data-constrained recurrent neural network models. *bioRxiv*, pages 2020–12, 2021.

[172] Jimmy Smith, Scott Linderman, and David Sussillo. Reverse engineering recurrent neural networks with jacobian switching linear dynamical systems. *Advances in Neural Information Processing Systems*, 34, 2021.

[173] Michael Kleinman, Chandramouli Chandrasekaran, and Jonathan Kao. A mechanistic multi-area recurrent network model of decision-making. *Advances in Neural Information Processing Systems*, 34, 2021.

[174] Friedrich Schuessler, Francesca Mastrogiuseppe, Alexis Dubreuil, Srdjan Ostojic, and Omri Barak. The interplay between randomness and structure during learning in rnns. *Advances in neural information processing systems*, 33:13352–13362, 2020.

[175] Rylan Schaeffer, Mikail Khona, Leenoy Meshulam, Ila Rani Fiete, et al. Reverse-engineering recurrent neural network solutions to a hierarchical inference task for mice. *bioRxiv*, 2020.

[176] Guangyu Robert Yang, Madhura R Joglekar, H Francis Song, William T Newsome, and Xiao-Jing Wang. Task representations in neural networks trained to perform many cognitive tasks. *Nature neuroscience*, 22(2):297–306, 2019.

[177] Valerio Mante, David Sussillo, Krishna V Shenoy, and William T Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *nature*, 503(7474):78–84, 2013.

[178] Joshua Glaser, Matthew Whiteway, John P Cunningham, Liam Paninski, and Scott Linderman. Recurrent switching dynamical systems models for multiple interacting neural populations. *Advances in neural information processing systems*, 33:14867–14878, 2020.

[179] Jonathan Kadmon, Jonathan Timcheck, and Surya Ganguli. Predictive coding in balanced neural networks with noise, chaos and delays. *Advances in neural information processing systems*, 33:16677–16688, 2020.

[180] Jonathan Dong, Ruben Ohana, Mushegh Rafayelyan, and Florent Krzakala. Reservoir

computing meets recurrent kernels and structured transforms. *Advances in Neural Information Processing Systems*, 33:16785–16796, 2020.

[181] Elia Turner, Kabir V Dabholkar, and Omri Barak. Charting and navigating the space of solutions for recurrent neural networks. *Advances in Neural Information Processing Systems*, 34:25320–25333, 2021.

[182] Sandra Nestler, Christian Keup, David Dahmen, Matthieu Gilson, Holger Rauhut, and Moritz Helias. Unfolding recurrence by green's functions for optimized reservoir computing. *Advances in Neural Information Processing Systems*, 33:17380–17390, 2020.

[183] Robert Kim and Terrence J Sejnowski. Strong inhibitory signaling underlies stable temporal dynamics and working memory in spiking neural networks. *Nature Neuroscience*, 24(1):129–139, 2021.

[184] Adrian Valente, Srdjan Ostojic, and Jonathan Pillow. Probing the relationship between linear dynamical systems and low-rank recurrent neural network models. *arXiv preprint arXiv:2110.09804*, 2021.

[185] Tom Macpherson, Anne Churchland, Terry Sejnowski, James DiCarlo, Yukiyasu Kamitani, Hidehiko Takahashi, and Takatoshi Hikida. Natural and artificial intelligence: A brief introduction to the interplay between ai and neuroscience research. *Neural Networks*, 144:603–613, 2021.

[186] Ben Tsuda, Stefan C Pate, Kay M Tye, Hava T Siegelmann, and Terrence J Sejnowski. Neuromodulators generate multiple context-relevant behaviors in a recurrent neural network by shifting activity hypertubes. *bioRxiv*, pages 2021–05, 2022.

[187] Nuttida Rungratsameetaweemana, Robert Kim, John T Serences, and Terrence J Sejnowski. Probabilistic visual processing in humans and recurrent neural networks. *Journal of Vision*, 22(3):24–24, 2022.

[188] Brian DePasquale, Christopher J Cueva, Kanaka Rajan, G Sean Escola, and LF Abbott. full-force: A target-based method for training recurrent networks. *PloS one*, 13(2):e0191527, 2018.

[189] Christopher Langdon and Tatiana A Engel. Latent circuit inference from heterogeneous neural responses during cognitive tasks. *bioRxiv*, 2022.

[190] Timothy P Lillicrap, Daniel Cownden, Douglas B Tweed, and Colin J Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications*, 7(1):1–10, 2016.

[191] Stephen J Smith, Uygar Sümbül, Lucas T Graybuck, Forrest Collman, Sharmishtaa Seshamani, Rohan Gala, Olga Gliko, Leila Elabbady, Jeremy A Miller, Trygve E Bakken, et al. Single-cell transcriptomic evidence for dense intracortical neuropeptide networks. *Elife*, 8:e47889, 2019.

[192] Stephen J Smith, Michael Hawrylycz, Jean Rossier, and Uygar Sümbül. New light on cortical neuropeptides and synaptic network plasticity. *Current Opinion in Neurobiology*, 63:176–188, 2020.

[193] Corentin Tallec and Yann Ollivier. Unbiased online recurrent optimization. *arXiv preprint arXiv:1702.05043*, 2017.

[194] Tim Cooijmans and James Martens. On the variance of unbiased online recurrent optimization. *arXiv preprint arXiv:1902.02405*, 2019.

[195] Christopher Roth, Ingmar Kanitscheider, and Ila Fiete. Kernel rnn learning (kernl). In *International Conference on Learning Representations*, 2018.

[196] Yinan Cao, Christopher Summerfield, and Andrew Saxe. Characterizing emergent representations in a space of candidate learning rules for deep networks. *Advances in Neural Information Processing Systems*, 33:8660–8670, 2020.

[197] Ganlin Song, Ruitu Xu, and John Lafferty. Convergence and alignment of gradient descent with random backpropagation weights. *Advances in Neural Information Processing Systems*, 34, 2021.

[198] Manuela Girotti, Ioannis Mitliagkas, and Gauthier Gidel. Convergence analysis and

implicit regularization of feedback alignment for deep linear networks. *arXiv preprint arXiv:2110.10815*, 2021.

[199] Huan Xu and Shie Mannor. Robustness and generalization. *Machine learning*, 86(3):391–423, 2012.

[200] Zeyuan Allen-Zhu and Yuanzhi Li. Can sgd learn recurrent neural networks with provable generalization? *Advances in Neural Information Processing Systems*, 32, 2019.

[201] Madhu S Advani, Andrew M Saxe, and Haim Sompolinsky. High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, 132:428–446, 2020.

[202] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *arXiv preprint arXiv:1806.07572*, 2018.

[203] Valentin Thomas, Fabian Pedregosa, Bart Merriënboer, Pierre-Antoine Manzagol, Yoshua Bengio, and Nicolas Le Roux. On the interplay between noise and curvature and its effect on optimization and generalization. In *International Conference on Artificial Intelligence and Statistics*, pages 3503–3513. PMLR, 2020.

[204] Mohammad Pezeshki, Oumar Kaba, Yoshua Bengio, Aaron C Courville, Doina Precup, and Guillaume Lajoie. Gradient starvation: A learning proclivity in neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

[205] Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural computation*, 9(1):1–42, 1997.

[206] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

[207] Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-sgd:

Biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124018, 2019.

[208] Xu Sun, Zhiyuan Zhang, Xuancheng Ren, Ruixuan Luo, and Liangyou Li. Exploring the vulnerability of deep neural networks: A study of parameter corruption. *arXiv preprint arXiv:2006.05620*, 2020.

[209] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[210] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.

[211] Yu Feng and Yuhai Tu. The inverse variance–flatness relation in stochastic gradient descent is critical for finding flat minima. *Proceedings of the National Academy of Sciences*, 118(9), 2021.

[212] Roman Novak, Yasaman Bahri, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. In *International Conference on Learning Representations*, 2018.

[213] Huan Wang, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. Identifying generalization properties in neural networks. *arXiv preprint arXiv:1809.07402*, 2018.

[214] Levent Sagun, Utku Evci, V Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

[215] Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. In *International Conference on Machine Learning*, pages 2232–2241. PMLR, 2019.

[216] Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael W Mahoney. Pyhessian: Neural networks through the lens of the hessian. In *2020 IEEE international conference on big data (Big data)*, pages 581–590. IEEE, 2020.

[217] Zhenyu Liao and Michael W Mahoney. Hessian eigenspectra of more realistic nonlinear models. *Advances in Neural Information Processing Systems*, 34, 2021.

[218] Zeke Xie, Qian-Yuan Tang, Yunfeng Cai, Mingming Sun, and Ping Li. On the power-law spectrum in deep learning: A bridge to protein science. *arXiv preprint arXiv:2201.13011*, 2022.

[219] Zhewei Yao, Amir Gholami, Qi Lei, Kurt Keutzer, and Michael W Mahoney. Hessian-based analysis of large batch training and robustness to adversaries. *Advances in Neural Information Processing Systems*, 31, 2018.

[220] Zeke Xie, Issei Sato, and Masashi Sugiyama. A diffusion theory for deep learning dynamics: Stochastic gradient descent exponentially favors flat minima. *arXiv preprint arXiv:2002.03495*, 2020.

[221] Stanisław Jastrzębski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Finding flatter minima with sgd. 2018.

[222] Chiyuan Zhang, Qianli Liao, Alexander Rakhlin, Brando Miranda, Noah Golowich, and Tomaso Poggio. Theory of deep learning iib: Optimization properties of sgd. *arXiv preprint arXiv:1801.02254*, 2018.

[223] Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. *Advances in neural information processing systems*, 30, 2017.

[224] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

[225] Carlo Baldassi, Fabrizio Pittorino, and Riccardo Zecchina. Shaping the learning landscape in neural networks around wide flat minima. *Proceedings of the National Academy of Sciences*, 117(1):161–170, 2020.

[226] Jean Kaddour, Linqing Liu, Ricardo Silva, and Matt J Kusner. Questions for flat-minima optimization of modern neural networks. *arXiv preprint arXiv:2202.00661*, 2022.

[227] Yaowei Zheng, Richong Zhang, and Yongyi Mao. Regularizing neural networks via adversarial model perturbation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8156–8165, 2021.

[228] Dongxian Wu, Shu-Tao Xia, and Yisen Wang. Adversarial weight perturbation helps robust generalization. *Advances in Neural Information Processing Systems*, 33:2958–2969, 2020.

[229] Ryo Karakida, Shotaro Akaho, and Shun-ichi Amari. The normalization method for alleviating pathological sharpness in wide neural networks. *Advances in neural information processing systems*, 32, 2019.

[230] Adepu Ravi Sankar, Yash Khasbage, Rahul Vigneswaran, and Vineeth N Balasubramanian. A deeper look at the hessian eigenspectrum of deep neural networks and its applications to regularization. *arXiv preprint arXiv:2012.03801*, 2020.

[231] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pages 1019–1028. PMLR, 2017.

[232] Tengyuan Liang, Tomaso Poggio, Alexander Rakhlin, and James Stokes. Fisher-rao metric, geometry, and complexity of neural networks. In *The 22nd international conference on artificial intelligence and statistics*, pages 888–896. PMLR, 2019.

[233] Akshay Rangamani, Nam H Nguyen, Abhishek Kumar, Dzung Phan, Sang H Chin, and Trac D Tran. A scale invariant flatness measure for deep network minima. *arXiv preprint arXiv:1902.02434*, 2019.

[234] Wilten Nicola and Claudia Clopath. Supervised learning in spiking neural networks with force training. *Nature communications*, 8(1):1–15, 2017.

[235] Travis Meyer, Xue-Lian Qi, Terrence R Stanford, and Christos Constantinidis. Stimulus selectivity in dorsal and ventral prefrontal cortex after training in working memory tasks. *Journal of neuroscience*, 31(17):6266–6276, 2011.

[236] Andrew M Saxe, James L McClelland, and Surya Ganguli. A mathematical theory of semantic development in deep neural networks. *Proceedings of the National Academy of Sciences*, 116(23):11537–11546, 2019.

[237] Gauthier Gidel, Francis Bach, and Simon Lacoste-Julien. Implicit regularization of discrete gradient dynamics in linear neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[238] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[239] Zhanxing Zhu, Jingfeng Wu, Bing Yu, Lei Wu, and Jinwen Ma. The anisotropic noise in stochastic gradient descent: Its behavior of escaping from sharp minima and regularization effects. *arXiv preprint arXiv:1803.00195*, 2018.

[240] Mingwei Wei and David J Schwab. How noise affects the hessian spectrum in overparameterized neural networks. *arXiv preprint arXiv:1910.00195*, 2019.

[241] Samuel L Smith and Quoc V Le. A bayesian perspective on generalization and stochastic gradient descent. *arXiv preprint arXiv:1710.06451*, 2017.

[242] Jeremy M Cohen, Simran Kaur, Yuanzhi Li, J Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. *arXiv preprint arXiv:2103.00065*, 2021.

[243] Aitor Lewkowycz, Yasaman Bahri, Ethan Dyer, Jascha Sohl-Dickstein, and Guy Gur-Ari. The large learning rate phase of deep learning: the catapult mechanism. *arXiv preprint arXiv:2003.02218*, 2020.

[244] Stanislaw Jastrzebski, Maciej Szymczak, Stanislav Fort, Devansh Arpit, Jacek Tabor, Kyunghyun Cho, and Krzysztof Geras. The break-even point on optimization trajectories of deep neural networks. *arXiv preprint arXiv:2002.09572*, 2020.

[245] Justin Gilmer, Behrooz Ghorbani, Ankush Garg, Sneha Kudugunta, Behnam Neyshabur, David Cardoze, George Edward Dahl, Zachary Nado, and Orhan Firat. A loss curvature perspective on training instabilities of deep learning models. In *International Conference on Learning Representations*, 2021.

[246] Sanjeev Arora, Zhiyuan Li, and Abhishek Panigrahi. Understanding gradient descent on edge of stability in deep learning. *arXiv preprint arXiv:2205.09745*, 2022.

[247] Vladimir Ivanov and Konstantinos Michmizos. Increasing liquid state machine performance with edge-of-chaos dynamics organized by astrocyte-modulated plasticity. *Advances in Neural Information Processing Systems*, 34, 2021.

[248] Lukas Braun and Tim Vogels. Online learning of neural computations from sparse temporal feedback. *Advances in Neural Information Processing Systems*, 34, 2021.

[249] Joel Dapello, Jenelle Feather, Hang Le, Tiago Marques, David Cox, Josh McDermott, James J DiCarlo, and SueYeon Chung. Neural population geometry reveals the role of stochasticity in robust perception. *Advances in Neural Information Processing Systems*, 34, 2021.

[250] Maurizio De Pittà, Nicolas Brunel, and Andrea Volterra. Astrocytes: Orchestrating synaptic plasticity? *Neuroscience*, 323:43–61, 2016.

[251] James M Murray and G Sean Escola. Remembrance of things practiced with fast and slow learning in cortical and subcortical pathways. *Nature Communications*, 11(1):1–12, 2020.

[252] Anthony M Zador. A critique of pure learning and what artificial neural networks can learn from animal brains. *Nature communications*, 10(1):1–7, 2019.

[253] Dhireesha Kudithipudi, Mario Aguilar-Simon, Jonathan Babb, Maxim Bazhenov, Douglas Blackiston, Josh Bongard, Andrew P Brna, Suraj Chakravarthi Raja, Nick Cheney, Jeff Clune, et al. Biological underpinnings for lifelong learning machines. *Nature Machine Intelligence*, 4(3):196–210, 2022.

[254] Yazan N Billeh, Binghuang Cai, Sergey L Gratiy, Kael Dai, Ramakrishnan Iyer, Nathan W Gouwens, Reza Abbasi-Asl, Xiaoxuan Jia, Joshua H Siegle, Shawn R Olsen, et al. Systematic integration of structural and functional data into multi-scale models of mouse primary visual cortex. *Neuron*, 106(3):388–403, 2020.

[255] James Hazelden, Yuhan Helena Liu, Eli Shlizerman, and Eric Shea-Brown. Evolutionary algorithms as an alternative to backpropagation for supervised training of biophysical neural networks and neural odes. *arXiv preprint arXiv:2311.10869*, 2023.

[256] Victor Geadah, Stefan Horoi, Giancarlo Kerg, Guy Wolf, and Guillaume Lajoie. Goal-driven optimization of single-neuron properties in artificial networks reveals regularization role of neural diversity and adaptation. *bioRxiv*, 2022.

[257] Ilenna Simone Jones and Konrad Paul Kording. Can single neurons solve mnist? the computational power of biological dendritic trees. *arXiv preprint arXiv:2009.01269*, 2020.

[258] Nicolas Perez-Nieves, Vincent CH Leung, Pier Luigi Dragotti, and Dan FM Goodman. Neural heterogeneity promotes robust learning. *Nature communications*, 12(1):1–9, 2021.

[259] Laureline Logiaco, LF Abbott, and Sean Escola. Thalamic control of cortical dynamics in a model of flexible motor sequencing. *Cell reports*, 35(9):109090, 2021.

[260] Corinne Teeter, Ramakrishnan Iyer, Vilas Menon, Nathan Gouwens, David Feng, Jim Berg, Aaron Szafer, Nicholas Cain, Hongkui Zeng, Michael Hawrylycz, et al. Generalized leaky integrate-and-fire models classify multiple neuron types. *Nature communications*, 9(1):1–15, 2018.

[261] Chloe Winston, Dana Mastrovito, Eric Shea-Brown, and Stefan Mihalas. Heterogeneity in neuronal dynamics is learned by gradient descent for temporal processing tasks. *bioRxiv*, 2022.

[262] Christoph Stöckl, Dominik Lang, and Wolfgang Maass. Probabilistic skeletons endow brain-like neural networks with innate computing capabilities. *bioRxiv*, 2021.

[263] Yuanzhi Li, Colin Wei, and Tengyu Ma. Towards explaining the regularization effect of initial large learning rate in training neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[264] Niru Maheswaranathan, David Sussillo, Luke Metz, Ruoxi Sun, and Jascha Sohl-Dickstein. Reverse engineering learned optimizers reveals known and novel mechanisms. *Advances in Neural Information Processing Systems*, 34, 2021.

[265] Cooper D Grossman, Bilal A Bari, and Jeremiah Y Cohen. Serotonin neurons modulate learning rate through uncertainty. *Current Biology*, 2021.

[266] Antonio Orvieto, Hans Kersting, Frank Proske, Francis Bach, and Aurelien Lucchi. Anticorrelated noise injection for improved generalization. *arXiv preprint arXiv:2202.02831*, 2022.

[267] Mo Zhou, Tianyi Liu, Yan Li, Dachao Lin, Enlu Zhou, and Tuo Zhao. Toward understanding the importance of noise in training neural networks. In *International Conference on Machine Learning*, pages 7594–7602. PMLR, 2019.

[268] Levent Sagun, Leon Bottou, and Yann LeCun. Eigenvalues of the hessian in deep learning: Singularity and beyond. *arXiv preprint arXiv:1611.07476*, 2016.

[269] Lloyd N Trefethen and David Bau III. *Numerical linear algebra*, volume 50. Siam, 1997.

[270] Carsen Stringer, Marius Pachitariu, Nicholas Steinmetz, Matteo Carandini, and Kenneth D Harris. High-dimensional geometry of population responses in visual cortex. *Nature*, 571(7765):361–365, 2019.

[271] Arna Ghosh, Arnab Kumar Mondal, Kumar Krishna Agrawal, and Blake Richards. Investigating power laws in deep representation learning. *arXiv preprint arXiv:2202.05808*, 2022.

[272] Guozhang Chen, Franz Scherr, and Wolfgang Maass. Analysis of visual processing capabilities and neural coding strategies of a detailed model for laminar cortical microcircuits in mouse v1. *bioRxiv*, 2021.

[273] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

[274] Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019.

[275] Benjamin Cramer, Sebastian Billaudelle, Simeon Kanya, Aron Leibfried, Andreas Grübl, Vitali Karasenko, Christian Pehle, Korbinian Schreiber, Yannik Stradmann, Johannes Weis, et al. Surrogate gradients for analog neuromorphic computing. *Proceedings of the National Academy of Sciences*, 119(4), 2022.

[276] Dongsung Huh and Terrence J Sejnowski. Gradient descent for spiking neural networks. *Advances in neural information processing systems*, 31, 2018.

[277] Friedemann Zenke and Surya Ganguli. Superspike: Supervised learning in multilayer spiking neural networks. *Neural computation*, 30(6):1514–1541, 2018.

[278] Wenrui Zhang and Peng Li. Temporal spike sequence learning via backpropagation for deep spiking neural networks. *Advances in Neural Information Processing Systems*, 33:12022–12033, 2020.

[279] Mingqing Xiao, Qingyan Meng, Zongpeng Zhang, Yisen Wang, and Zhouchen Lin. Training feedback spiking neural networks by implicit differentiation on the equilibrium state. *Advances in Neural Information Processing Systems*, 34, 2021.

[280] Nicolas Perez-Nieves and Dan Goodman. Sparse spiking gradient descent. *Advances in Neural Information Processing Systems*, 34, 2021.

[281] Yuhang Li, Yufei Guo, Shanghang Zhang, Shikuang Deng, Yongqing Hai, and Shi Gu. Differentiable spike: Rethinking gradient-descent for training spiking neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

[282] Nicolas Skatchkovsky, Osvaldo Simeone, and Hyeryung Jang. Learning to time-decode in spiking neural networks through the information bottleneck. *arXiv preprint arXiv:2106.01177*, 2021.

[283] Guillaume Bellec, Darjan Salaj, Anand Subramoney, Robert Legenstein, and Wolfgang Maass. Long short-term memory and learning-to-learn in networks of spiking neurons. *Advances in neural information processing systems*, 31, 2018.

[284] Alessandro Achille, Matteo Rovere, and Stefano Soatto. Critical learning periods in deep networks. In *International Conference on Learning Representations*, 2018.

[285] Lukas Braun, Clémentine Dominé, James Fitzgerald, and Andrew Saxe. Exact learning dynamics of deep linear networks with prior knowledge. *Advances in Neural Information Processing Systems*, 35:6615–6629, 2022.

[286] Dhruva V Raman and Timothy O'Leary. Frozen algorithms: how the brain's wiring facilitates learning. *Current Opinion in Neurobiology*, 67:207–214, 2021.

[287] D Simard, L Nadeau, and H Kröger. Fastest learning in small-world neural networks. *Physics Letters A*, 336(1):8–15, 2005.

[288] Abdulkadir Canatar, Blake Bordelon, and Cengiz Pehlevan. Spectral bias and task-model alignment explain generalization in kernel regression and infinitely wide neural networks. *Nature communications*, 12(1):2914, 2021.

[289] Marjorie Xie, Samuel Muscinelli, Kameron Decker Harris, and Ashok Litwin-Kumar. Task-dependent optimal representations for cerebellar learning. *bioRxiv*, pages 2022–08, 2022.

[290] Vishwa Goudar, Barbara Peysakhovich, David J Freedman, Elizabeth A Buffalo, and Xiao-Jing Wang. Schema formation in a neural population subspace underlies learning-to-learn in flexible sensorimotor problem-solving. *Nature Neuroscience*, 26(5):879–890, 2023.

[291] Joanna C Chang, Matthew G Perich, Lee E Miller, Juan A Gallego, and Claudia Clopath. De novo motor learning creates structure in neural activity space that shapes adaptation. *bioRxiv*, pages 2023–05, 2023.

[292] Guangyu Robert Yang and Manuel Molano-Mazón. Towards the next generation of recurrent network models for cognitive neuroscience. *Current opinion in neurobiology*, 70:182–192, 2021.

[293] Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *Advances in Neural Information Processing Systems*, 32, 2019.

[294] Timo Flesch, Keno Juechems, Tsvetomira Dumbalska, Andrew Saxe, and Christopher Summerfield. Rich and lazy learning of task representations in brains and neural networks. *BioRxiv*, pages 2021–04, 2021.

[295] Frederic Mery and Tadeusz J Kawecki. A cost of long-term memory in drosophila. *Science*, 308(5725):1148–1148, 2005.

[296] Pierre-Yves Plaçais and Thomas Preat. To favor survival under food shortage, the brain disables costly memory. *Science*, 339(6118):440–442, 2013.

[297] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[298] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[299] Thomas George, Guillaume Lajoie, and Aristide Baratin. Lazy vs hasty: linearization in deep networks impacts learning schedule based on example difficulty. *TMLR*, 2022.

[300] Blake Woodworth, Suriya Gunasekar, Jason D Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. Kernel and rich regimes in overparametrized models. In *Conference on Learning Theory*, pages 3635–3673. PMLR, 2020.

[301] Sen Song, Per Jesper Sjöström, Markus Reigl, Sacha Nelson, and Dmitri B Chklovskii. Highly nonrandom features of synaptic connectivity in local cortical circuits. *PLoS biology*, 3(3):e68, 2005.

[302] Sina Alemohammad, Zichao Wang, Randall Balestriero, and Richard Baraniuk. The recurrent neural tangent kernel. *arXiv preprint arXiv:2006.10246*, 2020.

[303] Atish Agarwala, Fabian Pedregosa, and Jeffrey Pennington. Second-order regression models exhibit progressive sharpening to the edge of stability. *arXiv preprint arXiv:2210.04860*, 2022.

[304] Shahar Azulay, Edward Moroshko, Mor Shpigel Nacson, Blake E Woodworth, Nathan Srebro, Amir Globerson, and Daniel Soudry. On the implicit bias of initialization shape: Beyond infinitesimal mirror descent. In *International Conference on Machine Learning*, pages 468–477. PMLR, 2021.

[305] Melikasadat Emami, Mojtaba Sahraee-Ardakan, Parthe Pandit, Sundeep Rangan, and Alyson K Fletcher. Implicit bias of linear rnns. In *International Conference on Machine Learning*, pages 2982–2992. PMLR, 2021.

[306] Mario Geiger, Stefano Spigler, Arthur Jacot, and Matthieu Wyart. Disentangling feature and lazy training in deep neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(11):113301, 2020.

[307] Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. When do neural networks outperform kernel methods? *Advances in Neural Information Processing Systems*, 33:14820–14830, 2020.

[308] Jonas Paccolat, Leonardo Petrini, Mario Geiger, Kevin Tyloo, and Matthieu Wyart. Geometric compression of invariant manifolds in neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(4):044001, 2021.

[309] Mor Shpigel Nacson, Kavya Ravichandran, Nathan Srebro, and Daniel Soudry. Implicit bias of the step size in linear diagonal neural networks. In *International Conference on Machine Learning*, pages 16270–16295. PMLR, 2022.

[310] Jeff Z HaoChen, Colin Wei, Jason Lee, and Tengyu Ma. Shape matters: Understanding the implicit bias of the noise covariance. In *Conference on Learning Theory*, pages 2315–2357. PMLR, 2021.

[311] Timo Flesch, Andrew Saxe, and Christopher Summerfield. Continual task learning in natural and artificial agents. *Trends in Neurosciences*, 46(3):199–210, 2023.

[312] Blake Bordelon and Cengiz Pehlevan. The influence of learning rule on representation dynamics in wide neural networks. *arXiv preprint arXiv:2210.02157*, 2022.

[313] Arna Ghosh, Yuhan Helena Liu, Guillaume Lajoie, Konrad Kording, and Blake Aaron Richards. How gradient estimator variance and bias impact learning in neural networks. In *The Eleventh International Conference on Learning Representations*, 2023.

[314] Andrew M Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan D Tracey, and David D Cox. On the information bottleneck theory of deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124020, 2019.

[315] Matthew Farrell, Stefano Recanatesi, Timothy Moore, Guillaume Lajoie, and Eric Shea-Brown. Gradient-based learning drives robust representations in recurrent neural networks by balancing compression and expansion. *Nature Machine Intelligence*, 4(6):564–573, 2022.

[316] Vardan Papyan, XY Han, and David L Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663, 2020.

[317] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 ieee information theory workshop (itw)*, pages 1–5. IEEE, 2015.

[318] Christian Lohmann and Helmut W Kessels. The developmental stages of synaptic plasticity. *The Journal of physiology*, 592(1):13–31, 2014.

[319] Friedrich Schuessler, Francesca Mastrogiuseppe, Srdjan Ostojic, and Omri Barak. Aligned and oblique dynamics in recurrent neural networks. *arXiv preprint arXiv:2307.07654*, 2023.

[320] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[321] Yasaman Bahri, Jonathan Kadmon, Jeffrey Pennington, Sam S Schoenholz, Jascha Sohl-Dickstein, and Surya Ganguli. Statistical mechanics of deep learning. *Annual Review of Condensed Matter Physics*, 11:501–528, 2020.

[322] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[323] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[324] Raman Arora, Sanjeev Arora, Joan Bruna, Nadav Cohen, Simon Du, Rong Ge, Suriya Gunasekar, Chi Jin, Jason Lee, Tengyu Ma, et al. Theory of deep learning, 2019.

[325] Kanaka Rajan and Larry F Abbott. Eigenvalue spectra of random matrices for neural networks. *Physical review letters*, 97(18):188104, 2006.

[326] Jesper R Ipsen and Andre DH Peterson. Consequences of dale's law on the stability-

complexity relationship of random neural networks. *Physical Review E*, 101(5):052412, 2020.

[327] Isabelle D Harris, Hamish Meffin, Anthony N Burkitt, and Andre DH Peterson. Eigenvalue spectral properties of sparse random matrices obeying dale's law. *arXiv preprint arXiv:2212.01549*, 2022.

[328] David Dahmen, Stefano Recanatesi, Gabriel K Ocker, Xiaoxuan Jia, Moritz Helias, and Eric Shea-Brown. Strong coupling and local control of dimensionality across brain areas. *Biorxiv*, pages 2020–11, 2020.

[329] Johnatan Aljadeff, Merav Stern, and Tatyana Sharpee. Transition to chaos in random networks with cell-type-specific connectivity. *Physical review letters*, 114(8):088101, 2015.

[330] Vincent Thibeault, Antoine Allard, and Patrick Desrosiers. The low-rank hypothesis of complex systems. *Nature Physics*, pages 1–9, 2024.

[331] Yuxiu Shao and Srdjan Ostojic. Relating local connectivity and global dynamics in recurrent excitatory-inhibitory networks. *PLOS Computational Biology*, 19(1):e1010855, 2023.

[332] Luke Campagnola, Stephanie C Seeman, Thomas Chartrand, Lisa Kim, Alex Hoggarth, Clare Gamlin, Shinya Ito, Jessica Trinh, Pasha Davoudian, Cristina Radaelli, et al. Local connectivity and synaptic dynamics in mouse and human neocortex. *Science*, 375(6585):eabj5861, 2022.

[333] MICrONS Consortium, J Alexander Bae, Mahaly Baptiste, Caitlyn A Bishop, Agnes L Bodor, Derrick Brittain, JoAnn Buchanan, Daniel J Bumbarger, Manuel A Castro, Brendan Celii, et al. Functional connectomics spanning multiple areas of mouse visual cortex. *BioRxiv*, pages 2021–07, 2021.

[334] Sven Dorkenwald, Claire E McKellar, Thomas Macrina, Nico Kemnitz, Kisuk Lee, Ran Lu, Jingpeng Wu, Sergiy Popovych, Eric Mitchell, Barak Nehoran, et al. Flywire: online community for whole-brain connectomics. *Nature methods*, 19(1):119–128, 2022.

[335] Johan Winnubst, Erhan Bas, Tiago A Ferreira, Zhuhao Wu, Michael N Economo, Patrick Edson, Ben J Arthur, Christopher Bruns, Konrad Rokicki, David Schauder, et al. Reconstruction of 1,000 projection neurons reveals new cell types and organization of long-range connectivity in the mouse brain. *Cell*, 179(1):268–281, 2019.

[336] Louis K Scheffer, C Shan Xu, Michal Januszewski, Zhiyuan Lu, Shin-ya Takemura, Kenneth J Hayworth, Gary B Huang, Kazunori Shinomiya, Jeremy Maitlin-Shepard, Stuart Berg, et al. A connectome and analysis of the adult drosophila central brain. *Elife*, 9:e57443, 2020.

[337] Omri Barak. Recurrent neural networks as versatile tools of neuroscience research. *Current opinion in neurobiology*, 46:1–6, 2017.

[338] H Francis Song, Guangyu R Yang, and Xiao-Jing Wang. Training excitatory-inhibitory recurrent neural networks for cognitive tasks: a simple and flexible framework. *PLoS computational biology*, 12(2):e1004792, 2016.

[339] Greg Yang. Tensor programs ii: Neural tangent kernel for any architecture. *arXiv preprint arXiv:2006.14548*, 2020.

[340] Dana Mastrovito, Yuhan Helena Liu, Lukasz Kusmierz, Eric Shea-Brown, Christof Koch, and Stefan Mihalas. Transition to chaos separates learning regimes and relates to measure of consciousness in recurrent neural networks. *bioRxiv*, pages 2024–05, 2024.

[341] Allen Institute. Allen institute for brain science, 2023. `https://portal.brain-map.org/explore/connectivity/ultrastructural-connectomics/`.

[342] Liqiong Zhao, Bryce Beverlin, Theoden Netoff, and Duane Q Nykamp. Synchronization from second order network connectivity statistics. *Frontiers in computational neuroscience*, 5:28, 2011.

[343] Shahab Bakhtiari, Patrick Mineault, Timothy Lillicrap, Christopher Pack, and Blake Richards. The functional specialization of visual cortex emerges from training parallel pathways with self-supervised predictive learning. *Advances in Neural Information Processing Systems*, 34:25164–25178, 2021.

[344] Patrick T Sadtler, Kristin M Quick, Matthew D Golub, Steven M Chase, Stephen I Ryu, Elizabeth C Tyler-Kabara, Byron M Yu, and Aaron P Batista. Neural constraints on learning. *Nature*, 512(7515):423–426, 2014.

[345] Matthew D Golub, Patrick T Sadtler, Emily R Oby, Kristin M Quick, Stephen I Ryu, Elizabeth C Tyler-Kabara, Aaron P Batista, Steven M Chase, and Byron M Yu. Learning by neural reassociation. *Nature neuroscience*, 21(4):607–616, 2018.

[346] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[347] Kiran Vodrahalli, Rakesh Shivanna, Maheswaran Sathiamoorthy, Sagar Jain, and Ed H Chi. Nonlinear initialization methods for low-rank neural networks. *arXiv preprint arXiv:2202.00834*, 2022.

[348] Peter L Bartlett, Andrea Montanari, and Alexander Rakhlin. Deep learning: a statistical viewpoint. *Acta numerica*, 30:87–201, 2021.

[349] Mikail Khona, Sarthak Chandra, Joy J Ma, and Ila R Fiete. Winning the lottery with neural connectivity constraints: Faster learning across cognitive tasks with spatially constrained sparse rnns. *Neural Computation*, 35(11):1850–1869, 2023.

[350] Lechao Xiao, Jeffrey Pennington, and Samuel Schoenholz. Disentangling trainability and generalization in deep neural networks. In *International Conference on Machine Learning*, pages 10462–10472. PMLR, 2020.

[351] Mariia Seleznova and Gitta Kutyniok. Neural tangent kernel beyond the infinite-width limit: Effects of depth and initialization. In *International Conference on Machine Learning*, pages 19522–19560. PMLR, 2022.

[352] Alexis Dubreuil, Adrian Valente, Manuel Beiran, Francesca Mastrogiuseppe, and Srdjan Ostojic. The role of population structure in computations through neural dynamics. *Nature neuroscience*, 25(6):783–794, 2022.

[353] Peiran Gao, Eric Trautmann, Byron Yu, Gopal Santhanam, Stephen Ryu, Krishna Shenoy, and Surya Ganguli. A theory of multineuronal dimensionality, dynamics and measurement. *BioRxiv*, page 214262, 2017.

[354] Benjamin Scellier and Yoshua Bengio. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11:24, 2017.

[355] Sigurd Diederich and Manfred Opper. Learning of correlated patterns in spin-glass networks by local learning rules. *Physical review letters*, 58(9):949, 1987.

[356] Geoffrey Hinton. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*, 2022.

[357] Axel Laborieux and Friedemann Zenke. Holomorphic equilibrium propagation computes exact gradients through finite size oscillations. *arXiv preprint arXiv:2209.00530*, 2022.

[358] Will Greedy, Heng Wei Zhu, Joseph Pemberton, Jack Mellor, and Rui Ponte Costa. Single-phase deep learning in cortico-cortical networks. *Advances in Neural Information Processing Systems*, 35:24213–24225, 2022.

[359] Pieter R Roelfsema and Anthony Holtmaat. Control of synaptic plasticity in deep cortical networks. *Nature Reviews Neuroscience*, 19(3):166–180, 2018.

[360] Alexander Meulemans, Nicolas Zucchet, Seijin Kobayashi, Johannes Von Oswald, and João Sacramento. The least-control principle for local learning at equilibrium. *Advances in Neural Information Processing Systems*, 35:33603–33617, 2022.

[361] Owen Marschall, Kyunghyun Cho, and Cristina Savin. A unified framework of online learning algorithms for training recurrent neural networks. *The Journal of Machine Learning Research*, 21(1):5320–5353, 2020.

[362] Darjan Salaj, Anand Subramoney, Ceca Kraisnikovic, Guillaume Bellec, Robert Legenstein, and Wolfgang Maass. Spike frequency adaptation supports network computations on temporally dispersed information. *Elife*, 10:e65459, 2021.

[363] Chloe N Winston, Dana Mastrovito, Eric Shea-Brown, and Stefan Mihalas. Heterogeneity in neuronal dynamics is learned by gradient descent for temporal processing tasks. *Neural Computation*, 35(4):555–592, 2023.

[364] Roman Pogodin, Jonathan Cornford, Arna Ghosh, Gauthier Gidel, Guillaume Lajoie, and Blake Richards. Synaptic weight distributions depend on the geometry of plasticity. *arXiv preprint arXiv:2305.19394*, 2023.

[365] Guangyu Robert Yang and Xiao-Jing Wang. Artificial neural networks for neuroscientists: a primer. *Neuron*, 107(6):1048–1070, 2020.

[366] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[367] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

[368] Christof Koch, Karel Svoboda, Amy Bernard, Michele A Basso, Anne K Churchland, Adrienne L Fairhall, Peter A Groblewski, Jérôme A Lecoq, Zachary F Mainen, Mackenzie W Mathis, et al. Next-generation brain observatories. *Neuron*, 110(22):3661–3666, 2022.

[369] Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pages 35151–35174. PMLR, 2023.