The network here is trained on learning a set of temporal sequences of outputs given a set of temporal sequences of inputs. The length of a sequence is 2000 timesteps. The network has 500 input units and 3 output units. The network with one hidden layer has 300 units in the hidden layer.

Training Set

The training set consists of 10 classes of input-target sequences – each class is based on a distinct set of input and target sequences which are perturbed slightly in order to generate different examples from that class. Note that the target here refers to the target *event rates* of the output layer units (see Network Structure and Dynamics). For a given class, we first randomly generate the canonical input & target output sequences which examples from that class will be based on. The canonical sequence for input unit i is given by a randomly generated sinusoidal curve:

$$x_i(t) = A_i \cos(B_i t + C_i) + 0.5 \tag{1}$$

where the amplitude factor A_i, the frequency B_i and the phase C_i are drawn from uniform distributions:

$$\begin{aligned} A_i &\sim \mathrm{U}(0.2, 0.4) \\ B_i &\sim \mathrm{U}(0.005, 0.03) \\ C_i &\sim \mathrm{U}(0, 1) \end{aligned} \tag{2}$$

The canonical target sequences for the 3 output units are generated in order to introduce some complexity in the functions that need to be learned – specifically, XOR functions. This is done because an XOR function cannot be properly learned without a hidden layer of neurons.

For each output unit j, we define the target sequence $\hat{\psi}_j^1$ as

$$\hat{\psi}_j^1(t) = \begin{cases} 0.8, & \text{if } x_a(t) > \gamma \text{ or } x_b(t) > \gamma \\ 0.2, & \text{if } x_a(t) > \gamma \text{ and } x_b(t) > \gamma \\ 0.2, & \text{if } x_a(t) \leq \gamma \text{ and } x_b(t) \leq \gamma \end{cases} \tag{3}$$

This is the XOR function applied to the thresholded versions of inputs $x_a(t)$ and $x_b(t)$, with a threshold γ, where a and b are different for each target unit. This creates a square wave, which is finally smoothed by fitting a cubic spline to $\hat{\psi}_j^1$.

a and b are chosen randomly for each output unit. The threshold γ is set to 0.5.

Thus, the target activity for each of the output units is to be active when either input x_a or x_b is active, but not when both are active, regardless of the activities of the other inputs (see Figure 1).

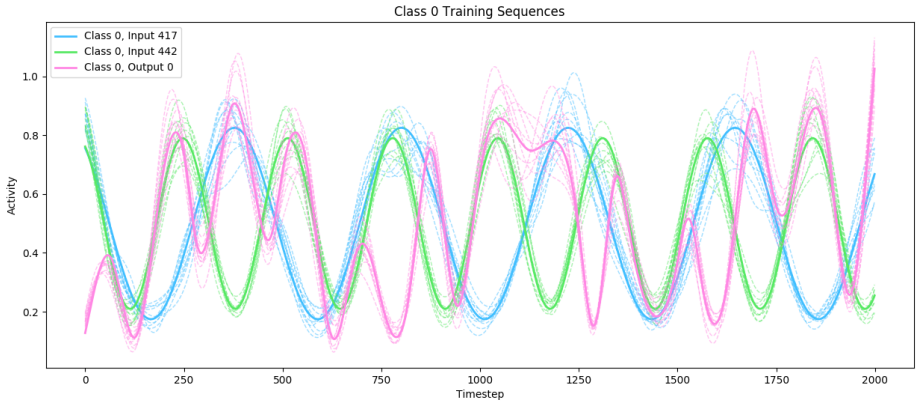

Figure 1: Activity sequence of output unit 0 for training class 0, and the activity sequences of the two input units which the target activity of output unit 0 was based on, $a = 417$ and $b = 442$. This shows the XOR function properties of the target activities – the target activity of output unit 0 is high when either input has high activity, but not when both do. *Solid lines:* Canonical sequences for the given class. *Dotted lines:* Training example sequences generated based on the canonical curves.

In order to generate training examples from this class, we add random variations to the amplitude and vertical shift of the canonical sequence curves. For input unit i we define:

$$\begin{aligned} a_i^x(t) &= A_i^x \cos(B_i^x t + C_i^x) + 1 \\ s_i^x(t) &= D_i^x \cos(E_i^x t + F_i^x) \end{aligned} \tag{4}$$

where

$$\begin{aligned} A_i^x &\sim \mathrm{U}(0.05, 0.2) \\ B_i^x &\sim \mathrm{U}(0.005, 0.05) \\ C_i^x &\sim \mathrm{U}(0, 1) \\ D_i^x &\sim \mathrm{U}(0.01, 0.05) \\ E_i^x &\sim \mathrm{U}(0.005, 0.05) \\ F_i^x &\sim \mathrm{U}(0, 1) \end{aligned} \tag{5}$$

Then, to generate an example sequence, $x_i(t)$ is adjusted so that:

$$x_i(t) \longrightarrow a_i^x(t)x_i(t) + s_i^x(t) \tag{6}$$

Similarly, for output unit j, we define:

$$\begin{aligned} a_k^{\hat{\psi}^1}(t) &= A_k^{\hat{\psi}^1}\cos(B_k^{\hat{\psi}^1}t + C_k^{\hat{\psi}^1}) + 1 \\ s_k^{\hat{\psi}^1}(t) &= D_k^{\hat{\psi}^1}\cos(E_k^{\hat{\psi}^1}t + F_k^{\hat{\psi}^1}) \end{aligned} \tag{7}$$

where

$$\begin{aligned} A_i^{\hat{\psi}^1} &\sim \mathrm{U}(0.05, 0.2) \\ B_i^{\hat{\psi}^1} &\sim \mathrm{U}(0.005, 0.05) \\ C_i^{\hat{\psi}^1} &\sim \mathrm{U}(0, 1) \\ D_i^{\hat{\psi}^1} &\sim \mathrm{U}(0.01, 0.05) \\ E_i^{\hat{\psi}^1} &\sim \mathrm{U}(0.005, 0.05) \\ F_i^{\hat{\psi}^1} &\sim \mathrm{U}(0, 1) \end{aligned} \tag{8}$$

and, to generate an example sequence, $\hat{\psi}_i^1(t)$ is adjusted so that:

$$\hat{\psi}_k^1(t) \longrightarrow a_k^{\hat{\psi}^1}(t)\hat{\psi}_k^1(t) + s_i^{\hat{\psi}^1}(t) \tag{9}$$

Figure 2 shows some of the input & target sequences generated for different classes, and training examples drawn from each class.

Network Structure and Dynamics

A diagram showing the network structure and dynamics is shown in Figure 3. Assume the network has l inputs, m hidden units and n output units. Unit j in the hidden layer has two compartments: a somatic compartment with voltage y_j^0 and an apical dendrite compartment with voltage g_j^0. At time t, $y_j^0(t)$ is given by:

$$y_j^0(t) = \sum_{i=1}^{l} W_{jk}^0 \tilde{x}_i(t-1) + b_j^0 \tag{10}$$

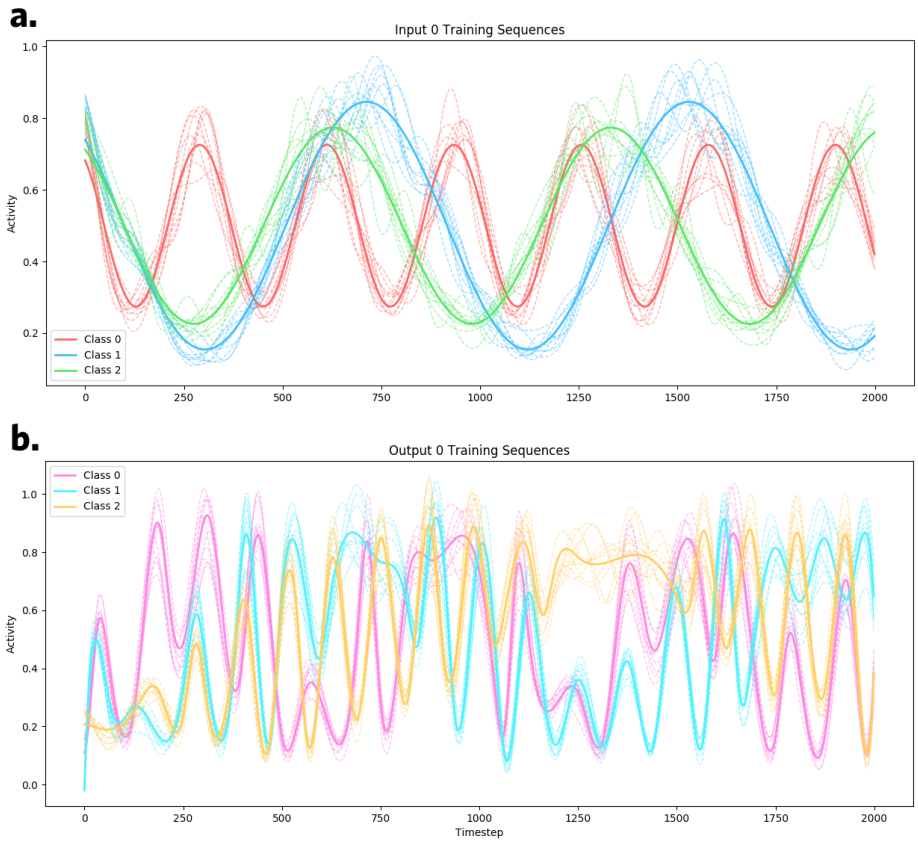

Figure 2: Input & target sequences for different classes, and sample sequences drawn from each class. **a.** Activity sequences of input unit 0 representing three different classes of training data. **b.** Activity sequences of output unit 0 representing three different classes of training data. *Solid lines:* Canonical sequences for the given class. *Dotted lines:* Training example sequences generated based on the canonical curves.

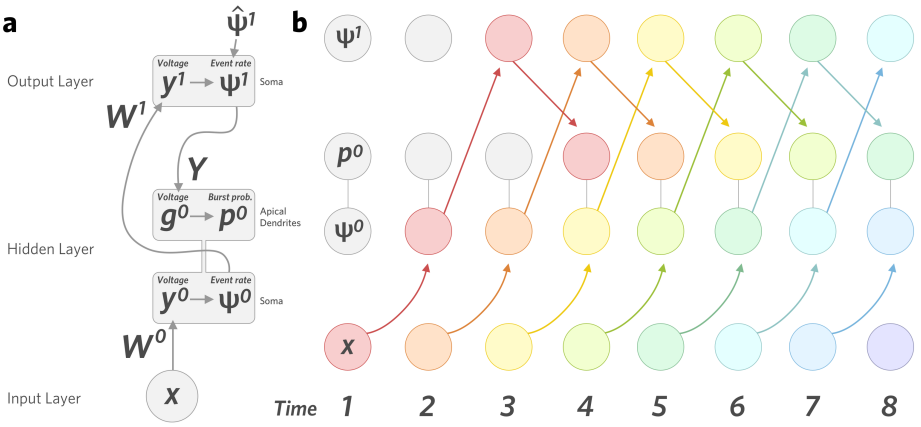

Figure 3: Diagram of the network. **a.** Network variables and connections. **b.** Temporal dynamics of the network.

where $\boldsymbol{W}^0$ is the $m \times l$ matrix of the synaptic weights between the inputs and hidden layer units, $\boldsymbol{b}^0$ is a vector containing bias terms for each hidden unit, and $\tilde{\boldsymbol{x}}$ is the exponentially smoothed input layer activity:

$$\begin{aligned} \tilde{x}_i(0) &= x_i(0) \\ \tilde{x}_i(t) &= \frac{1}{2}(x_i(t) + \tilde{x}_i(t-1)),\ t > 0 \end{aligned} \tag{11}$$

The hidden unit's *event rate* ψ_j^0, defined as the expected number of spike events (either single spikes or bursts) per unit time, is given by a sigmoid applied to the somatic voltage:

$$\psi_j^0(t) = \sigma(y_j^0(t)) = \frac{1}{1 + e^{-y_j^0(t)}} \tag{12}$$

This signal is received by units in the output layer. Unit k in the output layer has a somatic compartment with somatic voltage y_k^1 given by:

$$y_k^1(t) = \sum_{j=1}^{m} W_{ij}^1 \tilde{\psi}_j^0(t-1) + b_k^1 \tag{13}$$

where $\boldsymbol{W}^1$ are the feedforward synaptic weights between the hidden layer and output layer units, $\boldsymbol{b}^0$ are the bias terms for each output unit, and $\tilde{\boldsymbol{\psi}}^0$ are the

exponentially smoothed event rates of the hidden layer units, computed as in equation (11). Similarly, the event rate of output unit k, ψ_k^1, is given by:

$$\psi_k^1(t) = \sigma(y_k^1(t)) = \frac{1}{1 + e^{-y_k^1(t)}} \tag{14}$$

During some time steps, a target conductance nudges $\psi_k^1(t)$ towards the target $\hat{\psi}_k^1(t)$ (see Training and Testing). During these time steps, $\psi_k^1(t)$ is instead given by:

$$\psi_k^1(t) = \frac{1}{2}(\sigma(y_k^1(t)) + \hat{\psi}_k^1(t)) \tag{15}$$

Finally, the apical dendrite compartments of hidden layer units receive this signal from the output layer units. The apical voltage g_j^0 is given by:

$$g_j^0(t) = \sum_{k=1}^{n} Y_{jk} \tilde{\psi}_k^1(t-1) \tag{16}$$

where $\boldsymbol{Y}$ are the feedback synaptic weights between the output layer and hidden layer units, and $\tilde{\boldsymbol{\psi}}^1$ are the exponentially smoothed event rates of the output layer units, computed as in equation (11). The hidden unit's *burst probability* p_j^0, defined as the probability that a spike event will be a burst (rather than a single spike), is the given by applying the sigmoid function to the apical voltage:

$$p_j^0(t) = \sigma(g_j^0(t)) = \frac{1}{1 + e^{-g_j^0(t)}} \tag{17}$$

Finally, the *burst rate* φ_j^0 is the product of the burst probability and the event rate:

$$\varphi_j^0(t) = p_j^0(t)\psi_j^0(t) \tag{18}$$

Training and Testing

Training occurs over a number of epochs. The networks in the presented results were all trained for 20 epochs. During one epoch, each of the 100 training sequences (10 sequences per class) are randomly presented to the network in

sequence. While the input sequences $\boldsymbol{x}(t)$ are shown continuously, the target sequences $\hat{\boldsymbol{\psi}}^1(t)$ are only occasionally shown to the network – at each time step, there is a 5% chance of a target being presented. When a target is not presented, the the input $\boldsymbol{x}(t)$ simply propagated through the network, affecting the event rates and burst probabilities of the units. When a target is presented, the event rate of the output layer units is nudged towards this target, and feedforward weights throughout the network are updated (see Weight Updates). During time steps when the target is present, we record the training loss at each layer of the network (see Loss Functions).

Following training, the network is tested on a separate set of 10 test sequences, one per class. The test error for a given test sequence is defined as the mean absolute difference between the event rates of the output layer units and their target event rates:

$$\text{Test error} = \frac{1}{nT} \sum_{t=1}^{T} \sum_{k=1}^{n} |\hat{\psi}_k^1(t) - \psi_k^1(t)| \tag{19}$$

where $T = 2000$, the length of the sequence. Finally, each of the networks with one hidden layer are shown the 10 test sequences again, but halfway through each sequence are made to generate activity using feedback only (see Generating Activity Using Feedback).

Loss Functions

Loss functions are computed and feedforward weights are updated in order to descend their gradient only when a target, $\hat{\boldsymbol{\psi}}^1(t)$, is presented to the network. As shown above, if the target $\hat{\boldsymbol{\psi}}^1(t)$ is shown to the network at time t, it will also push the event rates of the output layer units closer to this target (equation (15)). Thus, at time t, we can define the loss function for the output layer as:

$$L^1(t) = ||\boldsymbol{\psi}^1(t) - \boldsymbol{\psi}^1(t-1)||_2^2 \tag{20}$$

As the network learns to minimize this loss, $\boldsymbol{\psi}^1(t)$ will approach $\hat{\boldsymbol{\psi}}^1(t)$ when the target is shown to the network, and this the network will also indirectly descend the following loss function:

$$L^{1*}(t) = ||\hat{\boldsymbol{\psi}}^1(t) - \boldsymbol{\psi}^1(t-1)||_2^2 \tag{21}$$

For the hidden layer, we first define a target event rate $\hat{\boldsymbol{\psi}}^0(t)$ as:

$$\hat{\psi}_j^0(t) = \psi_j^0(t-1) + p_j^0(t) - p_j^0(t-1) \tag{22}$$

Then, the loss function is defined as:

$$\begin{aligned} L^0(t) &= ||\hat{\boldsymbol{\psi}}^0(t) - \boldsymbol{\psi}^0(t-1)||_2^2 \\ &= ||p_j^0(t) - p_j^0(t-1)||_2^2 \end{aligned} \tag{23}$$

When the target $\hat{\boldsymbol{\psi}}^1(t)$ is presented to the network, feedforward weights are updated in order to descend the gradients of L^1 and L^0.

Weight Updates

Feedforward weight updates at the final layer occur at time t only if the target $\hat{\boldsymbol{\psi}}^1(t)$ is presented to the network. The feedforward weights W^1 and biases b^1 are updated as:

$$\begin{aligned} W_k^1 j &\longrightarrow W_k^1 j + \epsilon^1(\psi_k^1(t) - \psi_k^1(t-1))\psi_k^1(t-1)(1-\psi_k^1(t-1)) \cdot \tilde{\psi}_j^0(t-1) \\ b_k^1 &\longrightarrow b_k^1 + \epsilon^1(\psi_k^1(t) - \psi_k^1(t-1))\psi_k^1(t-1)(1-\psi_k^1(t-1)) \end{aligned} \tag{24}$$

where ϵ^1 is the learning rate, set to 0.01.

Feedforward weights at the hidden layer occur at the following timestep, $t+1$, as this is when the target information arrives at the apical compartments of hidden layer units. The feedforward weights W^0 and biases b^0 are updated as:

$$\begin{aligned} W_j^0 i &\longrightarrow W_j^0 i + \epsilon^0(p_j^0(t+1) - p_j^0(t))\psi_j^0(t)(1-\psi_j^0(t)) \cdot \tilde{x}_i(t) \\ &= W_j^0 i + \epsilon^0(p_j^0(t+1) - \varphi_j^0(t))(1-\psi_j^0(t)) \cdot \tilde{x}_i(t) \\ b_j^0 &\longrightarrow b_j^0 + \epsilon^0(p_j^0(t+1) - p_j^0(t))\psi_j^0(t)(1-\psi_j^0(t)) \\ &= b_j^0 + \epsilon^0(p_j^0(t+1) - \varphi_j^0(t))(1-\psi_j^0(t)) \end{aligned} \tag{25}$$

where ϵ^0 is the learning rate, set to 10.0.

Feedback Weight Updates

When training with feeback weight updates, feedback weights Y are updated at every time step t in order to descend the loss function:

$$L^Y(t) = ||\boldsymbol{\psi}^0(t) - \boldsymbol{\varphi}^0(t)||_2^2 \tag{26}$$

Thus, feedback weights Y are updated as:

$$Y_j k \longrightarrow Y_j k + \epsilon^Y(\psi_j^0(t) - \varphi_j^0(t))\psi_j^0(t)p_j^0(t)(1 - p_j^0(t)) \cdot \tilde{\psi}_k^1(t) \tag{27}$$

where ϵ^Y is the learning rate, set to 0.0001.

Generating Activity Using Feedback

In order to generate activity using the feedback of the output layer, we present a sequence $\boldsymbol{x}$ to the network after it has been trained. Starting halfway through the sequence ($t = 1000$), the event rates of the hidden layer units, $\boldsymbol{\psi}^0$, are set to be equal to their burst rates $\boldsymbol{\varphi}^0$:

$$\psi_j^0(t) = \varphi_j^0(t) \tag{28}$$

Thus, for the second half of the sequence, the event rates of the hidden layer units, which drive the event rates at the output layer, are determined only by their burst rates, which in turn are determined by the feedback from the output layer.

The generation error is defined similarly to the test error, it is the mean difference between the output of the network and the target during the second half of the sequence:

$$\text{Generation error} = \frac{1}{n(T/2)} \sum_{t=T/2}^{T} \sum_{k=1}^{n} |\hat{\psi}_k^1(t) - \psi_k^1(t)| \tag{29}$$

where $T = 2000$, the length of the sequence.

Figure Descriptions

avg_final_layer_training_losses_comparison. Plot of output layer loss L^1 over 20 epochs of training, averaged every 100 timesteps. Shown is the full range over 5 trials (light colors) and the mean across 5 trials (darker colors). (Note: x axis label should be Training Example, not Epoch).

mean_test_error_comparison. Plot of test error after 20 epochs of training, across 5 trials. For each network, there are 50 data points, corresponding to 5 trials, with 10 sequences (one from each class) being tested in each. The bars are the means.

mean_generation_error_comparison. Plot of the generation error after 20 epochs of training, across 5 trials. For each network, there are 50 data points, corresponding to 5 trials, with 10 sequences (one from each class) being tested in each. The bars are the means.

activity. *Top:* Plot of the target output and output (event rate) of the network during the first sequence of training. Dots represent when the target was shown to the network. *Bottom:* Plot of the target output and output of the network after training, during the test sequence that is the same class as in the top plot.

generated_activity. Plot of the target output and output of the network after training, where after 1000 timesteps the network begins to generate activity without input.