Simultaneous unsupervised and supervised learning of cognitive functions in biologically plausible spiking neural networks

Trevor Bekolay (tbekolay@uwaterloo.ca)
Carter Kolbeck (ckolbeck@uwaterloo.ca)
Chris Eliasmith (celiasmith@uwaterloo.ca)
Center for Theoretical Neuroscience, University of Waterloo
200 University Ave., Waterloo, ON N2L 3G1 Canada

Abstract

We present a novel learning rule for learning transformations of sophisticated neural representations in a biologically plausible manner. We show that the rule, which uses only information available locally to a synapse in a spiking network, can learn to transmit and bind semantic pointers. Semantic pointers have previously been used to build Spaun, which is currently the world's largest functional brain model (Eliasmith et al., 2012). Two operations commonly performed by Spaun are semantic pointer binding and transmission. It has not yet been shown how the binding and transmission operations can be learned. The learning rule combines a previously proposed supervised learning rule and a novel spiking form of the BCM unsupervised learning rule. We show that spiking BCM increases sparsity of connection weights at the cost of increased signal transmission error. We also demonstrate that the combined learning rule can learn transformations as well as the supervised rule and the offline optimization used previously. We also demonstrate that the combined learning rule is more robust to changes in parameters and leads to better outcomes in higher dimensional spaces, which is critical for explaining cognitive performance on diverse tasks.

Keywords: synaptic plasticity; spiking neural networks; unsupervised learning; supervised learning; Semantic Pointer Architecture; Neural Engineering Framework.

In this paper, we demonstrate learning of cognitively relevant transformations of neural representations online and in a biologically plausible manner. We improve upon a technique previously presented in MacNeil and Eliasmith (2011) by combining their error-minimization learning rule with an unsupervised learning rule, making it more biologically plausible and robust.

There are three weaknesses with most previous attempts at combining supervised and unsupervised learning in artificial neural networks (e.g., Backpropagation [Rumelhart, Hinton, & Williams, 1986], Self-Organizing Maps [Kohonen, 1982], Deep Belief Networks [Hinton, Osindero, & Teh, 2006]). These approaches 1) have explicit offline training phases that are distinct from functional use of the network, 2) require many layers with some layers connected with supervised learning and others with unsupervised learning, and 3) use non-spiking neuron models. The approach proposed here

---

BCM Bienenstock, Cooper, Munro learning rule; Eq (7)

hPES Homeostatic Prescribed Error Sensitivity; Eq (9)

NEF Neural Engineering Framework; see Theory

PES Prescribed Error Sensitivity; Eq (6)

SPA Semantic Pointer Architecture; see Theory

STDP Spike-timing dependent plasticity (Bi & Poo, 2001)

---

overcomes these limitations. Our approach 1) remains functional during online learning, 2) requires only two layers connected with simultaneous supervised and unsupervised learning, and 3) employs spiking neuron models to reproduce central features of biological learning, such as spike-timing dependent plasticity (STDP).

Online learning with spiking neuron models faces significant challenges due to the temporal dynamics of spiking neurons. Spike rates cannot be used directly, and must be estimated with causal filters, producing a noisy estimate. When the signal being estimated changes, there is some time lag before the spiking activity reflects the new signal, resulting in situations during online learning in which the inputs and desired outputs are out of sync. Our approach is robust to these sources of noise, while only depending on quantities that are locally available to a synapse.

Other techniques doing similar types of learning in spiking neural networks (e.g., SpikeProp [Bohte, Kok, & Poutre, 2002], ReSuMe [Ponulak, 2006]) can learn only simple operations, such as learning to spike at a specific time. Others (e.g., SORN [Lazar, Pipa, & Triesch, 2009], reservoir computing approaches [Paugam-Moisy, Martinez, & Bengio, 2008]) can solve complex tasks like classification, but it is not clear how these approaches can be applied to a general cognitive system. The functions learned by our approach are complex and have already been combined into a general cognitive system called the Semantic Pointer Architecture (SPA). Previously, the SPA has been used to create Spaun, a brain model made up of 2.5 million neurons that can do eight diverse tasks (Eliasmith et al., 2012). Spaun accomplishes these tasks by transmitting and manipulating semantic pointers, which are compressed neural representations that carry surface semantic content, and can be decompressed to generate deep semantic content (Eliasmith, in press). Semantic pointers are composed to represent syntactic structure using a "binding" transformation, which compresses the information in two semantic pointers into a single semantic pointer. Such representations can be "collected" using superposition, and collections can participate in further bindings to generate deep structures. Spaun performs these transformations by using the Neural Engineering Framework (NEF; Eliasmith & Anderson, 2003) to directly compute static connection weights between populations. We show that our approach can *learn* to transmit, bind, and classify semantic pointers.

Theory

Cognitive functions with spiking neurons

In order to characterize cognitive functions at the level of spiking neurons, we employ the methods of the Semantic Pointer Architecture (SPA), which was recently used to create the world's largest functional brain model (Spaun; Eliasmith et al., 2012), able to perform perceptual, motor, and cognitive functions. Cognitive functions in Spaun include working memory, reinforcement learning, syntactic generalization, and rule induction.

The SPA is implemented using the principles of the Neural Engineering Framework (NEF; Eliasmith & Anderson, 2003), which defines methods to 1) *represent* vectors of numbers through the activity of populations of spiking neurons, 2) *transform* those representations through the synaptic connections between those populations, and 3) incorporate *dynamics* through connecting populations recurrently. In the case of learning, we exploit NEF representations in order to learn transformations analogous to those that can be found through the NEF's methods.

Representing semantic pointers in spiking neurons Representation in the NEF is similar to population coding, as proposed by Georgopoulos, Schwartz, and Kettner (1986), but extended to *n*-dimensional vector spaces. Each population of neurons represents a point in an *n*-dimensional vector space over time. Each neuron in that population is sensitive to a direction in the *n*-dimensional space, which we call the neuron's *encoder*. The activity of a neuron can be expressed as

$$a = G[\alpha \mathbf{e} \cdot \mathbf{x} + J_{bias}], \tag{1}$$

where $G[\cdot]$ is the nonlinear neural activation function, α is a scaling factor (gain) associated with the neuron, $\mathbf{e}$ is the neuron's encoder, $\mathbf{x}$ is the vector to be encoded, and J_{bias} is the background current of the cell.

The vector (i.e., semantic pointer) that a population represents can be estimated from the recent activity of the population. The decoded estimate, $\hat{\mathbf{x}}$, is the sum of the activity of each neuron, weighted by an *n*-dimensional *decoder*.

$$\hat{\mathbf{x}}(t) = \sum_i \mathbf{d}_i a_i(t), \tag{2}$$

where $\mathbf{d}_i$ is the decoder and a_i is the activity of neuron *i*.

Neural activity is interpreted as a filtered spike train; i.e.,

$$a_i(t) = \sum_s h(t - t_s) = \sum_s e^{-(t-t_s)/\tau_{PSC}}, \tag{3}$$

where $h(\cdot)$ is the exponential filter applied to each spike, and *s* is the set of all spikes occurring before the current time *t*.

The decoders are found through a least-squares minimization of the difference between the decoded estimate and the actual encoded vector.

$$\mathbf{d} = \Upsilon^{-1}\Gamma \qquad \Gamma_{ij} = \int a_i a_j dx \qquad \Upsilon_j = \int a_j \mathbf{x} dx \tag{4}$$

Transforming semantic pointers through connection weights The encoders and decoders used to represent semantic pointers also enable arbitrary transformations (i.e., mathematical functions) of encoded semantic pointers. If population A encodes pointer *X*, and we want to connect it to population B, encoding pointer *Y*, a feedforward connection with the following connection weights transmits that semantic pointer, such that $Y \approx X$.

$$\omega_{ij} = \alpha_j \mathbf{e}_j \mathbf{d}_i, \tag{5}$$

where *i* indexes the presynaptic population A, and *j* indexes the postsynaptic population B. Other linear transformations are implemented by multiplying $\mathbf{d}_i$ by a linear operator. Nonlinear transformations are implemented by solving for a new set of decoding weights. This is done by minimizing the difference between the decoded estimate of $f(\mathbf{x})$ and the actual $f(\mathbf{x})$, rather than just $\mathbf{x}$, in Equation (4).

Supervised learning: PES rule

MacNeil and Eliasmith (2011) proposed a learning rule that minimizes the error minimized in Equation (4) online.

$$\Delta \mathbf{d}_i = \kappa \mathbf{E} a_i$$
$$\Delta \omega_{ij} = \kappa \alpha_j \mathbf{e}_j \cdot \mathbf{E} a_i, \tag{6}$$

where κ is a scalar learning rate, $\mathbf{E}$ is a vector representing the error we wish to minimize, and other terms are as before.

When put in terms of connections weights (ω_{ij}), the rule resembles backpropagation. The quantity $\alpha_j \mathbf{e}_j \cdot \mathbf{E}$ is analogous to the local error term δ in backpropagation (Rumelhart et al., 1986); they are both a means of translating a global error signal to a local error signal that can be used to change an individual synaptic connection weight. The key difference between this rule and backpropagation is that the global-to-local mapping is done by imposing the portion of the error vector space each neuron is sensitive to via its encoder. This limits flexibility, but removes the dependency on global information, making the rule biologically plausible. We will refer to Equation (6) as the Prescribed Error Sensitivity (PES) rule.

Unsupervised learning: Spiking BCM rule

A Hebbian learning rule widely studied in the context of the vision system is the BCM rule (Bienenstock, Cooper, & Munro, 1982). This rule has been used to explain orientation selectivity and ocular dominance (Bienenstock et al., 1982). Theoretically, it has been asserted that BCM is equivalent to triplet-based STDP learning rules (Pfister & Gerstner, 2006).

The general form is

$$\Delta \omega_{ij} = a_i a_j (a_j - \theta), \tag{7}$$

where θ is the *modification threshold*. When the postsynaptic activity, a_j, is greater than the modification threshold, the synapse will be potentiated; when the postsynaptic activity is less than the modification threshold, the synapse will be depressed.

The modification threshold reflects the expectation of a cell's activity. It is typically calculated as the temporal average of the cell's activity over a long time window (on the order of hours). The intuition behind BCM is that cells driven above their expectation must be playing an important role in a circuit, so their afferent synapses become potentiated. Cells driven less than normal have synapses depressed. If either of these effects persists long enough, the modification threshold changes to reflect the new expectation of the cell's activity.

However, BCM as originally formulated is based on non-spiking rate neurons. We implement BCM in biologically plausible spiking networks by interpreting neural activity as spikes that are filtered by a postsynaptic current curve.

$$\Delta\omega_{ij} = \kappa\alpha_j a_i a_j(a_j - \theta)$$
$$\theta(t) = e^{-t/\tau}\theta(t-1) + (1 - e^{-t/\tau}a_j(t)), \qquad (8)$$

where a, the activity of a neuron, is interpreted as a filtered spike train, as in Equation (3), and τ is the time constant of the modification threshold's exponential filter.

With our spiking BCM implementation, we aim to test the claim that BCM is equivalent to triplet STDP rules. Functionally, we hypothesize that unsupervised learning will have a small detrimental effect on the function being computed by a weight matrix, but will result in weight sparsification.

Simultaneous supervised and unsupervised learning: hPES rule

The PES rule gives us the ability to minimize some provided error signal, allowing a network to learn to compute a transformation online. However, biological synapses can change when no error signal is present. More practically, transformation learning may be easier in more sparse systems. For these reasons, we propose a new learning rule that combines the error-minimization abilities of the PES rule with the biological plausibility and sparsification of the spiking BCM rule. The rule is a weighted sum of the terms in each rule.

$$\Delta\omega_{ij} = \kappa\alpha_j a_i(S\mathbf{e}_j \cdot \mathbf{E} + (1-S)a_j(a_j - \theta)), \qquad (9)$$

where $0 \leq S \leq 1$ is the relative weighting of the supervised term over the unsupervised term. Note that this rule is a generalization of the previously discussed rules; if we set $S = 1$, this rule is equivalent to PES, and if we set $S = 0$, this rule is equivalent to spiking BCM.

We hypothesize that the unsupervised component helps maintain homeostasis while following the error gradient defined by the supervised component. Because of this, we call the rule the homeostatic Prescribed Error Sensitivity (hPES) rule. We hypothesize that this rule will be able to learn the same class of transformations that the PES rule can learn, and that this class includes operations critical to cognitive representation, such as semantic pointer binding.

Note that a similar combination can be done with other supervised learning rules that modify connection weights. A more general form of Equation (9) would replace the local error quantity $\mathbf{e}_j \cdot \mathbf{E}$ with δ, which could be determined through

any method. However, for clarity, we will use hPES to refer to the specific form in Equation (9).

Methods

We performed three experiments in order to test our hypotheses about the hPES rule. Experiments were implemented in the Nengo simulation environment, in which we implemented the PES, spiking BCM, and hPES learning rules. All experiments use leaky integrate-and-fire neurons with default parameters. The scripts used to implement and analyze these experiments are MIT licensed and available at http://github.com/tbekolay/cogsci2013.

Experiment 1: Unsupervised learning We constructed a network composed of two populations connected in a feedforward manner such that one population provides input to the other. The network can be run in a "control" regime, in which the weights between the two populations are solved for with the NEF's least-squares optimization and do not change, or in a "learning" regime, in which the weights are the same as the control network, but change over time according to the hPES rule (9) with $S = 0$ (i.e., according to the spiking BCM rule [8]). This experiment tests the hypothesis that the unsupervised component of the hPES rule increases sparsity of the connection weight matrix.

Experiment 2: Supervised learning We constructed a network composed of two populations connected in a feedforward manner, and one population that provides an error signal to the downstream population. The network can be run in a "control" regime, in which the weights between the two populations are computed to transmit a three-dimensional semantic pointer, or to bind two three-dimensional semantic pointers into one three-dimensional pointer. The network can be run in a "learning" regime, in which the weights between the two populations are initially random and are modified over time by the hPES rule. This experiment tests the hypothesis that the hPES rule can learn to transmit and bind semantic pointers as well as the control network and the supervised learning rule (i.e., hPES with $S = 1$).

Experiment 3: Digit classification In order to investigate how simultaneous supervised and unsupervised learning scales in higher dimensional situations, we constructed a network similar to that in Experiment 2, but whose input is handwritten digits.* In order to be computationally tractable, the 28-by-28 pixel images were compressed to a 50-dimensional semantic pointer using a sparse deep belief network that consists of four feedforward Restricted Boltzmann Machines trained with a form of contrastive divergence (full details in Tang & Eliasmith, 2010). Those 50-dimensional pointers were projected to an output population of 10 dimensions, where each dimension represents the confidence that the input representation should be classified into one of the 10 possible digits. The classified digit is the one corresponding

---

*Digits from MNIST: http://yann.lecun.com/exdb/mnist/

to the dimension with the highest activity over 30 ms when a 50-dimensional input is presented. 60,000 labeled training examples were shown to the network while the hPES rule was active. The network was then tested with 10,000 training examples in which the label was not provided. The results are compared to an analogous control network, in which the 50-dimensional pointers are classified with a cleanup memory whose connection weights are static, as in Eliasmith et al. (2012). This experiment examines how well the hPES rule scales to high-dimensional spaces.

Learning parameters While there are many hundreds of parameters involved in each network simulation, the vast majority are randomly selected within a biologically plausible range without significantly affecting performance. Some parameters, especially those affecting the learning rule, can have a significant performance effect. These significant parameters and the values used in specific simulations are listed in Tables 1 and 2. These parameters were optimized with a tree of Parzens estimators approach, using the `hyperopt` package (Bergstra, Yamins, & Cox, 2013).

Table 1: Parameters used for transmitting semantic pointers

Parameter	Description	Value
N/D	Neurons per dimension	25
κ	Learning rate	3.51×10^{-3}
S	Supervision ratio	0.798
κ for $S=1$	Learning rate (PES)	2.03×10^{-3}

Table 2: Parameters used for binding and classifying SPs

Parameter	Description	Value
N/D	Neurons per dimension	25
κ	Learning rate	2.38×10^{-3}
S	Supervision ratio	0.725
κ for $S=1$	Learning rate (PES)	1.46×10^{-3}

Results

hPES replicates STDP results

Previously, Pfister and Gerstner (2006) have theorized that BCM and STDP are equivalent. Our experiments support this theory. Varying the amount of time between presynaptic and postsynaptic spikes results in an STDP curve extremely similar to the classical Bi and Poo (2001) STDP curve (Figure 1).

However, these STDP curves do not capture the frequency dependence of STDP. In order to capture those effects, modellers have created STDP rules that take into account triplets and quadruplets of spikes, rather than just pre-post spike pairings (Pfister & Gerstner, 2006). These rules are able to replicate the frequency dependence of the STDP protocol. Figure 2 shows that, despite being a much simpler rule, the hPES rule with $S=0$ also exhibits frequency dependence.
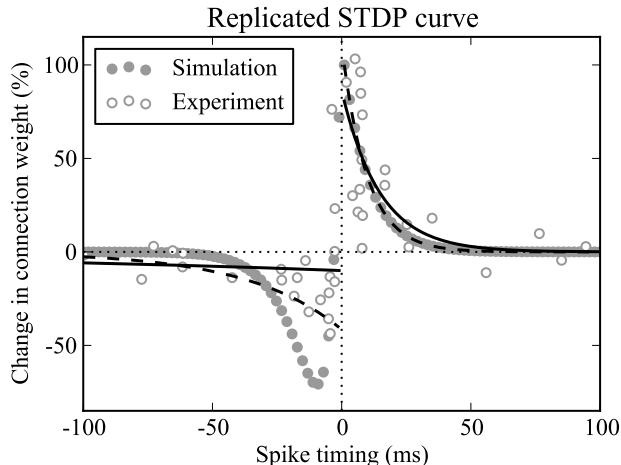

Figure 1: STDP curve replicated by two neurons connected with the hPES rule, $S=0$. Solid and dashed lines are best fits of the curve $ae^{-x/\tau}$ for the experimental and simulated data, respectively. Experimental data from Bi and Poo (2001).

hPES encourages sparsity while increasing signal transmission error

When the hPES rule is applied to a network that has been optimized to implement semantic pointer transmission, the hPES rule with no supervision ($S=0$) increases signal transmission error at a rate proportional to the learning rate. Figure 3 (top) shows the gradual decrease in accuracy over 200 seconds of simulation with an artificially large learning rate. Figure 3 (bottom) shows that the sparsity of the weight matrix, as measured by the Gini index, increases over time. Therefore, the hPES rule with $S=0$ increases network sparsity at the cost of an increase in signal transmission error.

hPES can learn cognitive functions

The error in the learned networks relative to the mean of 10 control networks can be seen decreasing over time in Figure 4. The parameters used for Figure 4 are listed in Tables 1 and 2. The transformations are learned quickly (approximately 25 seconds for transmission, 45 seconds for binding). Therefore, the central binding and transmission operations of the SPA are learnable with the hPES rule.

Critically, while hPES without error sensitivity introduces error while increasing sparsity (see Figure 3), with error sensitivity, this error can be overcome. Interestingly, binding, the intuitively more complex operation, is more reliably learned than transmission. This is due to how effectively the NEF can optimize weights to perform linear transformations like transmission in the control networks.

As a proof of concept that the hPES rule scales to high-dimensional spaces, Table 3 shows that the learned handwritten digit classification network classifies digits more accurately than Spaun's cleanup memory (Eliasmith et al., 2012).
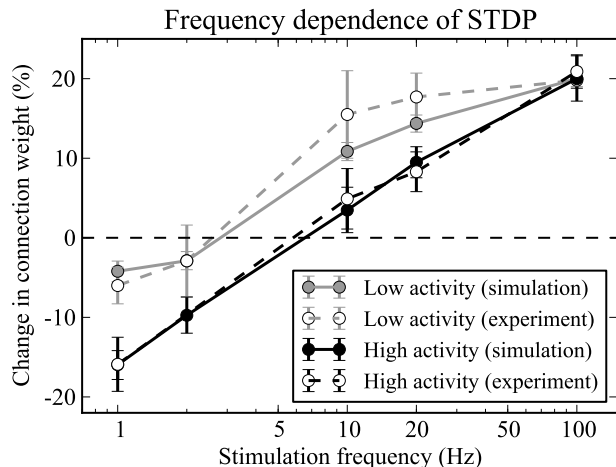
Figure 2: STDP frequency dependence replicated by two neurons connected with the hPES rule, $S = 0$. This also demonstrates the effect of different θ values on the frequency dependence curve. Experimental data from Kirkwood et al. (1996).

This supports the suggestion that the hPES rule scales to high-dimensional spaces. While the hPES rule with $S < 1$ achieved higher classification accuracy than hPES with $S = 1$, not enough trials were attempted to statistically confirm a benefit to combined unsupervised and supervised learning for classifying handwritten digits.

Table 3: Classification accuracy of handwritten digits

Classification technique	Accuracy
Cleanup memory (Spaun)	94%
hPES learning, $S = 1$	96.31%
hPES learning	98.47%

hPES is less sensitive to parameters

The parameters of the hPES rule were optimized with S fixed at 1 for 50 simulations of binding and transmission. A separate parameter optimization that allowed S to change was done for 50 simulations of binding and transmission.

Surprisingly, despite optimizing over an additional dimension, when S was allowed to change, error rates were lower during the optimization process for the binding network but not the transmission network. In both cases, the interquartile range of the hPES rule's performance when S was allowed to change is lower. Figure 5 summarizes the performance of all 200 networks generated for parameter optimization. While in all four cases parameters were found that achieve error rates close to the control networks, hPES was more robust to changes in parameters when S was allowed to change. This suggests that unsupervised learning may be beneficial in high-dimensional nonlinear situations.
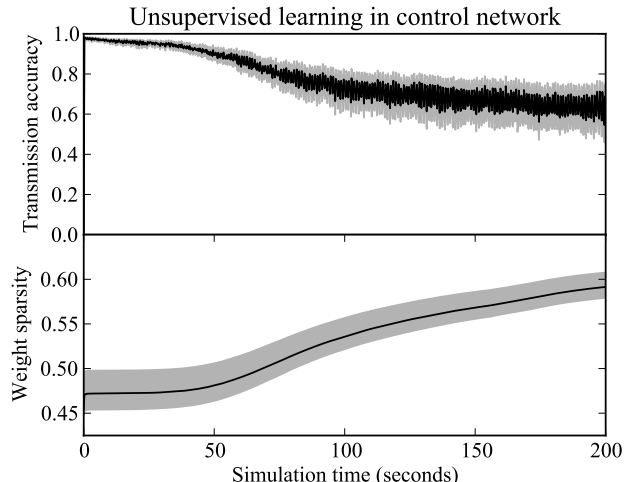

Figure 3: Data gathered from 50 trials of Experiment 1; filled region represents a bootstrapped 95% confidence interval. (Top) Accuracy of signal transmission. Accuracy is proportional to negative mean squared error, scaled such that accuracy of 1.0 denotes a signal identical to that transmitted by the NEF optimal weights with no learning, and accuracy of 0.0 represents no signal transmission (i.e., the error is equal to the signal). (Bottom) Sparsity of the connection weight matrix over time, as measured by the Gini index. This demonstrates the expected tradeoff between sparsity and accuracy.

Discussion

In this paper, we have presented a novel learning rule for learning cognitively relevant transformations of neural representations in a biologically plausible manner. We have shown that the unsupervised component of the rule increases sparsity of connection weights at the cost of increased signal transmission error. We have also shown that the combined learning rule, hPES, can learn transformations as well as the supervised rule and the offline optimization done in the Neural Engineering Framework. We have demonstrated that the combined learning rule is more robust to changes in parameters when learning nonlinear transformations.

However, it is still the case that the parameters of the learning rule were optimized for each transformation learned. This is a challenge shared by all learning rules, but in the context of biologically plausible simulations, there is the additional question of the biological correlate of these parameters. It could be the case that these parameters are a result of the structure of the neuron, and therefore act as a fixed property of the neuron. However, it could also be the case that these parameters are related to the activity of the network, and are modified by each neuron's activity, or by the activity of some external performance monitoring signal. Examining these possibilities is the subject of future work.
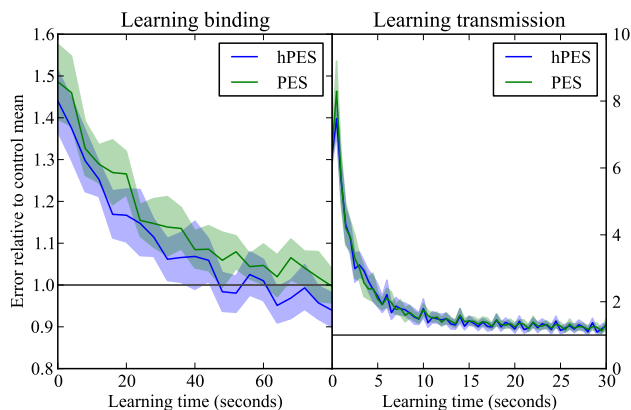
Figure 4: Error of learning networks over time compared to the mean error of 10 control networks. Each type of network is generated and simulated 15 times. For the binding network, every 4 seconds, the learning rule is disabled and error is accumulated over 5 seconds. For the transmission network, every 0.5 seconds, the learning rule is disabled, and error is accumulated over 2 seconds. Filled regions are bootstrapped 95% confidence intervals. Time is simulated time, not computation time.
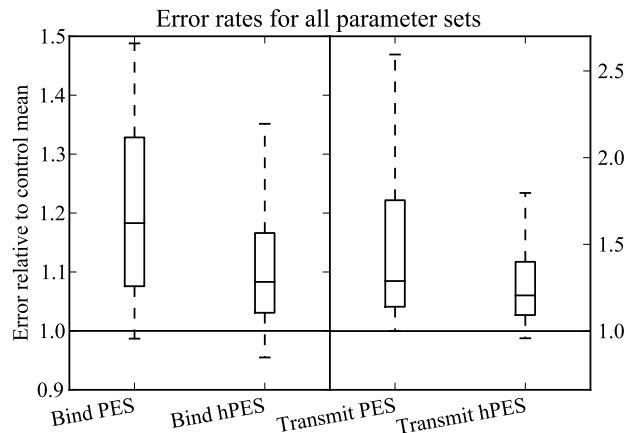
Figure 5: Summary of error rates for networks used to optimize learning parameters. Each column summarizes 50 experiments of the labeled condition. Boxes indicate the median and inter-quartile range, and whiskers indicate the inner fence (i.e., $Q_1 - 1.5 IQR$ and $Q_3 + 1.5 IQR$). Outliers are not shown.

References

Bergstra, J., Yamins, D., & Cox, D. D. (2013). Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. In *30th international conference on machine learning.*

Bi, G.-Q., & Poo, M.-M. (2001). Synaptic modification by correlated activity: Hebb's postulate revisited. *Annual Review of Neuroscience*, *24*(1), 139–66.

Bienenstock, E. L., Cooper, L. N., & Munro, P. (1982). Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex. *Journal of Neuroscience*, *2*(1), 32.

Bohte, S. M., Kok, J. N., & Poutre, H. L. (2002). SpikeProp: Backpropagation for networks of spiking neurons. *Neurocomputing*, *48*(1-4), 17.

Eliasmith, C. (in press). *How to build a brain: A neural architecture for biological cognition*. New York, NY: Oxford University Press.

Eliasmith, C., & Anderson, C. H. (2003). *Neural engineering: computation, representation, and dynamics in neurobiological systems*. Cambridge, MA: MIT Press.

Eliasmith, C., Stewart, T. C., Choo, X., Bekolay, T., DeWolf, T., Tang, Y., Tang, C., & Rasmussen, D. (2012). A large-

scale model of the functioning brain. *Science*, *338*(6111), 1202–5.

Georgopoulos, A. P., Schwartz, A. B., & Kettner, R. E. (1986). Neuronal population coding of movement direction. *Science*, *233*(4771), 1416–1419.

Hinton, G., Osindero, S., & Teh, Y. (2006). A fast Learning Algorithm for Deep Belief Nets. *Neural computation*, *1554*, 1527–1554.

Kirkwood, A., Rioult, M. G., & Bear, M. F. (1996). Experience-dependent modification of synaptic plasticity in visual cortex. *Nature*, *381*(6582), 526–528.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, *43*(1).

Lazar, A., Pipa, G., & Triesch, J. (2009). Sorn: a self-organizing recurrent neural network. *Frontiers in Computational Neuroscience*, *3*(23).

MacNeil, D., & Eliasmith, C. (2011). Fine-tuning and the stability of recurrent neural networks. *PloS One*, *6*(9).

Paugam-Moisy, H., Martinez, R., & Bengio, S. (2008). Delay learning and polychronization for reservoir computing. *Neurocomputing*, *71*(7), 1143–1158.

Pfister, J.-P., & Gerstner, W. (2006). Triplets of spikes in a model of spike timing-dependent plasticity. *Journal of Neuroscience*, *26*(38), 9673–82.

Ponulak, F. (2006). *Supervised learning in spiking neural networks with ReSuMe method*. Phd thesis, Poznan University of Technology.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536.

Tang, Y., & Eliasmith, C. (2010). Deep networks for robust visual recognition. In *27th international conference on machine learning.* Haifa, Il: ICML.