Supplementary materials: Cell-type-specific neuromodulation guides synaptic credit assignment in a spiking neural network

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

Supplementary Note 1 – Online modulatory signaling for leaky output

As mentioned in Methods, we allow leaky outputs

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

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

Based on this leaky output and loss $E = \sum_{k,t}(y^*_{k,t} - y_{k,t})^2$, we have the following partial derivative that appears in the naive implementation of modulatory emission (Eq. 24):

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

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

Consider the two additive components of our learning rule in Eq. 20. The first term, $\frac{\mathrm{d}\, E}{\mathrm{d}\, w_{pq}}\Big|_{e-prop} = L_{p,t} e_{pq,t}$, is implementable online following a simple change of summation order [1] (derivation can be generalized to the classification task with a simple replacement of $(y^*_{k,t} - y_{k,t})$ by $(\pi^*_{k,t} - \pi_{k,t})$):

$$\frac{\mathrm{d}\, E}{\mathrm{d}\, w_{pq}}\Big|_{e-prop} = \sum_{t'} \frac{\partial E}{\partial z_{p,t'}} e_{pq}^{t'}$$

$$= \sum_{k,t'} w_{kj}^{OUT} \sum_{t \geq t'} (y_{k,t}^* - y_{k,t}) \kappa^{t-t'} e_{pq}^{t'}$$

$$= \sum_{k,t} w_{kj}^{OUT} (y_{k,t}^* - y_{k,t}) \underbrace{\textstyle\sum_{t' \leq t} \kappa^{t-t'} e_{pq}^{t'}}_{\mathcal{F}_\kappa(e_{pq}^t)}, \tag{2}$$

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

We again apply the change of summation order trick to the second term of our learning rule (Eq. 20), $\Gamma_{pq,t}$, and assume that activity of neuron j is not correlated with the eligibility trace of synapse pq:

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

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

Supplementary Note 2 – Detailed Breakdown of MDGL's Components

In the main text, we stated that our MDGL learning rule combines the eligibility trace with both top-down learning signals and cell-type-specific weighted summation of secreted, diffuse modulators. We so far only expressed these components as derivatives (see Methods). With the derivation of the online implementation for MDGL in Eq. 3, we are now ready to provide the detailed expressions for each of these components. Combining Eq. 3 with Eq. 20 and rearranging the summation order gives the following component breakdown for our online approximation to MDGL:

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

The non-neuron-specific error signal $(y^*_{t,k} - y_{t,k})$ is passed to cells through neuron-specific feedback weights w^{OUT}_{kj}, thereby forming neuron-specific learning signal at the receiving end $L_{j,t} = \sum_k w^{OUT}_{kj}(y^*_{t,k} - y_{t,k})$. Here, we took top-down learning signals to be cell-specific rather than global, which is justified in part by recent reports that dopamine signals [2] and error-related neural firing [3] can be specific to a population of neurons [1]. On the other hand, loosening this neuron-specificity of learning signal can be achieved through approximations to the feedback weights, such as cell-type-specific approximations as in Eq. 23. Upon receipt, neuron j multiplies $L_{j,t}$ with $\mathcal{F}_\kappa(h_{j,t})$, its low-pass filtered activity, and sends the packaged signal $a_{j,t} = L_{j,t}\mathcal{F}_\kappa(h_{j,t})$. In updating w_{pq}, our addition allows postsynaptic cell p to collect information regarding the activities and learning signals of other cells through cell-type-specific gain $w_{\alpha\beta}$, and combine the received modulatory input with its low-pass filtered eligibility trace.

Supplementary Note 3 – Analysis and Simulation Details

Throughout this study, we used alignment angle to quantify the similarity between two vectors. The alignment angle θ between two vectors, a and b, was computed by $\theta = acos(\left\|a^T b\right\| / \|a\| \, \|b\|)$. The alignment between two 2D matrices was computed by flattening the matrices into vectors. For spectral analysis, we first performed root mean square normalization on the signal and then computed the power spectral density using Welch's method [4].

For the pattern generation task in Figure 3a, our network consisted of 400 LIF neurons. All neurons had a membrane time constant of $\tau_m = 30\text{ms}$, a baseline threshold of $v_{\text{th}} = 0.01$ and a refractory period of 2ms. Input to this network was provided by 100 Poisson spiking neurons with a rate of 10Hz. The fixed target signal had a duration of 2000ms and given by the sum of five sinusoids, with fixed frequencies of 0.5Hz, 1Hz, 2Hz, 3Hz and 4Hz. For learning, we used mean squared loss function and for visualization, we used normalized mean squared error NMSE $= \frac{\sum_{k,t}(y^*_{k,t} - y_{k,t})^2}{\sum_{k,t}(y^*_{k,t})^2}$ for zero-mean target output $y^*_{k,t}$. All weight updates were implemented using Adam with default parameters [5] and a learning rate of 1×10^{-3}. In addition, we applied firing rate regularization with $c_{\text{reg}} = 10$ and $f^{\text{target}} = 10Hz$.

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

For the evidence accumulation task in Figure 3c, our network consisted of 50 LIF neurons and 50 ALIF neurons. All neurons had a membrane time constant of $\tau_m = 20\mathrm{ms}$, a baseline threshold of $v_{\mathrm{th}} = 0.01$ and a refractory period of 5ms. The time constant of threshold adaptation was set to $\tau_b = 2000\mathrm{ms}$, and its impact on the threshold was set to $\beta = 1.8$. Input to this network was provided by four populations of 10 neurons each, as illustrated in Figure 3c. Each cue is represented by 40 Hz Poisson spiking input for 100ms and cues are separated by 50ms. The first (resp. the second) population produced Poisson spike trains with a rate of 40Hz when a cue was presented on the left (resp. right) side of the track. The third input population spiked randomly through the decision period with a firing rate of 40Hz and was silent otherwise. The last input population produced Poisson spike trains with a rate of 10Hz throughout the trial in order to prevent the network from being quiescent during the delay. For learning, we used the cross-entropy loss function and the target corresponding to the correct output was given at the end of the trial. A weight update was applied once every 64 trials and the gradients were accumulated during those trials additively. All weight updates were implemented using Adam with default parameters [5] and a learning rate of 2.5×10^{-3}. In addition, we applied firing rate regularization with $c_{\mathrm{reg}} = 0.1$ and $f^{\mathrm{target}} = 10Hz$. In the main text, all simulated tasks are constrained at 10% sparsity. This connection sparsity is maintained by fixing inactive synapses with zero weights. Cells have synapses that are sign constrained with 80% of the population being excitatory and the rest inhibitory. For a proof of concept, we implement MDGL with modulatory types mapped to the two main cell classes, thus obtaining four cell-type-specific gain values (Cell-type-specific receptor affinities, Methods). The same learning rate is used for all methods within a given task. We also compared the methods at their best learning rate within $\{1e-3, 2e-3, 5e-3, 1e-2, 2e-2, 5e-2\}$ and the trend still holds: BPTT learns the fastest, whereas MDGL leads to faster loss reduction over training iterations than e-prop. For all simulations, we used a time step of 1ms. We also assumed a synaptic delay of 1ms for all synapses.

Lastly, we note that while input, recurrent and output weights are all being trained, biologically plausible learning rules (i.e. e-prop and MDGL) only apply to input and recurrent weights. All approaches update the output weights using backpropagation, as output weights do not suffer the aforementioned nonlocality issue. (For updating the weights of a single output layer, random feedback alignment [6] has also been shown to be an effective and biologically plausible solution.)

– References — supplementary materials

[1] 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.

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

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

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

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

[6] 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.

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

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

– Supplementary Figures

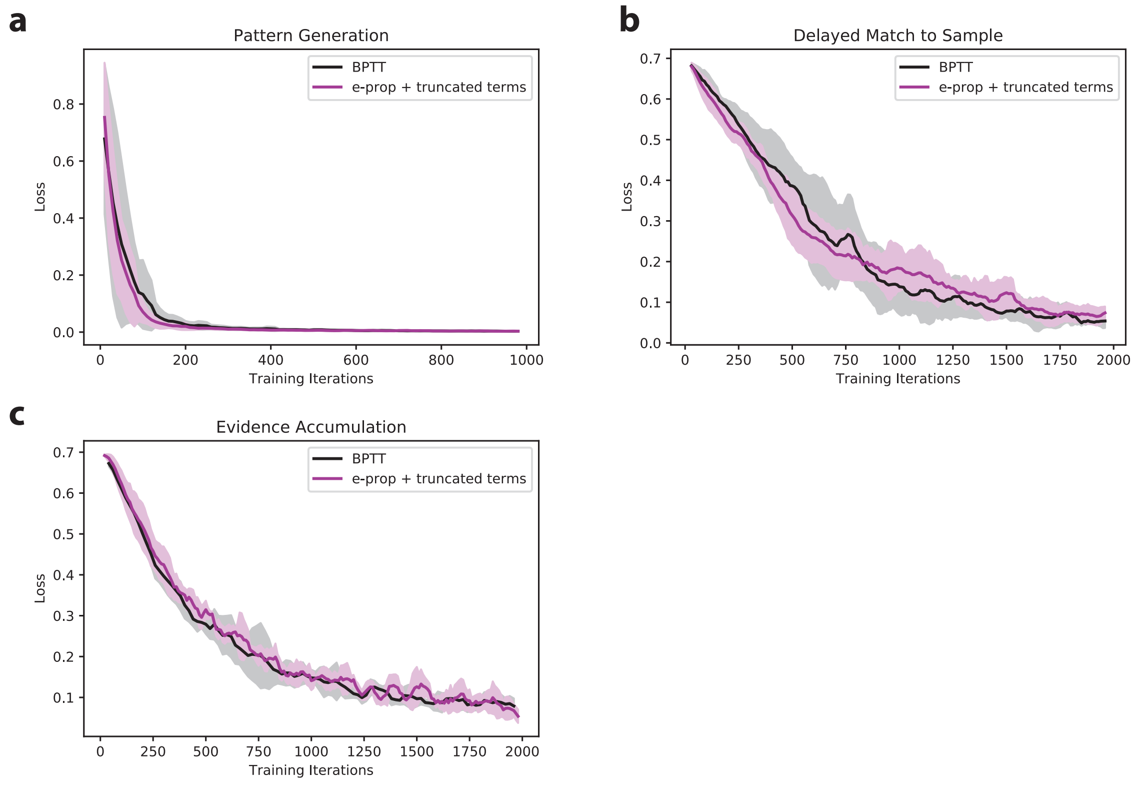


Figure 1: Checking e-prop implementation – recovering ignored terms recovers the performance of BPTT. As a sanity check, learning curves are plotted for e-prop plus all the truncated terms (see Eq. 17) to verify that the resulting learning rule recovers the performance of BPTT. The check is applied to a) pattern generation, b) delayed match to sample and c) evidence accumulation tasks. Solid lines show the mean averaged across five runs and shaded regions show the standard deviation. For all tasks, the learning curves do not differ significantly, suggesting the e-prop implementation is accurate.

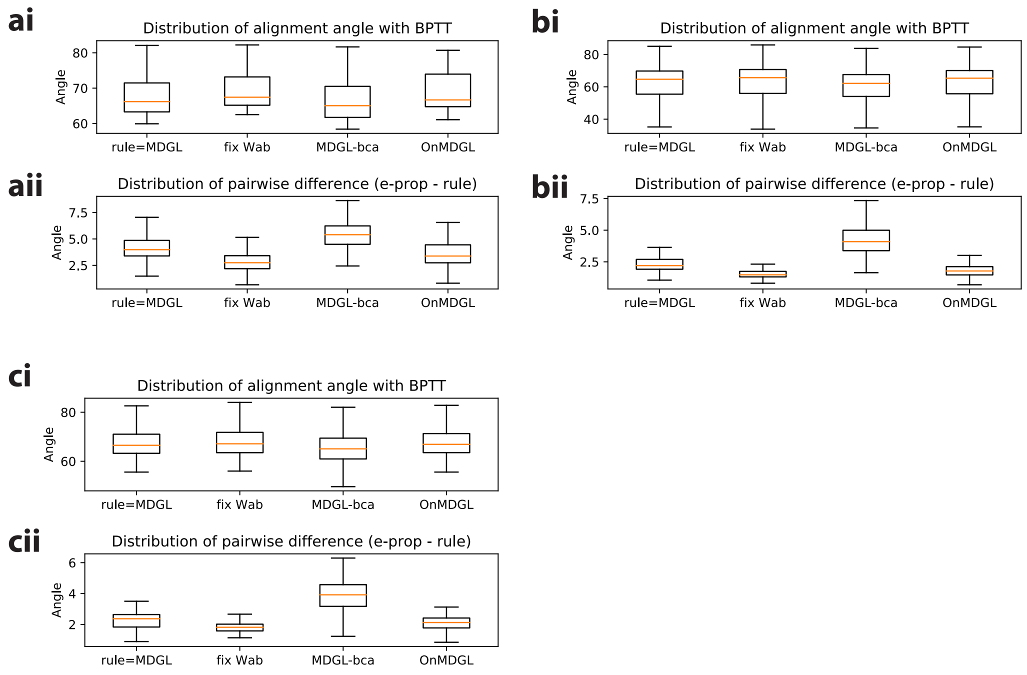


Figure 2: Alignment angle comparison shows that gradients approximated by MDGL are more similar (than e-prop) to the exact gradients. We quantify the similarity between approximated and exact gradients via the alignment angle, which describes the similarity in the direction of the two update vectors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

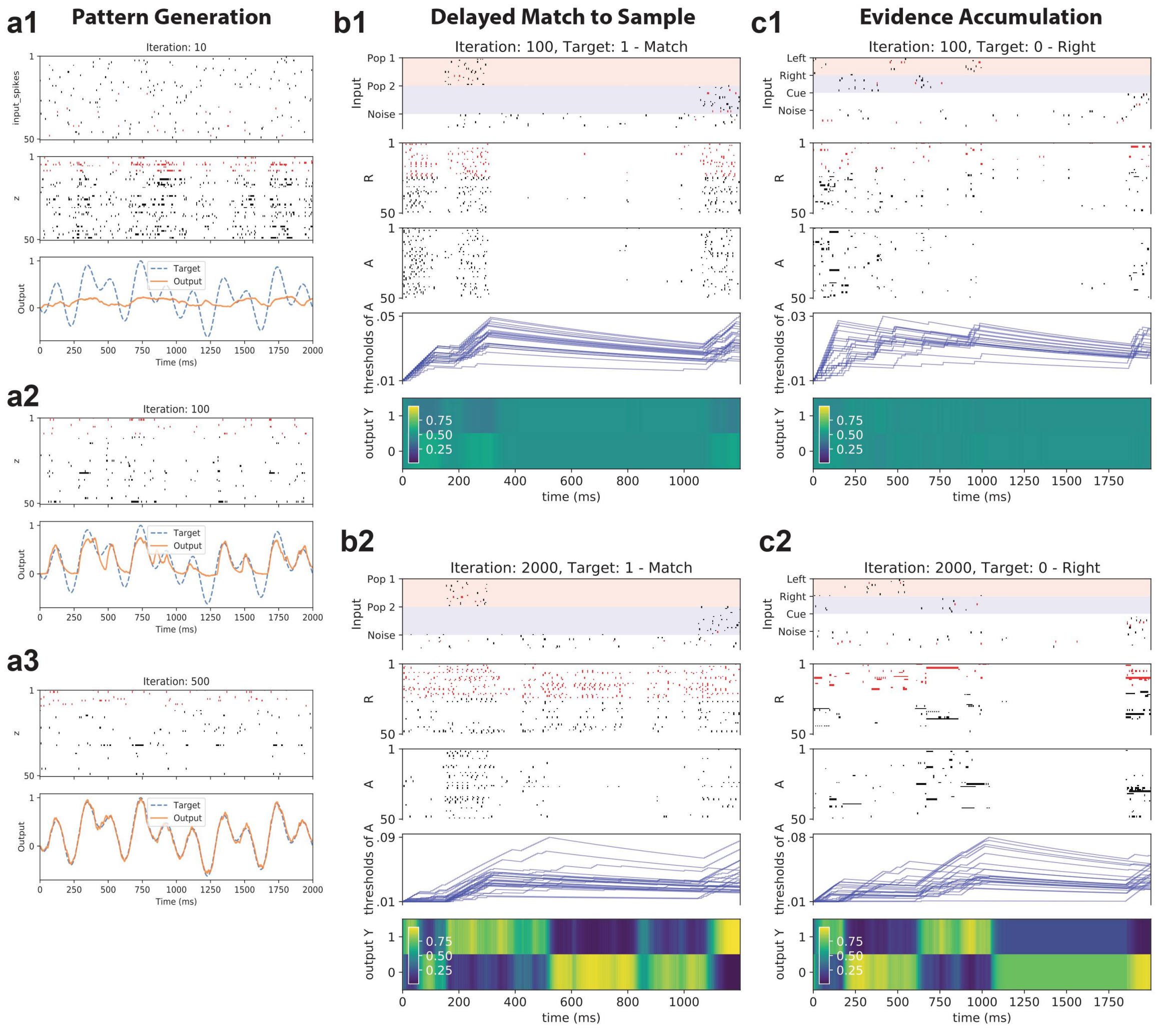


Figure 3: Network dynamics across multiple tasks investigated in Figure 3. a) Dynamics of the input, output and recurrent units are shown after 1, 100 and 500 iterations of training for the pattern generation task in Figure 3a using the MDGL method. Raster plots are shown for 50 selected sample cells, and E cells and I cells are color coded using black and red, respectively. All recurrent units have fixed thresholds for this task. Recurrent unit spikes are irregular throughout training. Network output approaches the target as training progresses. b) Network dynamics of an example trial after 100 and 2000 iterations of training for the delayed match to sample task in Figure 3b using MDGL. To emphasize the change in dynamics over training iterations, we used the same cue pattern for the illustrations. Again, E cells and I cells are color coded using black and red, respectively. For this task, both recurrent units with adaptive threshold (labeled as A) and without (labeled as R) are involved [1]. Threshold dynamics of sample neurons are illustrated. The network makes the correct prediction with greater confidence as training progresses. c) Network dynamics (Input spikes, recurrent unit spikes and readout) of an example trial after 100 and 2000 iterations of training for the evidence accumulation task in Figure 3c using MDGL. The network makes the correct prediction with greater confidence as training progresses. For all methods, results were obtained without using stochastic rewiring, which would allow for random formation of new synapses in each experience (Deep R) [7].

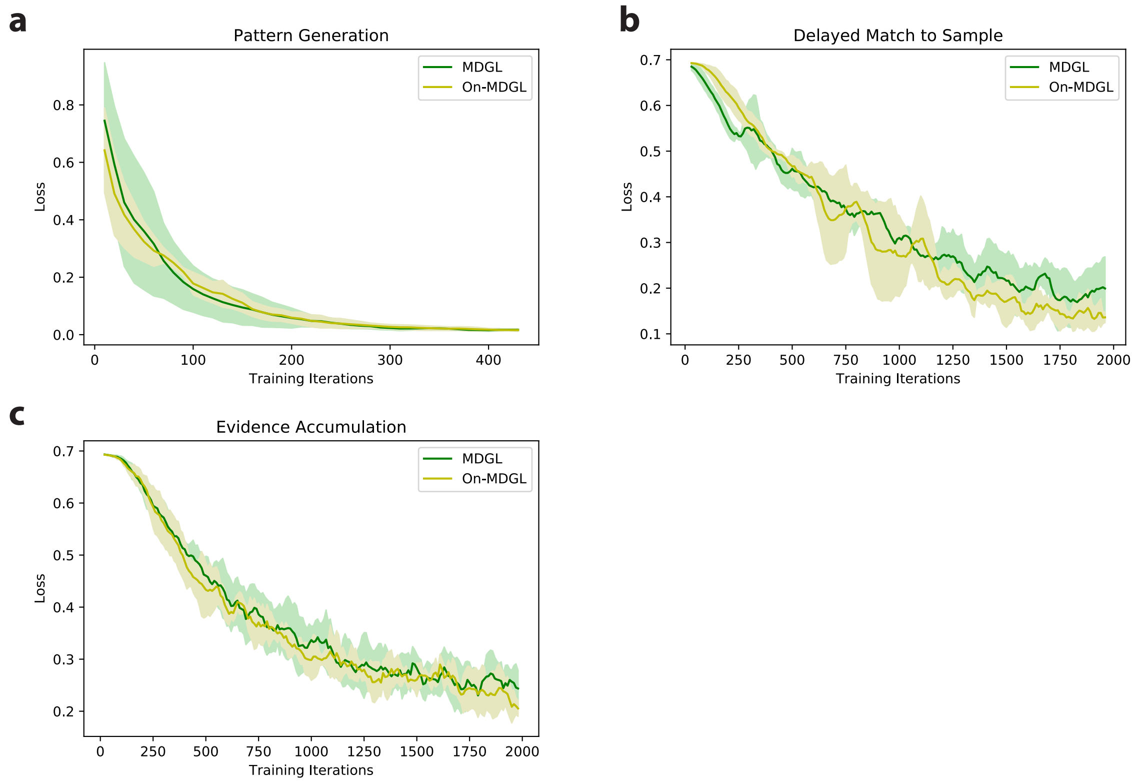


Figure 4: No significant degradation in performance observed for the online approximation of MDGL in Eq. 3. The naive implementation of activity dependent modulatory emission (Eq. 24 in Figures 3–4 depends on future errors, as explained in Supplementary Note 1 when the readout is leaky (depends on past output value). Therefore, we introduce an approximation in Eq. 3 for online implementation of MDGL. To check if this approximation leads to significant degradation in performance, learning curves are plotted for a) pattern generation, b) delayed match to sample and c) evidence accumulation tasks. For all tasks, there is no significant deviation in learning curves between MDGL and our proposed online approximation (On-MDGL).

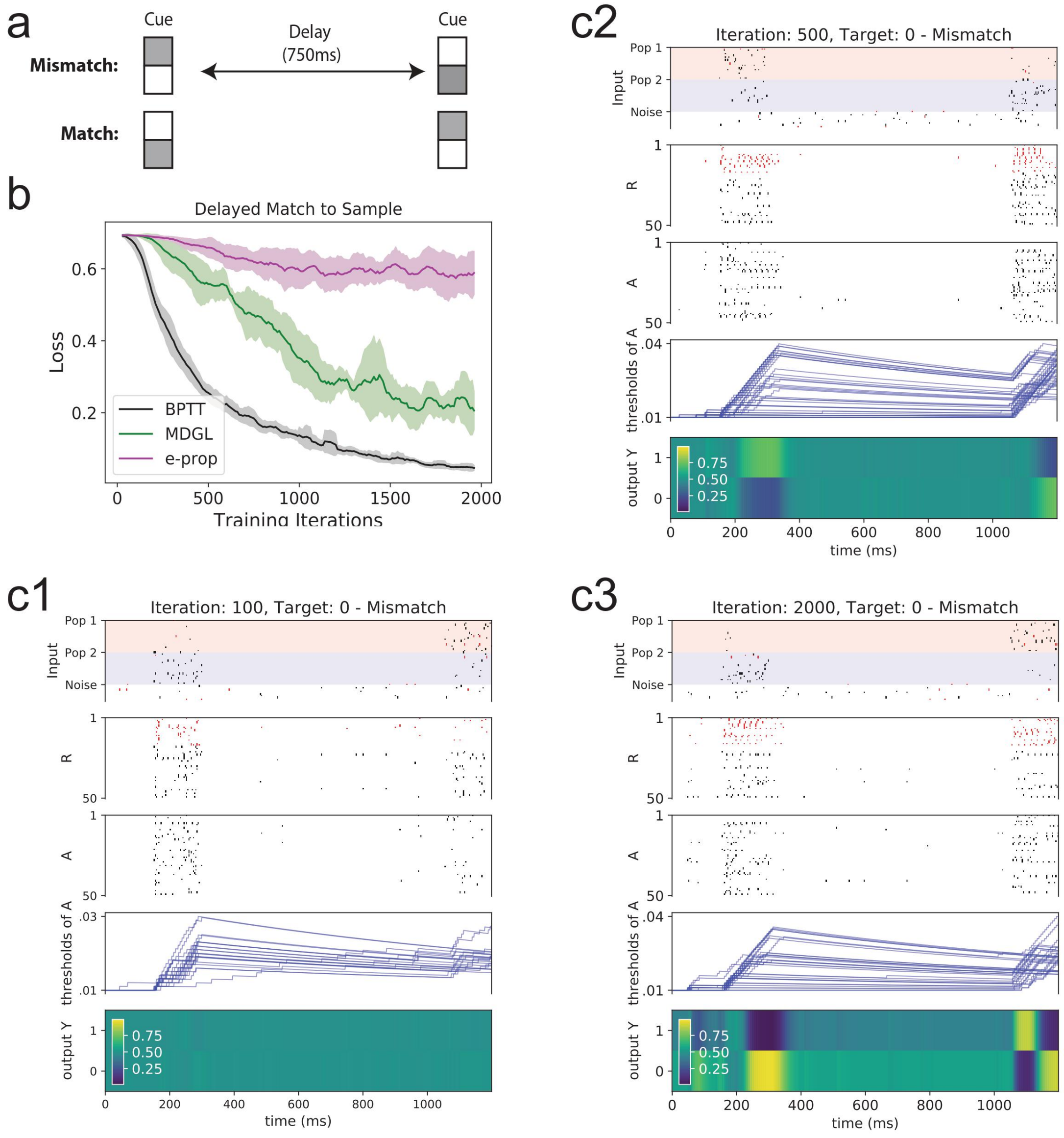
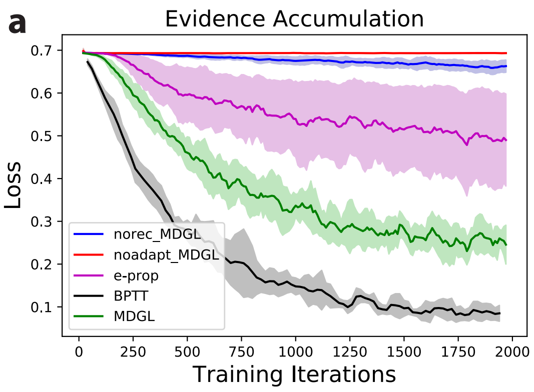


Figure 5: Similar observations for alternative task parameters in the task of Figure 3b. The delayed match to sample task in Figure 3b is repeated here, but with nonzero firing rates for the second cue alternative. As before, two input populations take on two different firing statistics to represent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



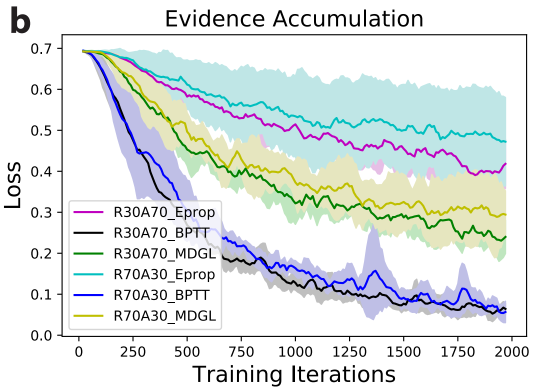


Figure 6: Threshold adaptation analysis for the evidence accumulation task in Figure 3c. a) Both threshold adaptation and recurrence are needed for successful completion of the task. MDGL trained on a network without ALIF cells (red) or with recurrent connections removed (blue) shows little decrease in loss over training iterations. b) Simulating with 70 LIF to 30 ALIF cells (R70A30) as well as 30 LIF to 70 ALIF cells (R30A70) led to similar ordering of performance for different learning methods as the default (50 LIF to 50 ALIF cells).

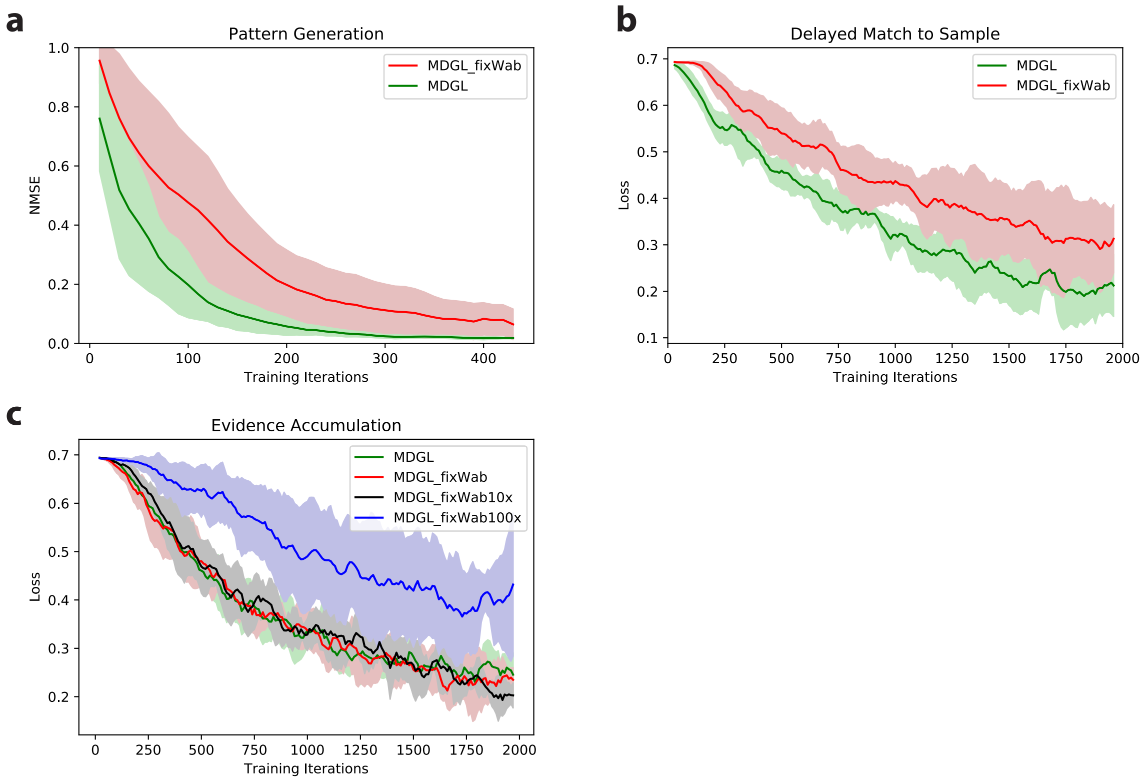


Figure 7: Comparing the learning curves for default MDGL versus MDGL with random fixed cell-type-specific receptor effcacies. As explained in Methods (Eq. 23), cell-type-specific receptor affinities were taken to be average connection weights. To explore the sensitivity of the learning performance to imprecise receptor affinities, here, the magnitude of each receptor affinity $w_{\alpha\beta}$ (for $\alpha \in \{I, E\}$, $\beta \in \{I, E\}$) is taken as the absolute value of a Gaussian random variable with zero mean and variance of $\frac{1}{\sqrt{N}}$ (N is the number of neurons), while the sign is kept as the neuron sign of type β; $w_{\alpha\beta}$ is randomized upon each initialization and fixed throughout the training. We observe a relatively mild degradation in performance for the pattern generation task and delayed match to sample task using this fixed random $w_{\alpha\beta}$ (labeled as MDGL_fixWab in each panel). For the evidence accumulation task, we did not observe any degradation even when the randomly generated $w_{\alpha\beta}$ was multiplied by a factor of 10 (labeled as MDGL_fixWab10x). A factor of 100 (labeled as MDGL_fixWab100x) pushes the network outside of an efficient operating range for the evidence accumulation task, suggesting that different tasks exhibit different degrees of tolerance to deviations in receptor affinities. This comparison is done for a) pattern generation, b) delayed match to sample and c) evidence accumulation tasks.

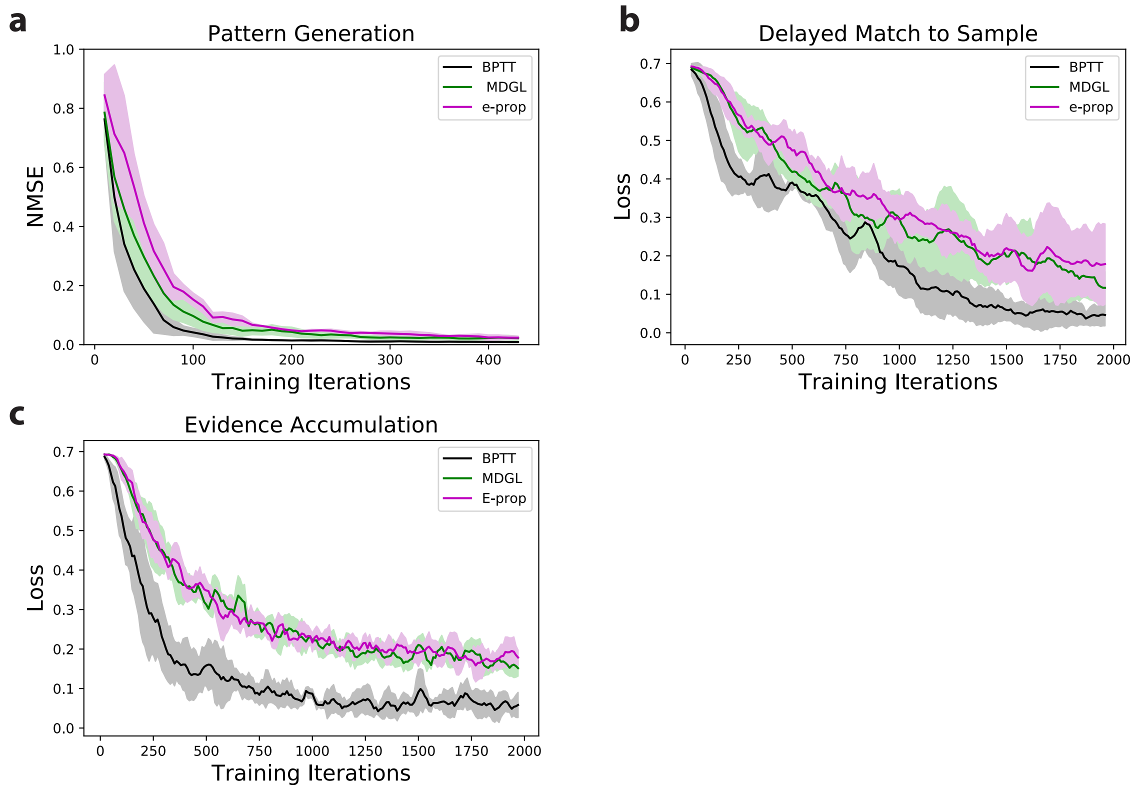


Figure 8: Advantage of MDGL disappears when the readout projection is dense. Through the paper, we instantiate the network with sparse connectivity to resemble neuronal circuits. Here, we repeat the tasks using networks with a dense connection to the readout (while keeping connections sparse within the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

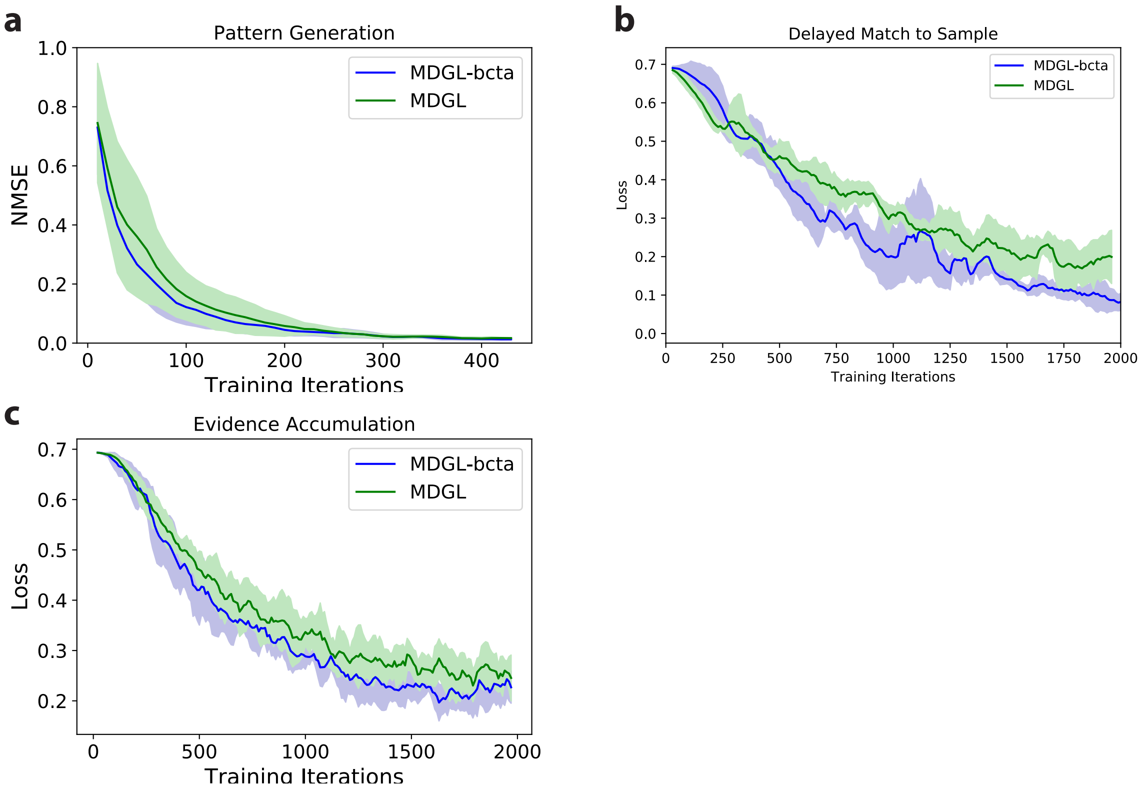


Figure 9: Effectiveness of minimal cell-type discretization. MDGL-bcta: MDGL before cell type approximation. In Methods, we defined the cell-type-specific receptor affinities to be average connection weights between cell types, i.e. $w_{\alpha\beta} =< w_{jp} >_{j\in\alpha,p\in\beta}$ (Eq. 23); we considered a minimal implementation of modulatory types mapped to the two main cell classes ($\alpha, \beta \in \{E, I\}$). We compare its learning performance to MDGL before cell type approximation as in Figure 5 (MDGL-bcta), which does not involve cell-type approximation, i.e. $w_{\alpha\beta} = w_{jp}$ (each cell is its own type). A cartoon illustration of MDGL-bcta can be found in Figure 5c. This comparison is applied to a) pattern generation, b) delayed match to sample and c) evidence accumulation tasks. The proximity of learning curves for MDGL-bcta and MDGL illustrates the effectiveness of such cell-type discretization.

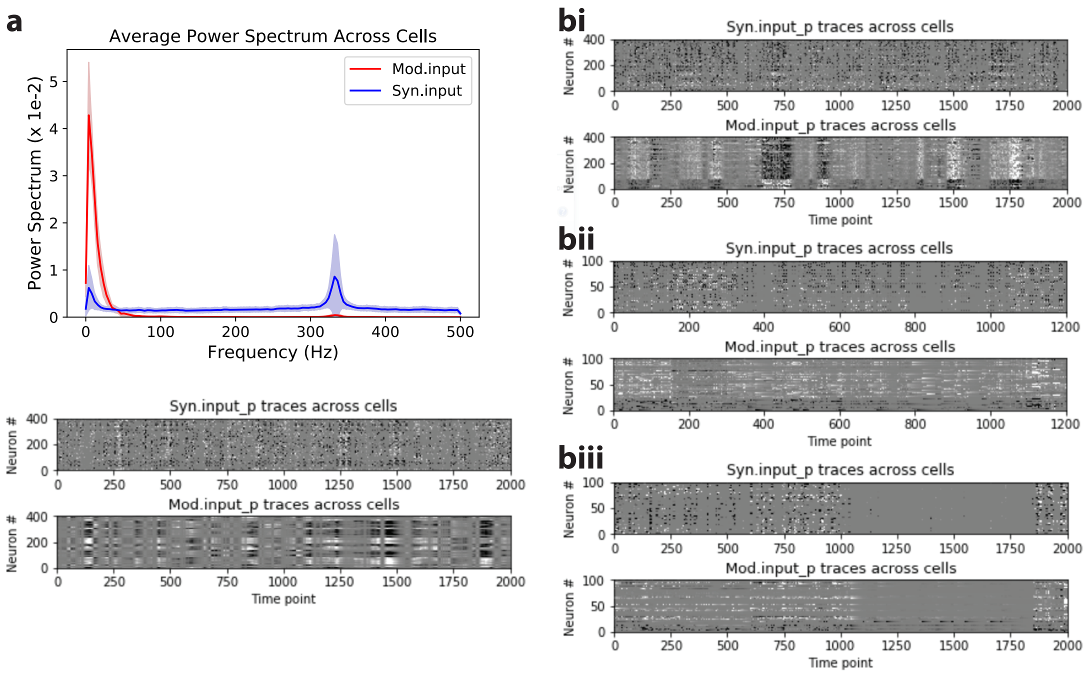


Figure 10: Slowness in modulatory signaling for the online approximation of MDGL. As explained in Figure 4, we proposed and tested an online implementation of activity-dependent modulatory release in Eq. 3. a) We repeat the spectral analysis in Figure 4a-c for this online implementation, i.e. replace $a_{j,t}$ in $Mod.input_p$ (Eq. 21) with online activity-dependent modulatory release $\bar{a}_{j,t}$ defined in Eq. 3. The observations here match those of Figure 4, where modulatory input is significantly slower than synaptic input. We note that the analysis here is done on the pattern generation task only, because for the other two tasks, the error signal is not available until the end of the trial, making the modulatory input too short (see Eq. 3) for any meaningful spectral analysis. Phenomenologically, the "slowness" of modulatory signaling can be explained by the modulatory input being a weighted summation of slow changing leaky outputs and low-pass filtered activity (Eq. 3). Raw synaptic and modulatory input (across time steps and cells) used for the frequency analysis are included beneath the frequency analysis plot. b) Raw synaptic and modulatory input traces for the frequency analysis in Figure 4a-c for i) pattern generation, ii) delayed match to sample and iii) evidence accumulation tasks.

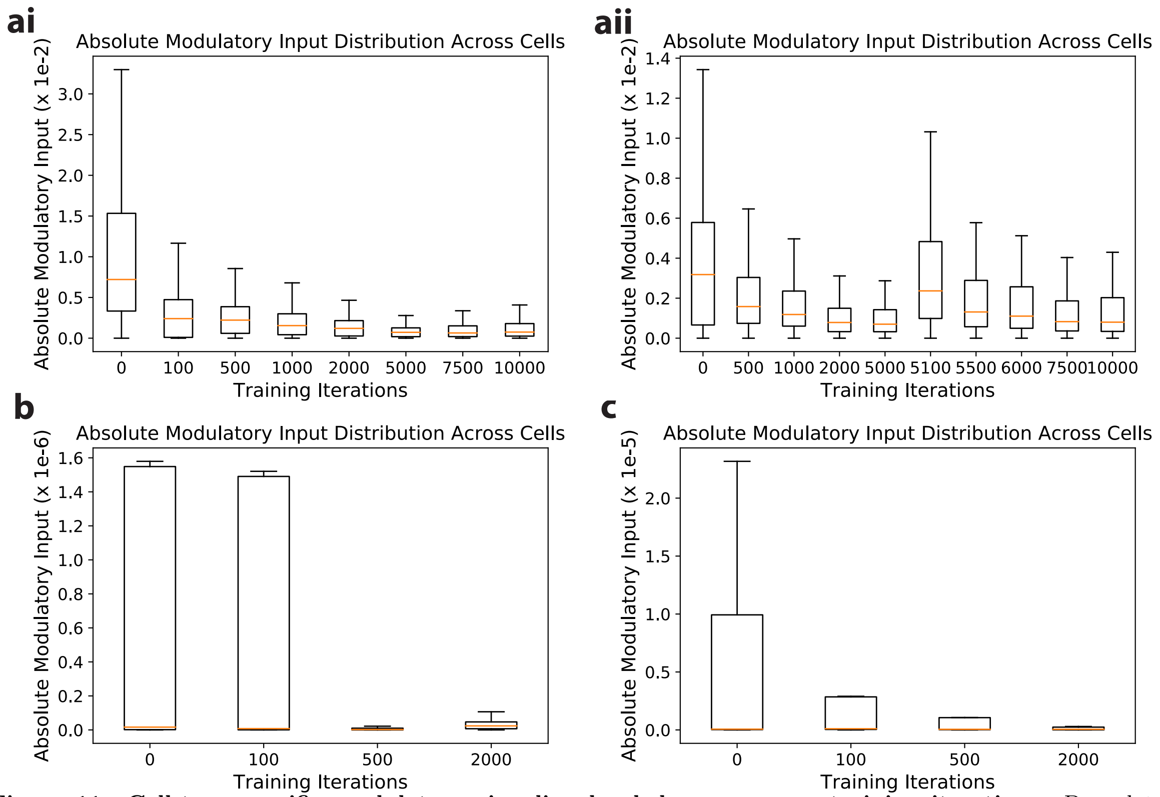


Figure 11: Cell-type-specific modulatory signaling level decreases over training iterations. Box plots for absolute cell-type-specific modulatory input distribution across cells show that modulatory signalling level drops over training iterations for ai) pattern generation, b) delayed match to sample and c) evidence accumulation tasks. aii) The target for the pattern generation task was changed after 5000 iterations, which resulted in a rapid increase in modulatory input immediately after the change, and a progressive decrease as training continued. This agrees with the prediction of our learning rule (Eq. 20, 24 and Supplementary Note 2) that cell-type-specific signaling carries information pertaining to top-down learning signals so that their levels can reflect the learning progress. To capture the magnitude of the signaling level, the absolute modulatory input for each cell p is defined similar to $Syn.input_p$ in Eq. 21, but with the absolute value of each factor; $\sum_{\alpha\in C} |w_{\alpha\beta}| \sum_{j\in\alpha,p\to j} |\bar{a}_{j,t}|$ ($\bar{a}_{j,t}$ defined in Eq. 3).

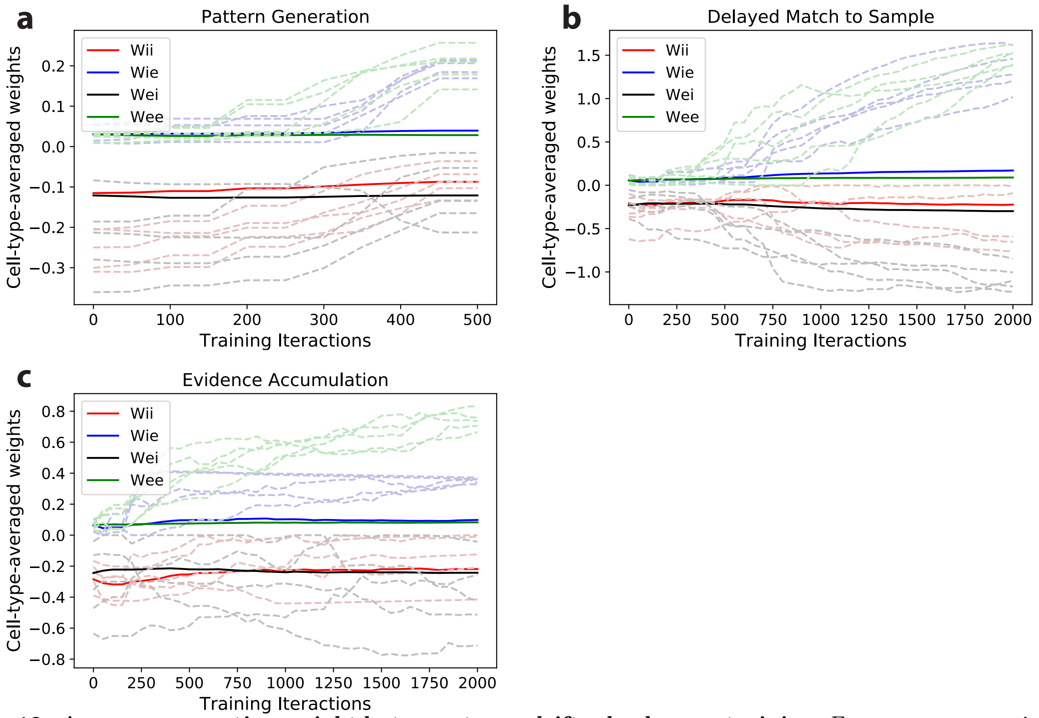


Figure 12: Average connection weight between types drifts slowly over training. Four average synaptic connection weight values (E to E, E to I, I to E and I to I) are illustrated in solid lines for the three tasks investigated. Several sample individual weights with large changes are illustrated in faint dashed lines, which can deviate significantly from the type averages. As explained earlier in Methods (Eq. 23) as well as Figure 9, these four average connection weights are used as our minimal implementation of cell-type-specific receptor affinities, $w_{\alpha\beta}$ with $\alpha, \beta \in \{E, I\}$ (see Eq. 23 and Eq. 20). How tightly the individual synaptic weights and cell-type-specific receptor affinities co-adapt may be explored in future work. Figure 7 suggests that the effect of imprecise GPCR affinities on the performance is task-dependent.

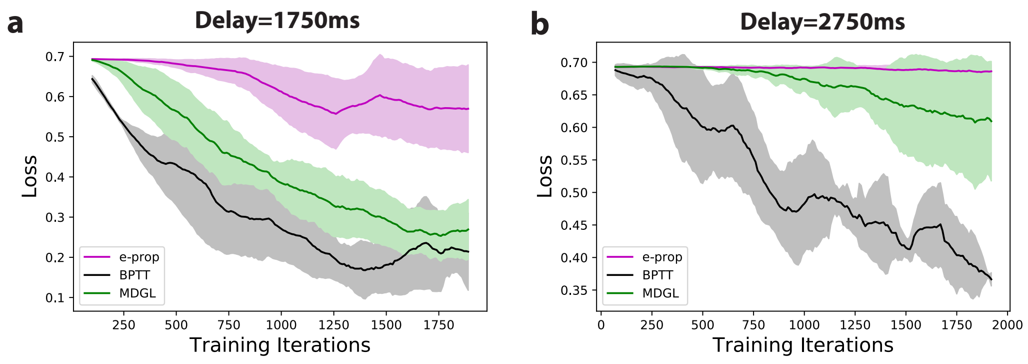


Figure 13: The performance of MGDL degrades when the delay period length is increased beyond a certain point. Here, the delay period is parametrically modulated for the delayed match to sample task. In a), the delay period is increased by 1000ms to 1750ms, and the network can still learn via MDGL. In b), the delay period is increased by 2000ms to 2750ms, and the network struggles to learn with MDGL while it still learns via BPTT. All other parameters (notably threshold adaptation time constant) are fixed in these simulations. It is interesting to note that worsened learning performance with increased delay period has previously been observed in animal experiments [8].