Biologically-plausible backpropagation through arbitrary timespans via local neuromodulators

Yuhan Helena Liu[1,2,3,*], Stephen Smith[2,4], Stefan Mihalas[1,2,3], Eric Shea-Brown[1,2,3], and Uygar Sümbül[2,*]

[1]Department of Applied Mathematics, University of Washington, Seattle, WA, USA
[2]Allen Institute, 615 Westlake Ave N, Seattle WA, USA
[3]Computational Neuroscience Center, University of Washington, Seattle, WA, USA
[4]Department of Molecular and Cellular Physiology, Stanford University, Stanford CA, USA
[*]Correspondence: hyliu24@uw.edu, uygars@alleninstitute.org

Abstract

The spectacular successes of recurrent neural network models where key parameters are adjusted via backpropagation-based gradient descent have inspired much thought as to how biological neuronal networks might solve the corresponding synaptic credit assignment problem [1, 2, 3]. There is so far little agreement, however, as to how biological networks could implement the necessary backpropagation through time, given widely recognized constraints of biological synaptic network signaling architectures. Here, we propose that extra-synaptic diffusion of local neuromodulators such as neuropeptides may afford an effective mode of backpropagation lying within the bounds of biological plausibility. Going beyond existing temporal truncation-based gradient approximations [4, 5, 6], our approximate gradient-based update rule, ModProp, propagates credit information through *arbitrary* time steps. ModProp suggests that modulatory signals can act on receiving cells by convolving their eligibility traces via causal, time-invariant and synapse-type-specific filter taps. Our mathematical analysis of ModProp learning, together with simulation results on benchmark temporal tasks, demonstrate the advantage of ModProp over existing biologically-plausible temporal credit assignment rules. These results suggest a potential neuronal mechanism for signaling credit information related to recurrent interactions over a longer time horizon. Finally, we derive an *in-silico* implementation of ModProp that could serve as a low-complexity and causal alternative to backpropagation through time.

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, 5, 6].

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

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

postsynaptic neurons [7, 8, 9, 10, 11], combined with a third "modulatory" factor — top-down learning or "reward" signals [12, 13, 14, 15], for which dopamine is a prominent candidate [16]. 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, 5, 6, 17].

How might neural circuits account for long-term recurrent interactions in order 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 [18, 19, 20, 21]. 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 [22, 23]. 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 [24], suggesting widespread interaction between synaptic and peptidergic modulatory networks for synaptic credit assignment [25, 6]. 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 time-scales much longer than the time scales for axonal propagation of action potentials or synaptic delays [26]. 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 [23, 25]. While Ref. [6] recently introduced a framework exploiting properties of neuropeptidergic signaling for temporal credit assignment, similar to [4] and [5], 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 1 and 2). Unlike temporal truncation-based approximations [4, 5, 6], 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 4). In particular, we demonstrate an *online* learning setting where weights are updated causally and in real-time (Figure 5).
- We also derive a low-complexity *in silico* implementation of our algorithm suitable for *online learning* (Proposition 1).

2 Related works

Neuromodulatory factors in synaptic plasticity: One of the most fundamental learning rules, Hebbian plasticity, attributes lasting change in synaptic strength and memory formation to correlations of spike timing between particular presynaptic and postsynaptic neurons [27, 28]. 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 [7, 8, 10, 9, 29, 30]. Moreover, impacts of correlated spike timing must be augmented by one or more additional modulatory factors to steer weight updates toward desired outcome [31, 16, 32, 15, 33, 7, 1, 34, 35, 36, 8, 37, 38]. This is commonly known as learning with three factors. A prominent candidate for such a modulatory factor is dopamine [13]. 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 [23, 24]. Like dopamine, their cellular actions are exerted by the activation of G protein-coupled receptors (GPCRs), which can persistently modulate Hebbian synaptic plasticity [15, 39, 40]. 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 [23], which suggests a dense interplay between synaptic and peptidergic modulatory networks to shape synaptic credit assignment [25].

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 [41] and have vast computational and storage demands [42]. 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 [41, 43, 44]. In feedforward networks, plenty of biologically-plausible learning rules have been proposed and demonstrated impressive performance that rival backpropagation on many different tasks [33, 41, 14, 44, 34, 45, 46, 47, 48, 49].

For efficient online learning in RNNs, approximations to RTRL have been proposed [17, 50, 51, 52, 4, 53, 54]. For instance, a recent influential study [5] conceived how the three-factor learning framework could approximate the gradient. Among the biologically-plausible proposals [4, 5, 6], 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 [50, 51]. For further discussions on related algorithms, please refer to Appendix C.

3 Results

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 [23] or the hierarchical organization of cell types [22]. We consider a discrete-time rate-based RNN similar to the form in [56] 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 A for the neuron model.

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:

$$\frac{\mathrm{d}\,E}{\mathrm{d}\,W_{pq}}|_t = \sum_j \frac{\partial E}{\partial s_{j,t}} \frac{\mathrm{d}\,s_{j,t}}{\mathrm{d}\,W_{pq}} \tag{1}$$

$$\frac{\mathrm{d}\,s_{j,t}}{\mathrm{d}\,W_{pq}} = \frac{\partial s_{j,t}}{\partial W_{pq}} + \sum_l \frac{\partial s_{j,t}}{\partial s_{l,t-1}} \frac{\mathrm{d}\,s_{l,t-1}}{\mathrm{d}\,W_{pq}} = \frac{\partial s_{j,t}}{\partial W_{pq}} + \frac{\partial s_{j,t}}{\partial s_{j,t-1}} \frac{\mathrm{d}\,s_{j,t-1}}{\mathrm{d}\,W_{pq}} + \underbrace{\textstyle\sum_{l\neq j} W_{jl} \frac{\partial z_{l,t-1}}{\partial s_{l,t-1}} \frac{\mathrm{d}\,s_{l,t-1}}{\mathrm{d}\,W_{pq}}}_{\text{depends on all weights } W_{jl}}, \tag{2}$$

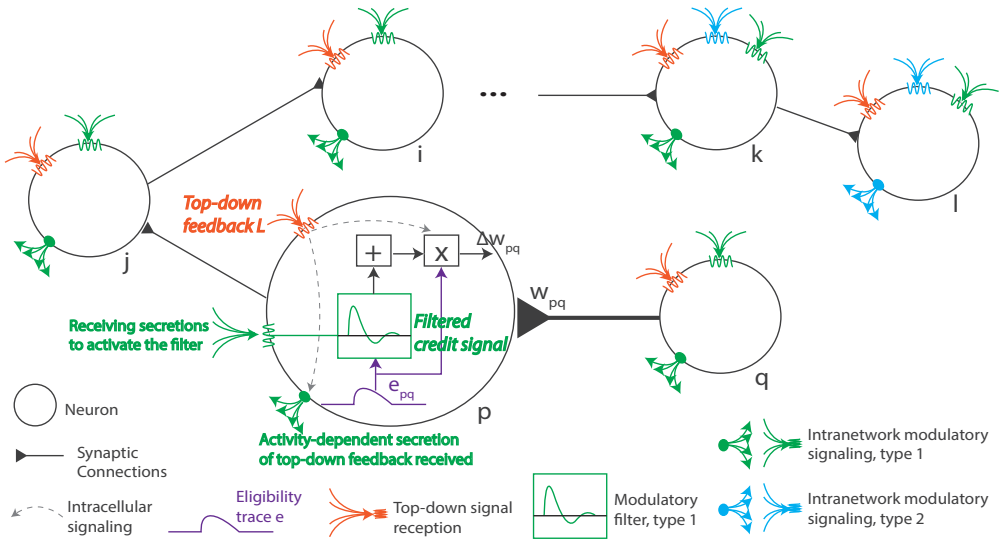

Figure 1: **Biologically-plausible temporal credit assignment via modulatory and synaptic message passing.** In addition to established biological learning ingredients (eligibility traces and a top-down learning signal [55, 8]), synapse-type-specific local modulatory networks may also be involved in weight updates [25]. 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 A and Theorem 1) predicts that the modulatory signal each neuron receives can represent a filtered credit signal regarding how its past firing (arbitrary steps back) contributes to the task outcome.

where ∂ denotes direct dependency and d accounts for all (direct and indirect) dependencies, following the notation in [5]. (See Appendix A.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. 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 [4] and [5] completely ignore the third nonlocal term (Figure 2), whereas reference [6] 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. 1 suggests two approximations to Eq. S4 and 2 to move beyond severe truncations of the exact gradient while retaining biological plausibility. **Approximation 1** replaces the activation derivative with a constant:

$$\frac{\partial z_{j,t}}{\partial s_{j,t}} \approx \mu \quad \text{(approximation 1)}, \tag{3}$$

where μ represents average activity of neurons in a ReLU network. This approximation assumes stationarity in neuron activity and uncorrelatedness of such activity with the weight, as explained in derivations leading up to Appendix Eq. S14 and Eq. S21 (Methods in Appendix A). 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 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. 6) to be *time-invariant*, a

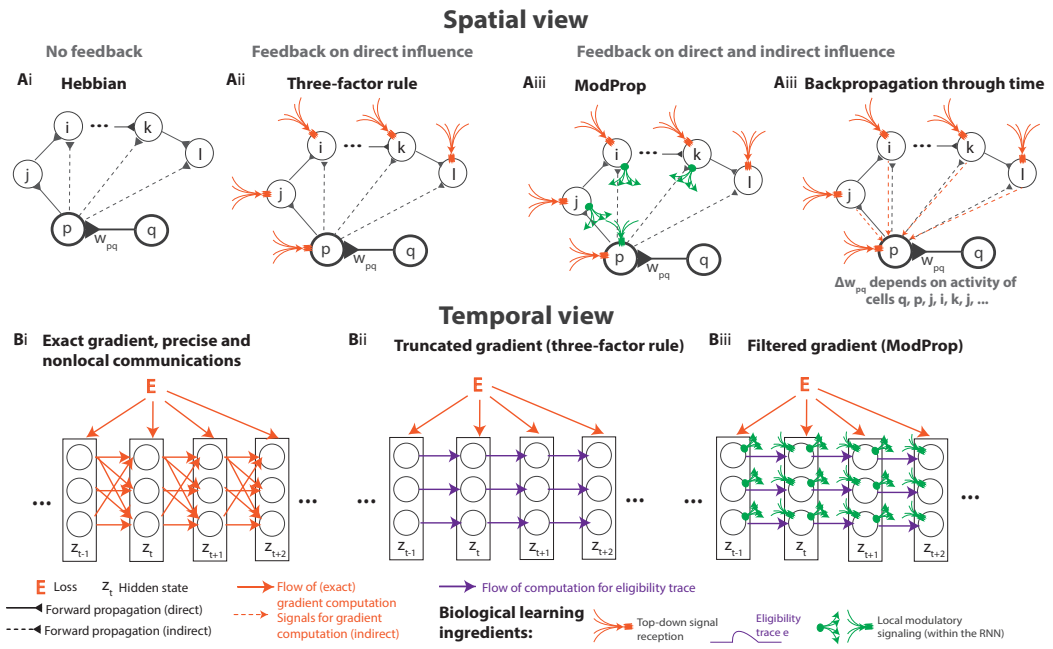

Figure 2: **Local modulatory signaling for gradient estimation.** A) A spatial view of learning rules for updating weight W_{pq}. (i) Hebbian learning, where weight update depends only on pre-/post-synaptic activities. (ii) Three-factor learning [55, 4, 5], which updates weights using additional top-down 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. 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 S1 provides a summary of approximations made by ModProp.

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:

$$\frac{\mathrm{d}\,E}{\mathrm{d}\,W_{pq}}|_t \approx \frac{\partial E}{\partial z_{p,t}} e_{pq,t} + \sum_j \frac{\partial E}{\partial s_{j,t}} \sum_{s=1}^{S} (W^s)_{jp}\, \mu^{s-1}\, e_{pq,t-s} \tag{4}$$

where $e_{pq,t} \approx \frac{dz_{p,t}}{dW_{pq}}$ ($e_{pq,t}$ is defined precisely in Appendix Eq. S24) can be interpreted as a persistent Hebbian "eligibility trace" [10, 9, 8] that keeps a fading memory of past coincidental pre- and postsynaptic activity [5]. 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 A.

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 F. 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:

$$(W^s)_{jp} \approx (W^s)_{\alpha\beta} \quad \text{(approximation 2).} \tag{5}$$

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 [23]. We call these modulatory weights synapse-type-specific (as opposed to cell-type-specific) to emphasize connectivity-based grouping. Details of how these modulatory weights were obtained can be found in Appendix A.

ModProp: filtering credit signals via local neuromodulation: Substituting **Approximation 1** and **Approximation 2** into Eq. S4 and 2 leads to the ModProp update:

$$\Delta W_{pq}|_{ModProp} \propto L_p \times e_{pq} + \left(\sum_{\alpha\in C} \left(\sum_{j\in\alpha} \mathrm{L}_j \text{ activity}_j \right) \times F_{\alpha\beta} \right) * \mathrm{e}_{pq},$$
$$F_{\alpha\beta,s} = \mu^{s-1}(W^s)_{\alpha\beta}, \tag{6}$$

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 S2 shows successful learning using fixed modulatory weights.) Details of a biological interpretation of Eq. 6 can be found in Appendix D.1. Briefly, 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) [23, 57], 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 [7, 8] – but also with local modulatory pathways through convolution (Figure 1). This modulatory mechanism allows the propagation of credit signal through arbitrary number of recurrent steps.

3.2 Properties of ModProp

Remark. *The biologically plausible implementation (Eq. 6) of ModProp complexity scales as $O(SN^2)$ per time step t.*

Here, N and S denote the number of recurrent units and number of filter taps. As seen in Eq. 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 D.1.

We next show through Theorem 1 that learning with synapse-type-specific weights leads to reduction in loss 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 S6 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 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 term 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 E.)*

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 Simulation results

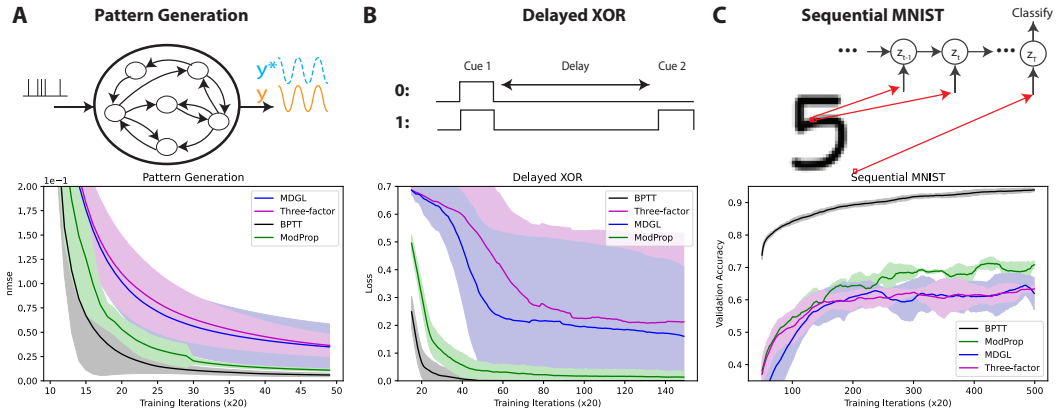

Figure 3: **Modulatory signaling of credit information on long-term recurrent interactions can improve learning outcomes.** ModProp improves the learning performance over existing bio-plausible rules. This experiment examines the performance due to Approximation 1 (Eq. 3) before any cell-type approximation of modulatory weights (Eq. 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 [58]. Note that this task is unlikely to be solved effectively by humans. (See text.) Consistent with the original MDGL paper [6], 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 [6] and e-prop [5] (labelled 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 should be identical in terms of performance. Here, we chose BPTT due to computational efficiency. Consistent with [6], we also found MDGL to confer little advantage over e-prop when all neurons are connected to the readout (Figure 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 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 [6]. Moreover, communicating longer indirect effects, despite being filtered, leads to improved learning outcomes, as demonstrated by the superior performance of ModProp over MDGL.

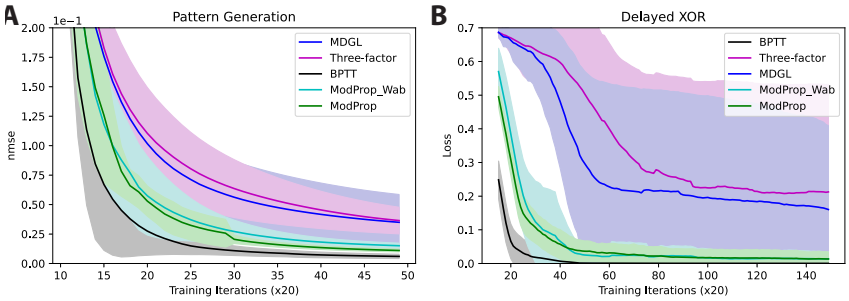

Figure 4: **Efficient learning with type-specific modulatory weights.** In addition to Approximation 1 studied in Figure 3, this figure investigates the effect of Approximation 2 (labeled as ModProp_Wab), which uses type-specific, rather than synapse-specific feedback weights for signaling credit information (Eq. 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.

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 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 S4, 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 [58] 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 3C illustrates the learning curves for this task. While the performance ordering of learning rules is still the same as previous tasks, we observe a wider gap between ModProp and BPTT. Since the time-invariant filter approximation (Eq 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). Next, in Figure 4, we investigate the learning performance using type-specific rather than synapse-specific feedback modulatory weights (Eq. 5). These type-specific weights were calculated using weight averages as in [6]. Later, we repeat these simulations using fixed random modulatory weights in Appendix Figure S2; we also demonstrate the superior performance of ModProp with fixed random modulatory weights for the "copy task" [59] in Appendix Figure S5. 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.

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 5). For that, we also derive a cost and storage efficient *in silico* implementation of ModProp in Proposition 1.

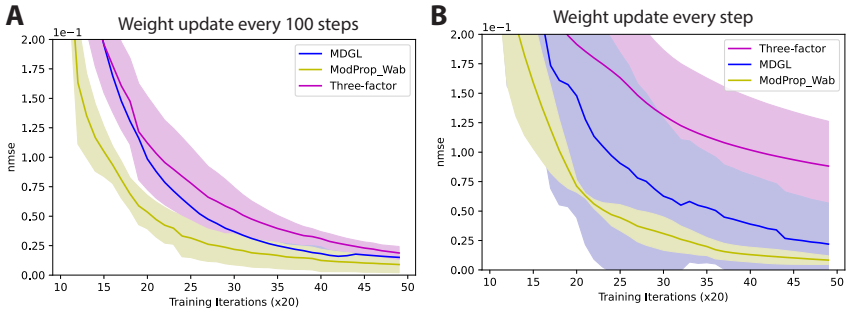

Figure 5: **Superior performance of ModProp in an online learning setting**. We investigate an online learning version of the pattern generation task, where weights are updated either A) every 100 time steps or B) every single step. Here, ModProp uses the efficient online learning implementation derived in Proposition 1. Plotting conventions follow those of previous figures.

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 D.2.)*

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 [3]. Motivated by recent genomic evidence on the widespread presence of local modulatory networks [23, 24], 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 [1] can be delivered in neuronal circuits. Instead of the severe temporal truncations of the gradient information proposed by the state-of-the-art [4, 5, 6], 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 [25, 60]. 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 [44] 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 S1 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 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 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 [61]. An interesting line of

research attends to the issue of extracting relevant, rather than full, memories [61, 62]. Investigating how our approximations could potentially be combined with memory sparsification techniques to perform very long-term biologically-plausible credit assignment can be fruitful [61].

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 towards curing certain diseases of the brain. Nevertheless, in the same vein, such future tools could also enable abuse if left unregulated. On the machine learning side, our method can be considered as 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.

Acknowledgements

We thank the Allen Institute founder, Paul G Allen, for his vision, encouragement and support. YHL is supported by the NSERC PGS-D program. This work was facilitated through the use of the UW Hyak supercomputer system.

References

[1] Timothy P Lillicrap, Adam Santoro, Luke Marris, Colin J Akerman, and Geoffrey Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, pages 1–12, 2020.

[2] 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.

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

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

[5] 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.

[6] 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.

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

[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] 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.

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

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

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

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

[14] 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.

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

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

[17] 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.

[18] 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.

[19] 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.

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

[21] 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.

[22] 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.

[23] 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.

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

[25] 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.

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

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

[28] 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.

[29] 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.

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

[31] 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.

[32] 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.

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

[34] Alexandre Payeur, Jordan Guerguiev, Friedemann Zenke, Blake A Richards, and Richard Naud. Burst-dependent synaptic plasticity can coordinate learning in hierarchical circuits. *Nature neuroscience*, pages 1–10, 2021.

[35] 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.

[36] 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.

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

[38] 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.

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

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

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

[42] 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.

[43] 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.

[44] 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.

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

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

[47] 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.

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

[49] 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.

[50] 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.

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

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

[53] 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.

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

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

[56] 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.

[57] 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.

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

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

[60] 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.

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

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

[63] 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.

[64] Yuhan Helena Liu, Arna Ghosh, Blake A Richards, Eric Shea-Brown, and Guillaume Lajoie. Beyond accuracy: generalization properties of bio-plausible temporal credit assignment rules. *arXiv preprint arXiv:2206.00823*, 2022.

[65] 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.

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

[67] 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.

[68] 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.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] We have supported our key contribution points (see Introduction) with simulations and mathematical proofs.
   (b) Did you describe the limitations of your work? [Yes] Please see discussion on future direction in the Discussion section.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Please see the Discussion section.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] We have carefully read the ethics review guidelines and confirm that this paper conforms to them.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] We have made sure that all assumptions are precisely stated in the Theorem statement. Details for where these assumptions come into the proof can be found in Appendix E.
   (b) Did you include complete proofs of all theoretical results? [Yes] We have provided complete and detailed proofs in Appendix E.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Anonymized code link is provided in Appendix F.

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Please see Appendix F.

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We ran experiments across random seeds and reported uncertainty for all applicable figures. We explained the plotting convention in the caption.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Please see Appendix F.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

   (a) If your work uses existing assets, did you cite the creators? [Yes] Please see Appendix F.

   (b) Did you mention the license of the assets? [Yes] Please see Appendix F.

   (c) Did you include any new assets either in the supplemental material or as a URL? [No]

   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Methods

A.1 Network Model

We consider a discrete-time implementation of a rate-based recurrent neural network (RNN) similar to the form in [56]. 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

$$\begin{aligned} s_{j,t+1} &= \eta\, s_{j,t} + (1-\eta)\left(\sum_{l\neq j} W_{jl}\, z_{l,t} + \sum_{p} W^{\text{IN}}_{jm}\, x_{m,t+1}\right) \\ z_{j,t} &= ReLU(s_{j,t}), \end{aligned} \tag{S1}$$

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 mth 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

$$y_{k,t} = \sum_j W^{\text{OUT}}_{kj} z_{j,t} + b^{\text{OUT}}_k, \tag{S2}$$

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:

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

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

A.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. S1, 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. S8.

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) [42], 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

$$\frac{\mathrm{d}\,E}{\mathrm{d}\,W_{pq}}|_t = \sum_j \frac{\partial E}{\partial s_{j,t}} \frac{\mathrm{d}\,s_{j,t}}{\mathrm{d}\,W_{pq}} \tag{S4}$$

$$\frac{\mathrm{d}\,z_{j,t}}{\mathrm{d}\,W_{pq}} = h_{j,t} \frac{\mathrm{d}\,s_{j,t}}{\mathrm{d}\,W_{pq}}, \text{where } h_{j,t} := \frac{\partial z_{j,t}}{\partial s_{j,t}} \tag{S5}$$

$$\frac{\partial s_{j,t}}{\partial W_{pq}} = \delta_{jp}(1-\eta) z_{q,t-1} \tag{S6}$$

$$\frac{\partial s_{j,t}}{\partial s_{l,t-1}} = \begin{cases} \eta, & j = l \\ \frac{\partial s_{j,t}}{\partial z_{l,t-1}} \frac{\partial z_{l,t-1}}{\partial s_{l,t-1}} = (1-\eta) W_{jl} h_{l,t-1}, & j \neq l \end{cases} \tag{S7}$$

$$\begin{aligned} \frac{\mathrm{d}\,s_{j,t}}{\mathrm{d}\,W_{pq}} &= \frac{\partial s_{j,t}}{\partial W_{pq}} + \sum_l \frac{\partial s_{j,t}}{\partial s_{l,t-1}} \frac{\mathrm{d}\,s_{l,t-1}}{\mathrm{d}\,W_{pq}} \\ &= \frac{\partial s_{j,t}}{\partial W_{pq}} + \eta \frac{\mathrm{d}\,s_{j,t-1}}{\mathrm{d}\,W_{pq}} + (1-\eta) \underbrace{\textstyle\sum_{l \neq j} W_{jl} \frac{\partial z_{l,t-1}}{\partial s_{l,t-1}} \frac{\mathrm{d}\,s_{l,t-1}}{\mathrm{d}\,W_{pq}}}_{\text{depends on all weights } W_{jl}}. \end{aligned} \tag{S8}$$

following the derivative notation explained above. The factor $\frac{\partial E}{\partial z_{j,t}}$ in Eq. S4 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 [5]. 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 [4] and [5] 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.

A.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 [5]. 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. S8 into Eq. S4 and repeatedly expand the expensive $\frac{\mathrm{d}\,s}{\mathrm{d}\,W}$ factor using Eq. S8:

$$\frac{\mathrm{d}\,E}{\mathrm{d}\,W_{pq}}|_t = \sum_j \frac{\partial E}{\partial z_{j,t}} h_{j,t} \left(\delta_{jp} z_{q,t-1} + \sum_l W_{jl} h_{l,t-1} \frac{\mathrm{d}\,s_{l,t-1}}{\mathrm{d}\,W_{pq}} \right) \tag{S9}$$

$$= \frac{\partial E}{\partial z_{p,t}} h_{p,t} z_{q,t-1} + \sum_j \frac{\partial E}{\partial z_{j,t}} h_{j,t} \sum_l W_{jl} h_{l,t-1} \frac{\mathrm{d}\,s_{l,t-1}}{\mathrm{d}\,W_{pq}} \tag{S10}$$

$$= \frac{\partial E}{\partial z_{p,t}} e_{pq,t} + \sum_j \frac{\partial E}{\partial z_{j,t}} h_{j,t} \sum_l W_{jl} h_{l,t-1} \left(\delta_{lp} z_{q,t-2} + \sum_k W_{lk} h_{k,t-2} \frac{\mathrm{d}\,s_{k,t-2}}{\mathrm{d}\,W_{pq}} \right) \tag{S11}$$

$$= \dots$$

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

$$\sum_j \frac{\partial E}{\partial z_{j,t}} h_{j,t} \sum_{s=2}^{S} \sum_{i_1,\ldots,i_{s-1}} W_{ji_1} W_{i_1 i_2} \ldots W_{i_{s-1}p} h_{i_1,t-1} \ldots h_{i_{s-1},t-s+1} e_{pq,t-s} \tag{S12}$$

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

$$+ \sum_j \frac{\partial E}{\partial z_{j,t}} h_{j,t} \sum_{s=2}^{S} (W^s)_{jp}\, e_{pq,t-s} \frac{1}{N^{s-1}} \sum_{i_1,\ldots,i_{s-1}} h_{i_1,t-1} h_{i_2,t-2} \ldots h_{i_{s-1},t-s+1} \tag{S13}$$

$$= \frac{\partial E}{\partial z_{p,t}} e_{pq,t} + \sum_j \frac{\partial E}{\partial z_{j,t}} h_{j,t} \sum_{s=1}^{S} (W^s)_{jp}\, H(t,s)\, e_{pq,t-s}, \tag{S14}$$

where $H(t,1) = 1$, $H(t,s) = \frac{1}{N^{s-1}} \sum_{i_1,\ldots,i_{s-1}} h_{i_1,t-1} h_{i_2,t-2} \ldots h_{i_{s-1},t-s+1}$ for $s = 1, \ldots, S$ 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

$$\prod_{\phi=1}^{l} W_{i_\phi i_{\phi+1}}, \tag{S15}$$

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

$$\prod_{\theta=1}^{l'} h_{j_\theta, t-\theta}, \tag{S16}$$

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

$$\mathbb{E}_{i_1,\ldots,i_{s-1}} W_{i_1,\ldots,i_{s-1}} h_{i_1,\ldots,i_{s-1}} = \mathbb{E}_{i_1,\ldots,i_{s-1}} W_{i_1,\ldots,i_{s-1}} \mathbb{E}_{i_1,\ldots,i_{s-1}} h_{i_1,\ldots,i_{s-1}}. \tag{S17}$$

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

$$\sum_{i_1,\ldots,i_{s-1}} W_{i_1,\ldots,i_{s-1}} h_{i_1,\ldots,i_{s-1}} \sim \mathcal{N}(N^{s-1}\mathbb{E}[W_{i_1,\ldots,i_{s-1}} h_{i_1,\ldots,i_{s-1}}], N^{s-1}\mathrm{Var}(W_{i_1,\ldots,i_{s-1}} h_{i_1,\ldots,i_{s-1}})) \tag{S18}$$

as the sum tends to infinity. Here, we simply use the i.i.d. assumption even though stronger version 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:

$$\sum_{i_1,\ldots,i_{s-1}} W_{i_1,\ldots,i_{s-1}} h_{i_1,\ldots,i_{s-1}} \approx N^{s-1}\mathbb{E} W_{i_1,\ldots,i_{s-1}} h_{i_1,\ldots,i_{s-1}} = N^{s-1}\mathbb{E} W_{i_1,\ldots,i_{s-1}} \mathbb{E} h_{i_1,\ldots,i_{s-1}}. \tag{S19}$$

Since $\mathbb{E} W_{i_1,\ldots,i_{s-1}} = \frac{1}{N^{s-1}} (W^s)_{jp}$ (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

$$\sum_{i_1,\ldots,i_{s-1}} W_{i_1,\ldots,i_{s-1}} h_{i_1,\ldots,i_{s-1}} \approx (W^s)_{jp} \frac{1}{N^{s-1}} \sum_{i_1,\ldots,i_{s-1}} h_{i_1,\ldots,i_{s-1}}, \tag{S20}$$

where h-chain with their empirical estimates.

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-indicies 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. S14). 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

$$\begin{aligned} H(t,s) &= \frac{1}{N^{s-1}} \sum_{i_1,\ldots,i_{s-1}} h_{i_1,t-1} \ldots h_{i_{s-1},t-s+1} \\ &\overset{(a)}{=} \frac{1}{N}\sum_{i_1} h_{i_1,t-1} \frac{1}{N}\sum_{i_2} h_{i_2,t-2} \cdots \frac{1}{N}\sum_{i_{s-1}} h_{i_{s-1},t-s+1} \\ &\approx \mu^{s-1}, \end{aligned} \tag{S21}$$

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 [26]. 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. S14, the approximated gradient becomes:

$$\frac{\mathrm{d}\,E}{\mathrm{d}\,W_{pq}}|_t \approx \frac{\partial E}{\partial z_{p,t}} e_{pq,t} + \sum_{\alpha\in C} \left(\sum_{j\in\alpha} \frac{\partial E}{\partial z_{j,t}} h_{j,t} \right) \sum_{s=1}^{S} (W^s)_{\alpha\beta}\, H(s)\, e_{pq,t-s}, \tag{S22}$$

and this leads to the ModProp update:

$$\begin{aligned} \Delta W_{pq}|_{ModProp} &\propto L_p \times e_{pq} + (\sum_{\alpha\in C} \left(\sum_{j\in\alpha} \mathrm{L}_j h_j \right) \times F_{\alpha\beta}) * \mathrm{e}_{pq}, \\ F_{\alpha\beta,s} &= \mu^{s-1} (W^s)_{\alpha\beta}, \end{aligned} \tag{S23}$$

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 S2. 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 $e_{pq,t}$:

$$e_{pq,t} := \frac{\partial z_{p,t}}{\partial s_{p,t}} \epsilon_{pq,t}, \tag{S24}$$

$$\epsilon_{pq,t} = \frac{\partial s_{p,t}}{\partial w_{pq}} + \frac{\partial s_{p,t}}{\partial s_{p,t-1}} \epsilon_{pq,t-1}, \tag{S25}$$

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. S1). Reference [5] provides a comprehensive discussion on how eligibility traces can be interpreted as derivatives.

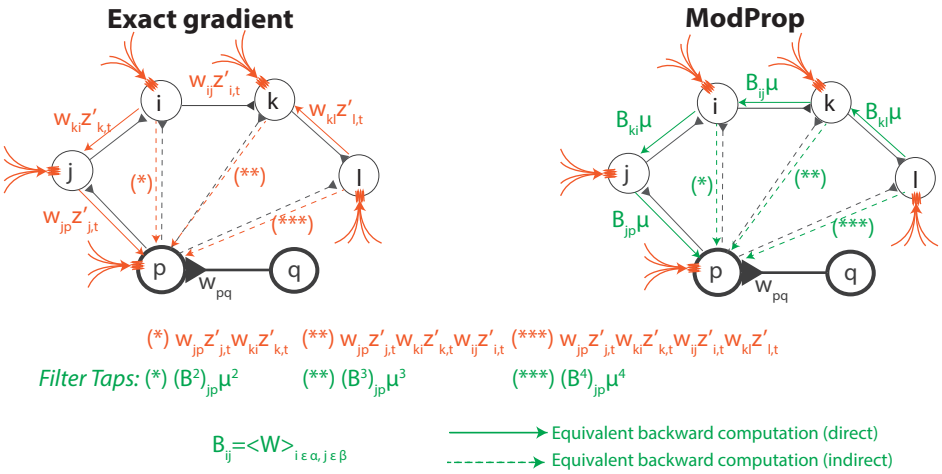

Figure S1: **ModProp brings two approximations to the nonlocal gradient terms.** First, ModProp uses type-specific feedback weights (modulatory weights), rather than cell-specific feedback weights (Eq. 3). Second, ModProp approximates the activation derivative with a constant (Eq. 5). Similar to feedback alignment for feedforward networks [44], different weights are used for the feedback path than the feedforward computation due to the well-known weight transport problem. On top of that, time-invariant activation derivative approximation was applied for time-invariant filtering. Unspecified symbols were defined in Figure 2.

B Additional simulations

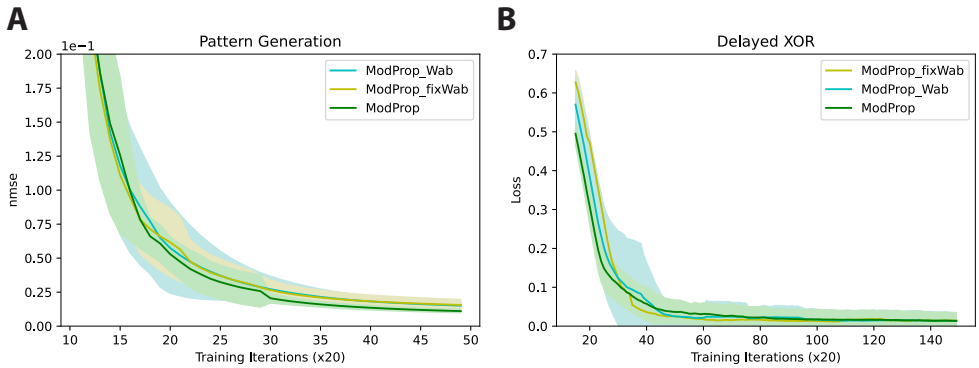

Figure S2: **Effective learning is achievable with *fixed* random synapse-type-specific modulatory weights.** Figure 4 computes type-specific modulatory weights by averaging forward weight entries in the corresponding pre- and postsynaptic cell group. This assumes that modulatory weights co-adapt with 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.

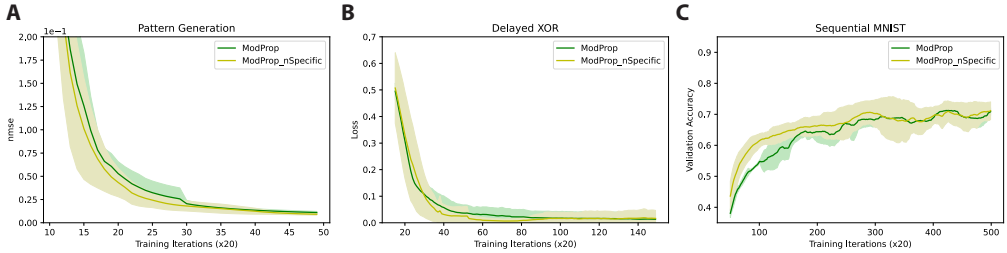

Figure S3: **Restoring neuron specificity in the activation derivative does not lead to significant improvements.** Here, ModProp_global is the basic form of ModProp investigated in the main text, where the activation derivative exhibited no spatiotemporal specificity. ModProp_nSpecific (Eq. S26) takes 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 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. S12 as

$$\begin{aligned}&\sum_{i_1,\ldots,i_{s-1}} W_{ji_1}h_{i_1,t-1}W_{i_1i_2}h_{i_2,t-2}\ldots W_{i_{s-1}p}h_{i_{s-1},t-s+1}\\ &\approx W_{ji_1}\overline{h_{i_1}}W_{i_1i_2}\overline{h_{i_2}}\ldots W_{i_{s-1}p}\overline{h_{i_{s-1}}} = (\overline{W})^s_{jp},\end{aligned} \tag{S26}$$

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

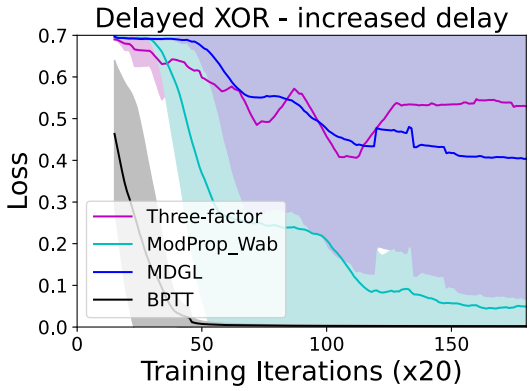

Figure S4: **Delayed XOR task with a longer delay period.** We simulate the delayed XOR task with 1.5 times the delay period used in Figure 3B. Although ModProp (with cell-type approximation) still outperforms other bio-plausible learning rules, the performance degrades (compared to ModProp_Wab in Figure 4B). Moreover, we found all rules (including BPTT) struggle to learn if we increased the delay period to twice of that in Figure 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.

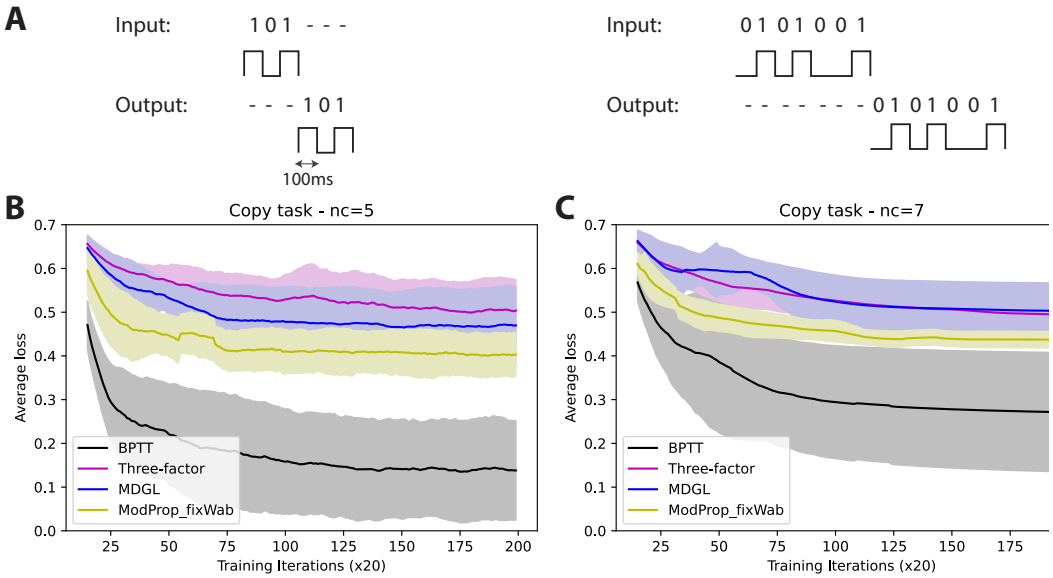

Figure S5: **Copy task with *fixed* random synapse-type-specific modulatory weights.** Sequences of binary cues are presented to an RNN. For each sequence, once the full sequence has been presented, the network should output the original sequence (with the same value and duration) without any further information [59]. **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.

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 shows that this restoration of neuron-specificity in activation derivative did not lead to significant performance improvement.

C Further discussion on related algorithms

For efficient online learning in RNNs, approximations to RTRL have been proposed [17, 50, 51, 52, 4, 53, 54]. For biological realism (use only local information for local updates) and reduced computational cost, **e-prop** [5] and **RFLO** [4] 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** [6, 63] 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 [6] 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 [64], reference [53] 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. S8) 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** [52] 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. 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 S2).

D Cost analysis and biological implementation

D.1 Cost and interpretation for biologically-plausible implementation of ModProp in Eq. 6

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

$$\Delta W_{pq} \propto \text{ET}_{pq} \times \text{TD}_p + \text{ET}_{pq} * \sum_{\alpha \in C} \text{LM}_{\alpha\beta}$$

$$\text{ET}_{pq} * \sum_{\alpha \in C} \text{LM}_{\alpha\beta} = \sum_{s=1}^{S} \text{ET}_{pq,t-s} \times \sum_{\alpha \in C} \text{LM}_{\alpha\beta,s}$$

$$\text{LM}_{\alpha\beta,s} = (\text{affinity } W^s_{\alpha\beta}) \times \sum_{j \in \alpha} \underbrace{\text{TD}_j \ \times \ (\text{activity } j)}_{\text{modulatory signal } j}. \tag{S27}$$

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 3.2:

- $a_{j,t} = \text{TD}_j \ \times \ (\text{activity } j)$ 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.
- $\sum_{s=1}^{S} \text{ET}_{pq,t-s} \times \text{LM}_{\beta,s}$ 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 $e_{pq,t-S}, \ldots e_{pq,t}$ for every (pq), leading to **a storage cost of** $O(SN^2)$.

D.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:

$$[(W^{s+1})_{\alpha\beta}] = \sum_{\gamma}[(W^1)_{\alpha\gamma}][(W^s)_{\gamma\beta}] = [\sum_{\gamma} N_\gamma (W^s)_{\alpha\gamma}(W^s)_{\gamma\beta}]$$
$$\rightarrow (W^{s+1})_{\gamma\beta} = \sum_{\gamma} N_\gamma (W^1)_{\alpha\gamma}(W^s)_{\gamma\beta}. \tag{S28}$$

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

$$\begin{aligned}
G^{t+1}_{\alpha,pq} &= \sum_{s=1}^{t+1}(W^s)_{\alpha\beta}\,\mu^{s-1}\,e_{pq,t+1-s} \\
&= \sum_{s=0}^{t}(W^{s+1})_{\alpha\beta}\,\mu^{s}\,e_{pq,t-s} \\
&= \sum_{s=1}^{t}(W^{s+1})_{\alpha\beta}\,\mu^{s}\,e_{pq,t-s} + (W^1)_{\alpha\beta}e_{pq,t} \\
&= \sum_{s=1}^{t}(\sum_{\gamma} N_\gamma (W^1)_{\alpha\gamma}(W^s)_{\gamma\beta})\,\mu^{s}\,e_{pq,t-s} + (W^1)_{\alpha\beta}e_{pq,t} \\
&= \sum_{\gamma} N_\gamma (W^1)_{\alpha\gamma}\mu \sum_{s=1}^{t}(W^s)_{\gamma\beta}\,\mu^{s-1}\,e_{pq,t-s} + (W^1)_{\alpha\beta}e_{pq,t} \\
&= \sum_{\gamma} N_\gamma (W^1)_{\alpha\gamma}\mu G^{t}_{\gamma,pq} + (W^1)_{\alpha\beta}e_{pq,t}
\end{aligned} \tag{S29}$$

And the overall update is:

$$\Delta W_{pq}|_{ModProp} = \frac{\partial E}{\partial z_{p,t}} e_{pq,t} + \sum_{\alpha} G^{t}_{\alpha,pq} \sum_{j\in\alpha} \frac{\partial E}{\partial z_{j,t}} h_{j,t} \tag{S30}$$

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 D). 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) [53] 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.

□

E 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 $s + 1$:

$$\begin{aligned}
\mathbb{E}[(\epsilon_{s+1})_{ij}] &= \mathbb{E}[(W^{s+1})_{ij} - (W^{s+1})_{\alpha\beta}] \\
&= \mathbb{E}\left[\sum_k (W^s)_{ik} W_{kj}\right] - \sum_\gamma N_\gamma (W^s)_{\alpha\gamma} W_{\gamma\beta} \\
&= \mathbb{E}\left[\sum_\gamma \sum_{k\in\gamma} ((W^s)_{\alpha\gamma} + (\epsilon_s)_{ik})(W_{\gamma\beta} + \epsilon_{kj})\right] - \sum_\gamma N_\gamma (W^s)_{\alpha\gamma} W_{\gamma\beta} \\
&= \sum_\gamma N_\gamma (W^s)_{\alpha\gamma} W_{\gamma\beta} + \sum_\gamma \sum_{k\in\gamma} \mathbb{E}[(\epsilon_s)_{ik}] W_{\gamma\beta} + \sum_\gamma \sum_{k\in\gamma} (W^s)_{\alpha\gamma} \mathbb{E}[\epsilon_{kj}] \\
&\quad + \sum_\gamma \sum_{k\in\gamma} \mathbb{E}[(\epsilon_s)_{ik}\epsilon_{kj}] - \sum_\gamma N_\gamma (W^s)_{\alpha\gamma} W_{\gamma\beta} \\
&= \sum_\gamma \sum_{k\in\gamma} \mathbb{E}[(\epsilon_s)_{ik}] W_{\gamma\beta} + \sum_\gamma \sum_{k\in\gamma} (W^s)_{\alpha\gamma} \mathbb{E}[\epsilon_{kj}] + \sum_\gamma \sum_{k\in\gamma} \mathbb{E}[(\epsilon_s)_{ik}] \mathbb{E}[\epsilon_{kj}] \\
&= 0.
\end{aligned} \tag{S31}$$

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:

$$\begin{aligned}
\mathbb{E}\left[\Delta E|_{pq,t}\right] &= -\eta \mathbb{E}\left[\widehat{\frac{dE}{dW_{pq}}} \frac{dE}{dW_{pq}}\right] \\
&= -\eta \mathbb{E}\Bigg[(y_t - y_t^*)^2 \Bigg[W_p^{\text{OUT}} e_{pq,t} + \sum_{s,\alpha} \sum_{j\in\alpha} W_j^{\text{OUT}} (W^s)_{\alpha\beta} e_{pq,t-s}\Bigg] \\
&\qquad \times \Bigg[W_p^{\text{OUT}} e_{pq,t} + \sum_{u,\alpha'} \sum_{j'\in\alpha'} W_{j'}^{\text{OUT}} [(W^u)_{\alpha'\beta} + (\epsilon_u)_{j'p}] e_{pq,t-u}\Bigg]\Bigg] \\
&= -\mathbb{E}[\Gamma_{pq}^2] - \eta \mathbb{E}\Bigg[(y_t - y_t^*)^2 \Bigg[W_p^{\text{OUT}} e_{pq,t} + \sum_{s,\alpha} \sum_{j\in\alpha} W_j^{\text{OUT}} (W^s)_{\alpha\beta} e_{pq,t-s}\Bigg] \\
&\qquad \times \Bigg[\sum_{u,\alpha'} \sum_{j'\in\alpha'} W_{j'}^{\text{OUT}} (\epsilon_u)_{j'p} e_{pq,t-u}\Bigg]\Bigg] \\
&= -\mathbb{E}[\Gamma_{pq}^2] - \eta \sum_{u,\alpha'} \sum_{j'\in\alpha'} W_{j'}^{\text{OUT}} W_p^{\text{OUT}} \mathbb{E}\left[(\epsilon_u)_{j'p} (y_t - y_t^*)^2 e_{pq,t} e_{pq,t-u}\right] \\
&\qquad - \eta \sum_{s,u,\alpha,\alpha'} \sum_{j\in\alpha, j'\in\alpha'} (W^s)_{\alpha\beta} W_j^{\text{OUT}} W_{j'}^{\text{OUT}} \mathbb{E}\left[(\epsilon_u)_{j'p} (y_t - y_t^*)^2 e_{pq,t-s} e_{pq,t-u}\right] \\
&\overset{(a)}{=} -\mathbb{E}[\Gamma_{pq}^2] - \eta \sum_{u,\alpha'} \sum_{j'\in\alpha'} W_{j'}^{\text{OUT}} W_p^{\text{OUT}} \mathbb{E}\left[(\epsilon_u)_{j'p}\right] \mathbb{E}\left[(y_t - y_t^*)^2 e_{pq,t} e_{pq,t-u}\right] \\
&\qquad - \eta \sum_{s,u,\alpha,\alpha'} \sum_{j\in\alpha, j'\in\alpha'} (W^s)_{\alpha\beta} W_j^{\text{OUT}} W_{j'}^{\text{OUT}} \mathbb{E}\left[(\epsilon_u)_{j'p}\right] \mathbb{E}\left[(y_t - y_t^*)^2 e_{pq,t-s} e_{pq,t-u}\right] \\
&\overset{(b)}{=} -\mathbb{E}[\Gamma_{pq}^2] \leq 0,
\end{aligned} \tag{S32}$$

where (a) follows from the uncorrelatedness condition and (b) follows from the result of (S31). 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,

$$\mathbb{E}[\Delta E|_t] = -\eta \mathbb{E}\left[\widehat{\nabla E}^T \nabla E\right] = -\eta \sum_{p,q} \mathbb{E}\left[\widehat{\frac{dE}{dW_{pq}}} \frac{dE}{dW_{pq}}\right] \leq 0. \tag{S33}$$

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 [65]. Despite that, ModProp still lead to performance improvement over existing rules for tasks examined in this study. Nevertheless, it is important to investigate ModProp across a broad range of tasks in the future.

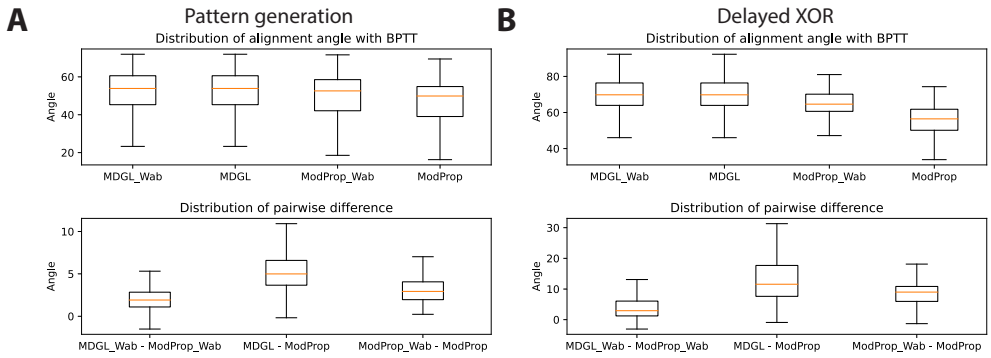

Figure S6: **Alignment angle comparison shows that gradients approximated by ModProp (with or without type-specific modulatory weights) are more similar (than MDGL) to the exact gradients**. We quantify the similarity between approximated and exact gradients via the alignment angle, which describes the similarity in the direction of the two update vectors (Appendix F) 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.

F Simulation details

All weight updates were implemented using Adam with default parameters [66]. 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) 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. S8) 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 [67].

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. S1. 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 [58], 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 [68] version 1.14 and based it on top of [67]. [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 and 4) on the server.

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