Biologically feasible deep learning with segregated dendrites

Jordan Guergiuev[1,2], Timothy P. Lillicrap[3], Blake A. Richards[1,2,*]

1 Department of Biological Sciences, University of Toronto Scarborough, Toronto, ON, Canada
2 Department of Cell and Systems Biology, University of Toronto, Toronto, ON, Canada
3 Department of Pharmacology, University of Oxford, Oxford, UK

* Corresponding author, email: blake.richards@utoronto.ca

Abstract

Deep learning has led to significant advances in artificial intelligence in recent years, in part by adopting architectures and functions motivated by neurophysiology. However, current deep learning algorithms are biologically infeasible, because they assume non-spiking units, discontinuous-time, and non-local synaptic weight updates. Here, we build on recent discoveries in artificial neural networks to develop a spiking, continuous-time neural network model that learns to categorize images from the MNIST data-set with local synaptic weight updates. The model achieves this via a three-compartment cellular architecture, motivated by neocortical pyramidal cell neurophysiology, wherein feedforward sensory information and feedback from higher layers are received at separate compartments in the neurons. We show that, thanks to the separation of feedforward and feedback information in different dendrites, our learning algorithm can coordinate learning across layers, taking advantage of multilayer architectures to identify abstract categories—the hallmark of deep learning. Our model demonstrates that deep learning can be achieved within a biologically feasible framework using segregated dendritic compartments, which may help to explain the anatomy of neocortical pyramidal neurons.

Introduction

In recent years, deep learning in artificial neural networks has revolutionized machine learning, opening the door to artificial intelligence applications that can rival human capabilities in pattern recognition and control (Mnih et al., 2015; Kubilius et al., 2016; Silver et al., 2016; He et al., 2015). The key to deep learning is the ability to coordinate weight updates across multiple layers, which allows deep neural networks to construct hierarchical representations wherein each successive layer identifies more abstract, relevant variables for a given task (LeCun et al., 2015). Interestingly, deep networks generate representations that resemble those observed in the neocortex, particularly in higher-order sensory areas (Kubilius et al., 2016; Khaligh-Razavi and Kriegeskorte, 2014; Cadieu et al., 2014), suggesting that something akin to deep learning is occurring in the mammalian brain (Yamins and DiCarlo, 2016; Marblestone et al., 2016).

However, there are several ways in which current deep learning algorithms, such as back-propagation of error (Rumelhart et al., 1986), are wildly unrealistic from a biological perspective (Bengio et al., 2015). First, most deep learning algorithms rely on non-local synaptic weight updates, wherein synapses in lower layers of a network are updated using information about the synapses in the higher layers (Bengio et al., 2015; Lillicrap et al., 2014). This is completely infeasible from a biological standpoint, as it would require early processing areas (e.g. V1) to have precise information about the *billions* of synaptic connections in higher-order areas (V2, V4, etc.). According to our current understanding, this level

of non-local communication is well-beyond animal physiology. Second, most deep learning algorithms utilize non-spiking units that transmit real numbers (LeCun et al., 2015). To be biologically feasible, deep learning must be shown to work with neurons that communicate with discrete spikes, and at the relatively slow rates (tens of Hz) observed in the neocortex (Steriade et al., 2001). Finally, deep learning algorithms operate with discontinuous time, such that activity flows through the network layer-by-layer (Bengio et al., 2015). Obviously, in the real brain, neurons' voltages fluctuate constantly in real-time, and to be biologically realistic deep learning would have to work in networks that communicate in continuous-time. Altogether, these artificial aspects of deep learning algorithms have rendered some scientists skeptical to the proposal that deep learning occurs in the real brain (Crick, 1989; Grossberg, 1987).

Recent findings in neural networks have shown that these problems may be surmountable, though. First, Lee et al. (2015) demonstrated that it is possible to use feedback from higher layers to construct *local* targets for neurons' activity that coordinate synaptic updates across layers (Lee et al., 2015). Their algorithm, which they called "difference target propagation", eliminates the need for non-local computation in synaptic updates (Lee et al., 2015). Similarly, Lillicrap et al. (2014) recently demonstrated that non-local updates can be avoided by propagating error signals backwards through *random* synaptic weight matrices (Lillicrap et al., 2014). This algorithm, which the authors called "feedback alignment", comes to approximate a Gauss-Newton update mechanism, and thus, it can produce learning that is actually better than that produced by back-propagation in some circumstances (Lillicrap et al., 2014). Similar conclusions were reached by Liao et al. (2015) (though they found that the sign of the feedback weights could not be fully randomized). These studies are remarkable as they have shown that deep learning, i.e. coordinated learning across multiple layers (LeCun et al., 2015), can be achieved with purely local weight updates as long as higher layers send back some information about their activity to lower layers. This would suggest that the feedback from higher order sensory areas that arrives in the distal apical dendrites of pyramidal neurons in primary sensory areas (Manita et al., 2015; Budd, 1998; Spratling, 2002) may provide the necessary information for deep learning in biological neural networks (Körding and König, 2000). However, there has yet to be a demonstration that these ideas can be extended into spiking networks that operate in continuous time.

Here, we show how a biologically feasible version of Lee et al.'s (2015) difference target propagation algorithm can be implemented using spiking neurons that operate in continuous-time, if the neurons contain separate "basal" and "apical" dendritic compartments for integrating feedforward and feedback signals, respectively. Our model builds on previous computational studies of supervised learning in multi-compartment neurons (Urbanczik and Senn, 2014; Körding and König, 2000), but importantly, we use the distinct basal and apical compartments to build a local target for each layer that coordinates learning across the layers of the network. We demonstrate that even with random synaptic weights for feedback into the apical compartment, our algorithm can produce high levels of accuracy in classification of the MNIST database of hand-written digits, similar to the error rates achieved by vanilla back-propagation of error (LeCun et al., 1998). Furthermore, we show that our algorithm allows the network to take advantage of multilayer structures to build hierarchical, abstract representations, one of the hallmarks of deep learning (LeCun et al., 2015). Our results demonstrate that deep learning can be implemented in a biologically feasible manner if feedforward and feedback signals are received at electrotonically segregated dendrites.

Results

A network architecture with segregated dendritic compartments

In order to achieve deep supervised learning with local weight updates, neurons in each layer must somehow ensure that their weight updates are sensible given how the changes will impact the error at the output layer. This sort of coordination can be achieved in point neurons in continuous time using energy-based models (Scellier and Bengio, 2016), but energy-based models require symmetric weight matrices in feedforward and feedback connections, which is effectively as biologically implausible as

non-local synaptic weight updates. However, in the difference target propagation algorithm, Lee et al. (2015) demonstrated that non-symmetric feedback weights can be used to construct appropriate local targets for a hidden layer of neurons. To do this, though, Lee et al. (2015) assumed that feedforward activity and feedback activity were distinct. In a point neuron that operates in continuous time this would be difficult to achieve, because the feedforward and feedback information would be integrated into a single value.

In considering how the real brain may address this issue, we were inspired by two observations: (1) in the neocortex, feedforward sensory information and higher-order feedback are largely received by distinct dendritic compartments, namely the basal dendrites and distal apical dendrites, respectively (Spratling, 2002; Budd, 1998), (2) the distal apical dendrites are electrotonically distant from the soma, such that transmission from the distal dendrites to the soma is significantly attenuated in the absence of non-linear calcium spikes in the apical dendritic shaft that can drive burst firing at the soma (Larkum et al., 1999, 2009). With these physiological realities in mind, we hypothesized that the computation required for difference target propagation could be achieved by having two distinct dendritic compartments in each hidden layer neuron: a "basal" compartment, strongly coupled to the soma for integrating feedforward information, and an "apical" compartment for integrating feedback information that would only drive activity at the soma when apical spikes occurred (**Fig. 1A**). As an initial test of this concept we built a network with a single hidden layer, as this was more mathematically tractable. Although a single hidden layer is not very "deep", the addition of a hidden layer can only improve performance in the context of a deep learning algorithm that coordinates learning in the output and hidden layers (Lillicrap et al., 2014). Hence, we wanted to initially determine whether our network could take advantage of the hidden layer to reduce error at the output layer.

The network architecture is illustrated in **Fig. 1A**. An image from the MNIST data set is used to set the spike rates of a set of Poisson point-process neurons in the input layer (one unit per image pixel, rates-of-fire determined by pixel intensity). These project to the hidden layer, which is composed of model neurons with three distinct compartments/voltages, the apical compartments (with voltages $\mathbf{A}(t)$), the basal compartments (with voltages $\mathbf{B}(t)$), and the somatic compartments (with voltages $\mathbf{C}(t)$). The somata generate spikes using Poisson processes whose instantaneous rates, $\boldsymbol{\lambda}_C(t)$, are determined by a non-linear function, $\phi(\cdot)$, applied to the somatic voltages. Spiking inputs from the input layer arrive at the basal compartments via the synaptic weight matrix W^0, and spiking inputs from the output layer arrive at the apical compartment via the synaptic weight matrix Y. In both apical and basal compartments, the effect of a presynaptic spike on a the postsynaptic voltage is determined by convolving the spike train with a double exponential function (see Methods). The hidden layer somatic spikes are projected to the output layer neurons via the synaptic weight matrix W^1. The output layer neurons consist of 10 two compartment neurons (one for each image category) similar to those used in a previous model of dendritic prediction learning (Urbanczik and Senn, 2014). The output dendritic voltages ($\mathbf{V}(t)$) and somatic voltages ($\mathbf{U}(t)$) are updated in a similar manner to the hidden layer basal compartment and soma, respectively. As well, like the hidden layer neurons, the output neurons spike using Poisson processes whose instantaneous rates, $\boldsymbol{\lambda}_U(t)$, are determined by the somatic voltages. Unlike the hidden layer, however, the output layer neurons can also receive a set of "teaching signals", specifically excitatory and inhibitory conductances that push their voltages towards target values. The dynamics for the voltages in all the neurons/compartments in the model are leaky and continuous. Mathematical details of the simulation are given in the Methods, and can also be seen by downloading the code for the model from a github repository (https://github.com/jordan-g/Segregated-Dendrite-Deep-Learning).

Critically, we define two different modes of integration in the hidden layer neurons: "transmit" and "burst". During the transmit mode, the apical compartment is considered to be electrotonically segregated from the soma, and so it does not affect the somatic voltage, only the basal compartments do (**Fig. 1B**, left). (We note that total segregation is unnecessary, though it makes the mathematics more tractable. Further on we relax this constraint and show that learning can still work with attenuated apical inputs.) During the burst mode, the apical voltage is averaged over the most recent 20-30 ms and a non-linearity, $\alpha(\cdot)$, is applied to it. This non-linear version of the averaged apical voltage is then transmitted through to the somatic/basal compartments (**Fig. 1B**, right). The intention behind this design was to partially

mimic the non-linear transmission from the apical dendrites to the soma that occurs during a calcium spike in the apical dendritic shaft of neocortical pyramidal neurons (Larkum et al., 1999). Furthermore, the temporal averaging was intended to mimic, in part, the temporal dynamics introduced by NMDA plateaus (Schiller et al., 2000), which are particularly good for driving apical spikes (Larkum et al., 2009).

To train the network we alternate between two phases. First, we present an image to the input layer without any signals to the output layer during the "forward" phase. Following a burst, the "target" phase begins. During this phase the image continues to drive the input layer, but now the output layer also receives teaching signals, which force the output units closer to the correct answer. For example, if an image of a '9' is presented, then the '9' neuron in the output layer receives strong excitatory conductances from the teaching signal, while the other neurons receive strong inhibitory conductances (**Fig. 1C**). The target phase is also ended by a burst. As mentioned, the network is simulated in continuous-time (each burst is considered to be effectively instantaneous), and the occurrence of the bursts are randomly sampled from an inverse Gaussian distribution (**Fig. 1D**, top). As such, the specific amount of time that the network is presented with each image and teaching signal is stochastic, though usually somewhere between 50-60 ms of simulated time (**Fig. 1D**, bottom). In the data presented in this paper, all 60,000 images in the MNIST training set were presented to the network one at a time, and each exposure to the full set of images was considered an "epoch" of training. At the end of each epoch, the network's classification error rate on a separate set of 10,000 test images was assessed with a single forward phase.

It is important to note that there are many aspects of this design that are not physiologically accurate. Most notably, stochastic generation of bursts is not an accurate reflection of how real pyramidal neurons operate, since apical calcium spikes and somatic bursts are determined by a number of concrete physiological factors, including back-propagating action potentials, spike-timing and inhibitory inputs (Larkum et al., 1999, 2007, 2009). However, we note that calcium spikes in the apical dendrites can be prevented from occurring via the activity of distal dendrite targeting somatostatin (SST) positive inhibitory interneurons (Murayama et al., 2009), which themselves can be rapidly inhibited via vasoactive intestinal polypeptide (VIP) inhibitory interneurons (Pi et al., 2013; Pfeffer et al., 2013; Karnani et al., 2016), in response to temporally precise neuromodulatory inputs (Hangya et al., 2015). Therefore, it is entirely plausible that neocortical micro-circuits would generate pyramidal bursts at punctuated periods of time in response to disinhibition of the apical dendrites governed by neuromodulatory signals that determine "phases" of processing. Of course, this may not be the case, but the point is that the arrangement we introduce here is not biologically *infeasible* in the way that previous deep learning models have been. As well, importantly for the present study, the use of segregated dendrites and distinct transmit and burst modes provides the mathematical tractability that allows us to guarantee coordination between local hidden layer updates and output layer errors.
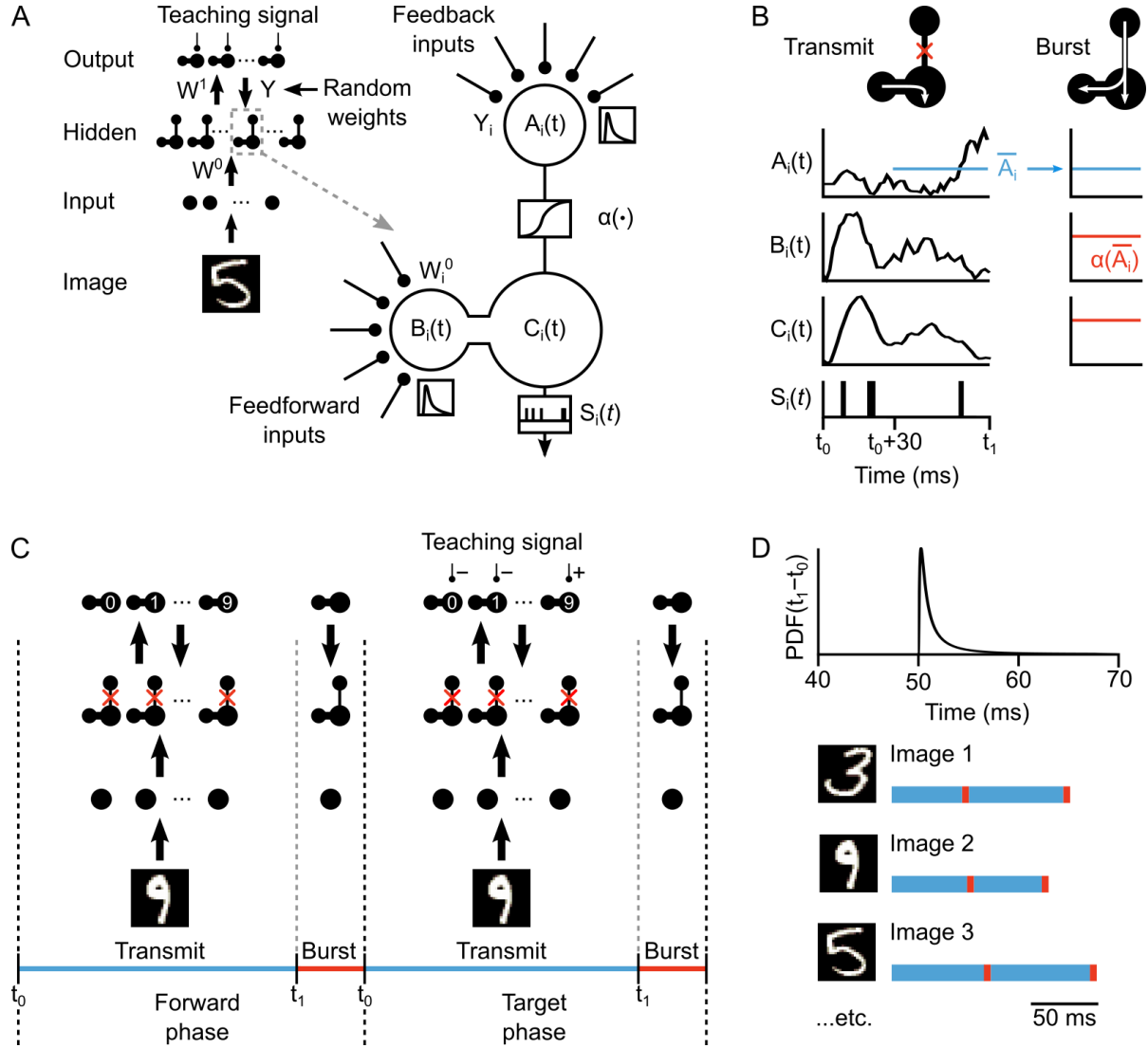
Figure 1. **A.** *Left*: Diagram of our network architecture. An input image is represented by the spiking of input units which propagate to the hidden layer through weights W^0. Hidden layer activities arrive at the output layer through weights W^1. Feedback from the output layer is sent back to the hidden layer through fixed, random weights Y. *Right*: Illustration of unit i in the hidden layer. The unit has a basal dendrite, soma, and apical dendrite with membrane potentials B_i, C_i and A_i, respectively. The apical dendrite communicates to the soma only using non-linear signals $\alpha(\cdot)$. The spiking output of the cell, $S_i(t)$, is a Poisson process with rate $\phi(C_i(t))$. **B.** Illustration of neuronal dynamics in the two modes of processing. In the transmit mode, the apical dendrite accumulates a measure of its average membrane potential beginning at 30 ms after t_0. In the burst mode, an apical burst equal to the nonlinearity α applied to the average apical potential travels to the soma and basal dendrite. **C.** Illustration of the sequence of network phases that occur for each training example. The network undergoes a forward phase (transmit & burst) and a target phase (transmit & burst) in sequence. In the target phase, a teaching signal is introduced to the output layer, shaping its activity. **D.** Illustration of phase length sampling. For each training example, the lengths of forward & transmit phases are randomly drawn from a shifted inverse Gaussian distribution with a minimum of 50 ms.

Coordinating synaptic weight updates across layers with segregated dendrites

To achieve deep learning with local synaptic weight updates, we had to define local error functions for each layer that were somehow guaranteed to be compatible. To accomplish this, we defined local targets for the hidden layer similar to those used by Lee et al. (2015). These targets coordinate with the output layer by incorporating the feedback information propagated to the somata during the bursts. Specifically, we define the target rates-of-fire for the output layer to be the rates-of-fire during the target transmit phase, $\hat{\boldsymbol{\lambda}}_U = \overline{\boldsymbol{\lambda}}_U^t$, and for the hidden layer we define the local target, $\hat{\boldsymbol{\lambda}}_C$ by:

$$\hat{\boldsymbol{\lambda}}_C = \phi(\overline{\boldsymbol{C}}^f) + \boldsymbol{\alpha}^t - \boldsymbol{\alpha}^f \tag{1}$$

...where $\overline{\boldsymbol{C}}^f$ is the vector of average somatic voltages during the forward transmit phase, and $\boldsymbol{\alpha}^t$ and $\boldsymbol{\alpha}^f$ represent the burst activities during the target and forward bursts, respectively. For both the output layer and the hidden layer we then define error functions based on the difference between the actual activity of the neurons during the forward transmission phase and their targets (**Fig. 2A**):

$$L^1 = ||\hat{\boldsymbol{\lambda}}_U - \phi(\overline{\boldsymbol{U}}^f)||_2^2$$
$$L^0 = ||\hat{\boldsymbol{\lambda}}_C - \phi(\overline{\boldsymbol{C}}^f)||_2^2 \tag{2}$$

...where L^1 and L^0 are the output layer and hidden layer error, respectively, while $\overline{\boldsymbol{U}}^f$ is the average somatic voltages the output and hidden layer neurons during the forward phase. It can be shown that these error functions will agree with each other on average. Specifically, it can be shown that if the output layer error is sufficiently small, then:

$$||\hat{\boldsymbol{\lambda}}_U - \phi(W^1\hat{\boldsymbol{\lambda}}_C)||_2^2 < ||\hat{\boldsymbol{\lambda}}_U - \phi(E[\overline{\boldsymbol{U}}^f])||_2^2 \tag{3}$$

See Theorem 1 in the Methods for the proof. In plain language, equation (3) says that when we consider the difference between the output target and the activity that the hidden target *would* have produced at the output layer, it is less that the difference between the output target and the expected value of the output layer activity during the forward phase. In other words, the output layer's error would have, on average, been *smaller* if the hidden layer had achieved its own target during the forward phase.

With these error functions, we update the weights in the network with local gradient descent:

$$\Delta W^1 \propto \frac{\partial L^1}{\partial W^1}$$
$$\Delta W^0 \propto \frac{\partial L^0}{\partial W^0} \tag{4}$$

...where ΔW^i refers to the update term for weight matrix W^i (see Methods for details on the learning algorithm). Given the coordination implied by equation (3), as the hidden layer reduces its own local error with gradient descent, the output layer's error should also be reduced, i.e. hidden layer learning should imply output layer learning.

To test this, and provide empirical support for our proof, we compared the local error at the hidden layer to the output layer error across all of the image presentations to the network. We observed that, generally, whenever the hidden layer error was low, the output layer error was also low. For example, when we consider the errors for the set of '9' images presented to the network during the second epoch, there was a Pearson correlation coefficient between the hidden layer error and the output layer error of $r = 0.61$, which was much higher than what was observed for shuffled data, wherein output and hidden

activities were randomly paired (**Fig. 2B**). Furthermore, these correlations were observed across all epochs of training, with most correlation coefficients for the hidden and output errors falling between $r = 0.2 - 0.6$, which was again, much higher than the correlations observed for shuffled data (**Fig. 2BC**). Interestingly, the correlations were smaller on the first epoch of training (though still higher than the shuffled data). This suggests that the coordination between the layers may only come into full effect once the output layer has engaged in some learning, which make sense given the requirement in the proof that the output layer error be sufficiently small. Altogether, our results demonstrate that by segregating the feedforward and feedback information, we can define local targets for the hidden layer neurons that are guaranteed, on average, to improve the classification error at the output layer.
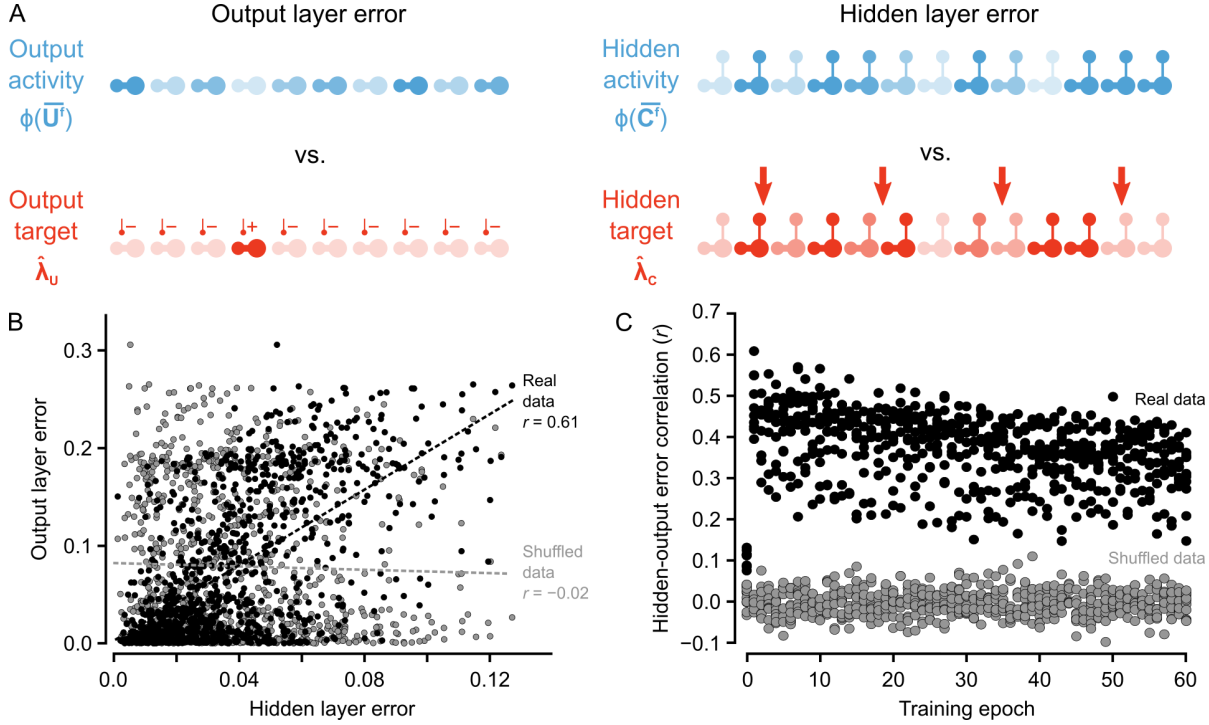
Figure 2. A. Illustration of output error and local hidden error. For a given test example shown to the network in a forward phase, the output layer error is defined as the squared norm of the difference between target firing rates $\hat{\boldsymbol{\lambda}}_U$ and the firing rate function ϕ applied to the average somatic potentials $\overline{\boldsymbol{U}}$ across all output units. Hidden layer error is defined similarly, except the target is $\hat{\boldsymbol{\lambda}}_C$ (as defined in the text). **B.** Plot of output vs. hidden layer error for all of the '9' images after one epoch of training. There is a strong correlation between hidden layer error and output layer error (real data, black), as opposed to when output and hidden errors were randomly paired (shuffled data, gray). **C.** Plot of correlation between hidden layer error and output layer error across training for each category of images (each dot represents one category). Note that the correlation is significantly higher in the real data than the shuffled data throughout training.

Learning with segregated dendrites produces deep learning

Given our finding of coordinated error rates in the hidden and output layers of the network, we had reason to believe that our spiking, continuous-time network with local weight-updates would exhibit deep learning, i.e. an ability to take advantage of a multi-layer structure. To test this, we examined the effects of including hidden layers. If deep learning is indeed operational in the network, then the inclusion of hidden layers should improve the ability of the network to classify images
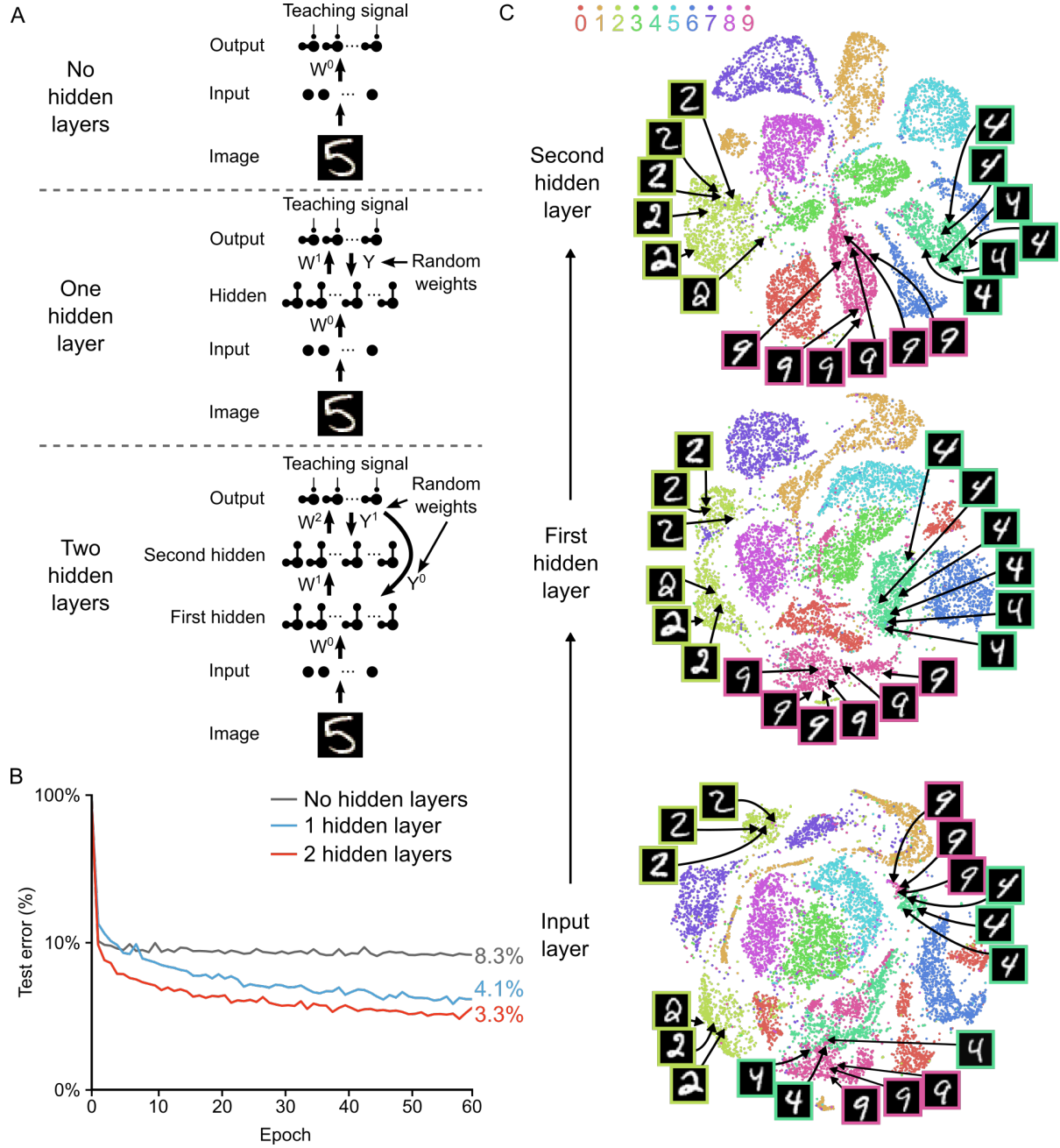
We built three different versions of the network (**Fig. 3A**). The first was a network that had no hidden layer, i.e. the input neurons projected directly to the dendrites of the output neurons. The

second was the network illustrated in **Figure 1**, with a single hidden layer. The third contained two hidden layers, with the output layer projecting directly back to both hidden layers. This direct projection allowed us to build our local targets for each hidden layer using the instantaneous bursts, i.e. avoiding a "backward pass" through the entire network as has been used in other models (Lillicrap et al., 2014; Lee et al., 2015). We trained each network on the 60,000 MNIST training images for 60 epochs, and recorded the percentage of images in the 10,000 test image set that were incorrectly classified. The network with no hidden layers rapidly learned to classify the images, but its error rate also rapidly plateaued at 8.3% (**Fig. 3B**, gray line). In contrast, the network with one hidden layer did not exhibit a rapid plateau of its error rate. Instead, it continued to improve throughout all 60 epochs, achieving an error rate of 4.1% by the 60th epoch (**Fig. 3B**, blue line). This is in-line with the error rates on MNIST achieved by simple, one hidden layer neural networks trained with back-propagation to reduce mean-squared error (LeCun et al., 1998). Interestingly, we found that the addition of a second hidden layer further improved learning. The network with two hidden layers learned more rapidly than the network with one hidden layer and achieved an error rate of 3.3% on the test images by the 60th epoch (**Fig. 3B**, red line). However, it should be noted that additional hidden layers beyond two did not significantly improve the error rate (data not shown), which suggests that our particular algorithm could not be used to construct very deep networks. Nonetheless, our network was clearly able to take advantage of multi-layer architectures to improve its learning, which is the key feature of deep learning (LeCun et al., 2015).

Another key feature of deep learning is the ability to generate abstract representations in the upper layers that capture task-relevant information while discarding sensory details (LeCun et al., 2015; Mnih et al., 2015). To examine whether our network exhibited this type of abstraction, we used the t-Distributed Stochastic Neighbor Embedding algorithm (t-SNE). The t-SNE algorithm reduces the dimensionality of data while preserving local structure and non-linear manifolds that exist in high-dimensional space, thereby allowing accurate visualization of the structure of the high-dimensional data (Maaten and Hinton, 2008). We applied t-SNE to the activity patterns at each layer of the two hidden layer network for all of the images in the test set after 60 epochs of training. At the input level, there was already some clustering of images based on their categories. However, the clusters were quite messy, with different categories showing outliers, several clusters, or merged clusters (**Fig. 3C**, bottom). For example, the '2' digits in the input layer exhibited two distinct clusters separated by a cluster of '7's: one cluster contained '2's with a loop and one contained '2's without a loop. Similarly, there were two distinct clusters of '4's and '9's that were very close to each other, with one pair for digits on a pronounced slant and one for straight digits (**Fig. 3C**, bottom, example images). Thus, although there is built-in structure to the categories of the MNIST dataset, there are a number of low-level features that do not respect category boundaries. In contrast, at the first hidden layer, the activity patterns were much cleaner, with far fewer outliers and split/merged clusters (**Fig. 3C**, middle). For example, the '2' digit clusters were much closer to each other and were now only separated by a very small cluster of '7's. Likewise, the '9' and '4' clusters were now distinct and no longer split based on the slant of the digit. Interestingly, when we examined the activity patterns at the second hidden layer the categories were even better segregated with only a bit of splitting or merging of category clusters (**Fig. 3C**, top). Therefore, the network had learned to develop increasingly abstract representations across its layers, such that by the second hidden layer the categories were very distinct and low-level features unrelated to the categories were largely ignored. This, abstract representation is likely to be key to the improved error rate in the two hidden layer network. Altogether, our data demonstrates that our spiking, continuous-time network with segregated dendritic compartments can achieve deep learning.

A

No hidden layers

Teaching signal

Output ● ● ● ⋯ ●
W^0 ↑
Input ● ● ⋯ ●
↑
Image 5

One hidden layer

Teaching signal

Output ● ● ● ⋯ ● ●
W^1 ↑ ↓ Y ← Random weights
Hidden ● ● ● ⋯ ● ●
W^0 ↑
Input ● ● ⋯ ●
↑
Image 5

Two hidden layers

Teaching signal

Output ● ● ● ⋯ ● ● ← Random weights
W^2 ↑ ↓ Y^1
Second hidden ● ● ● ⋯ ● ●
W^1 ↑ Y^0
First hidden ● ● ● ⋯ ● ●
W^0 ↑
Input ● ● ⋯ ●
↑
Image 5

B

Test error (%)

100%

No hidden layers
1 hidden layer
2 hidden layers

10%

8.3%
4.1%
3.3%

0%

0 10 20 30 40 50 60
Epoch

C

0 1 2 3 4 5 6 7 8 9

Second hidden layer

First hidden layer

Input layer

Figure 3. A. Illustration of the three networks used in the simulations. *Top*: a shallow network with only an input layer and an output layer. *Middle*: a network with one hidden layer. *Bottom*: a network with two hidden layers. Both hidden layers receive feedback from the output layer, but through separate synaptic connections with random weights Y^0 and Y^1. **B.** Plot of test error (measured on 10,000 MNIST images not used for training) across 60 epochs of training, for all three networks described in **A**. The networks with hidden layers exhibit deep learning, because hidden layers decrease the test error. **C.** Results of t-SNE dimensionality reduction applied to the activity patterns of the first three layers of a two hidden layer network (after 60 epochs of training). Each data point corresponds to a test image shown to the network. Points are color-coded according to the digit they represent. Moving up through the network, images from identical categories are clustered closer together and separated from images of different digits. Thus the hidden layers learn increasingly abstract representations of digit categories.

## Coordinated local learning mimics back-propagation of error

The back-progagation of error algorithm (Rumelhart et al., 1986) is still the primary learning algorithm used for deep supervised learning in artificial neural networks (LeCun et al., 2015). Previous work has shown that learning with random feedback weights can actually match the synaptic weight updates specified by the back-propagation algorithm after a few epochs of training (Lillicrap et al., 2014). This fascinating observation suggests that deep learning with random feedback weights is not so much a different algorithm than back-propagation of error, but rather, a way of learning to approximate back-propagation (Lillicrap et al., 2014). Hence, we were curious as to whether or not our network was, in fact, learning to approximate the synaptic weight updates prescribed by back-propagation. To test this, we trained our one hidden layer network as before, but now, in addition to calculating the vector of hidden layer synaptic weight updates specified by our local learning rule (ΔW^0 in equation (4)), we also calculated the vector of hidden layer synaptic weight updates that would be specified by non-locally back-propagating the error from the output layer, (ΔW_{BP}^0). We then calculated the angle between these two alternative weight updates. In a very high-dimensional space, any two independent vectors will be roughly orthogonal to each other (i.e. $\Delta W^0 \angle \Delta W_{BP}^0 \approx 90°$). If the two synaptic weight update vectors are *not* orthogonal to each other (i.e. $\Delta W^0 \angle \Delta W_{BP}^0 < 90°$), then it suggests that the two algorithms are specifying similar weight updates.

As in previous work (Lillicrap et al., 2014), we found that the initial weight updates for our network were orthogonal to the updates specified by back-propagation. But, as the network learned the angle dropped to approximately 65°, before rising again slightly and to roughly 70° (**Fig. 4A**, blue line). This suggests that our network was learning to develop local weight updates in the hidden layer that were in rough agreement with the updates that explicit back-propagation would produce. However, this drop in orthogonality was still much less than that observed in non-spiking, discrete time artificial neural networks learning with random feedback weights, which show a drop to below 45° (Lillicrap et al., 2014). We suspected that the higher angle between the weight updates that we observed may have been because we were using spikes to communicate the feedback from the upper layer, which could introduce both noise and bias in the estimates of the output layer activity. To test this, we also examined the weight updates that our algorithm would produce if we propagated the somatic voltages of the output layer neurons back directly through the random feedback weights. In this scenario, we observed a much sharper drop in the $\Delta W^0 \angle \Delta W_{BP}^0$ angle, which reduced to roughly 35° before rising again to 40° (**Fig. 4A**, red line). These results show that, in principle, our algorithm is learning to approximate the back-propagation algorithm, though with some drop in accuracy introduced by the use of spikes to propagate output layer activities to the hidden layer.

To further examine how out local learning algorithm compared to back-propagation we compared the low-level features that the two algorithms learned. To do this, we trained the one hidden layer network with both our algorithm and back-propagation. We then examined the receptive fields (i.e. the synaptic weights) produced by both algorithms in the hidden layer synapses (W^0) after 60 epochs of training. The two algorithms produced qualitatively similar receptive fields (**Fig. 4B**). Both produced receptive fields with clear, high-contrast features for detecting particular strokes or shapes. To quantify the similarity, we conducted pair-wise correlation calculations for the receptive fields produced by the two algorithms and identified the maximum correlation pairs for each. Compared to shuffled versions of the receptive fields, there was a very high level of maximum correlation (**Fig. 4C**), showing that the receptive fields were indeed quite similar. Thus, our data demonstrate that our learning algorithm using random feedback weights and spikes can in fact come to approximate the back-propagation of error algorithm.

Learning with partial apical attenuation or symmetric weights

Once we had convinced ourselves that our learning algorithm was, in fact, producing deep learning similar to that produced by back-propagation of error, we wanted to examine some of the constraints on learning. First, we wanted to explore whether total segregation of the apical inputs as we had done was necessary, given that real pyramidal neurons only show an attenuation of apical inputs to the soma (Larkum et al., 1999). To examine this, we re-ran our two hidden layer network, but now, we allowed the
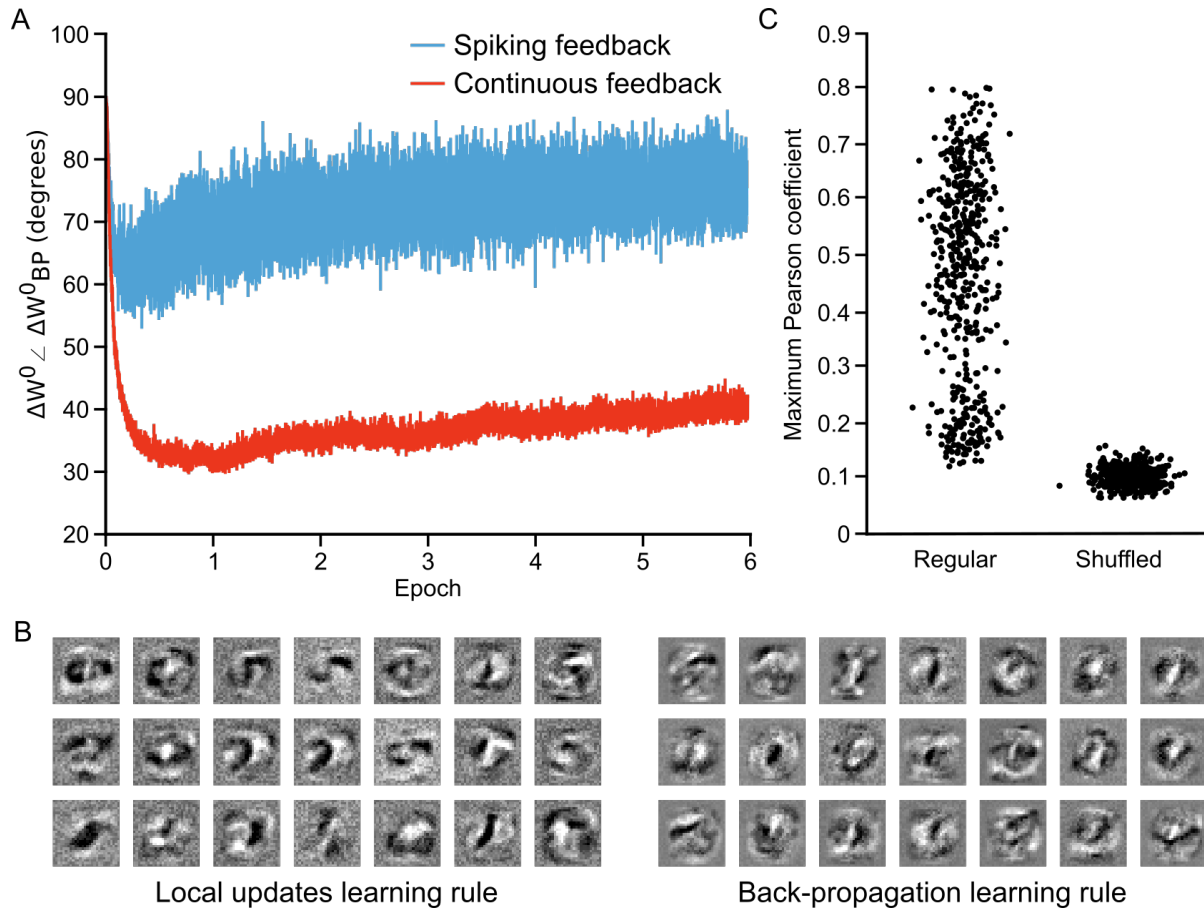
Figure 4. A. Plot of the angle between weight updates prescribed by our local update learning algorithm compared to those prescribed by back-propagation of error, for a one hidden layer network over 6 epochs of training (each point on the x axis corresponds to one image presentation). Data was time-averaged using a sliding window of 100 image presentations. When training the network using the local update learning algorithm, feedback was sent to the hidden layer either using spiking activity from the output layer units (blue) or by directly sending the somatic potentials of output units (red). The angle between the local update ΔW^0 and back-propagation weight updates ΔW_{BP}^0 remains under $90°$ during training, indicating that both algorithms point weight updates in a similar direction. **B.** Examples of hidden layer receptive fields (synaptic weights) obtained by training the network in **A** using our local update learning rule (left) and backpropagation of error (right) for 60 epochs. **C.** Plot of correlation between local update receptive fields and backpropagation receptive fields. For each of the receptive fields produced by local update, we plot the maximum Pearson correlation coefficient between it and all 500 receptive fields learned using backpropagation (Regular). Overall, the maximum correlation coefficients are greater than those obtained after shuffling all of the values of the local update receptive fields (Shuffled).

apical dendritic voltage to influence the somatic voltage, though with twelve times more attenuation than the basal compartments, in-line with experimental data (Larkum et al., 1999, 2009) (**Fig. 5A**). When we compared the learning in this scenario to the scenario with total apical segregation, we observed no difference in the error rates on the test set (**Fig. 5B**, gray and red lines). Importantly, though, we found that if we decreased the apical attenuation to the same level as the basal compartment then the learning was impaired (**Fig. 5B**, blue line). This demonstrates that although total apical attenuation is not necessary, partial segregation of the apical compartment from the soma is.
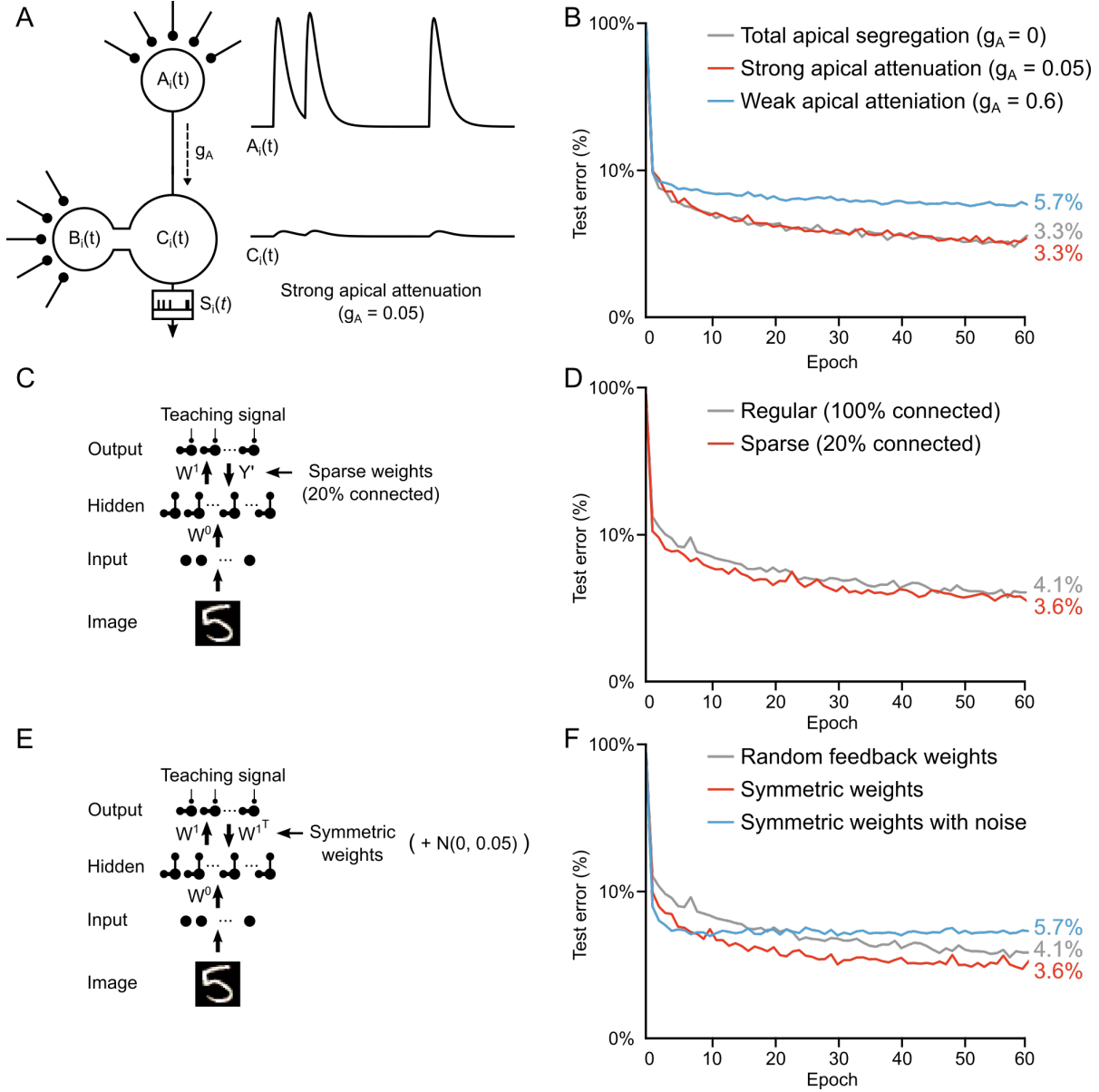
Figure 5. A. *Left*: Diagram of hidden layer neuron with an unsegregated apical dendrite that affects the somatic potential C_i. g_A represents the strength of the coupling between the apical dendrite and soma. *Right*: Example traces of the apical voltage A_i and the somatic voltage C_i in response to spikes arriving at apical synapses. Here $g_A = 0.05$, so the apical activity is strongly attenuated at the soma. **B.** Plot of test error across 60 epochs of training on MNIST of a two hidden layer network, with total apical segregation (gray), strong apical attenuation (red) and weak apical attenuation (blue). Apical input to the soma did not prevent learning if it was strongly attenuated, but weak apical attenuation impaired deep learning. **C.** Diagram of a one hidden layer network trained in **D**, with 80% of feedback weights set to zero. The remaining feedback weights Y' were multiplied by 5 in order to maintain a similar overall magnitude of feedback signals. **D.** Plot of test error across 60 epochs of our standard one hidden layer network (gray) and a network with sparse feedback weights (red). Sparse feedback weights resulted in improved learning performance compared to fully connected feedback paths. **E.** Diagram of a one hidden layer network trained in **F**, with feedback weights that are symmetric to feedforward weights W^1, and symmetric but with added noise. Noise added to feedback weights is drawn from a normal distribution with variance $\sigma = 0.05$. **F.** Plot of test error across 60 epochs of our standard one hidden layer network (gray), a network with symmetric weights (red), and a network with symmetric weights with added noise (blue). Symmetric weights result in improved learning performance compared to random feedback weights, but adding noise to symmetric weights results in impaired learning.

Another constraint that we wished to examine was the structure of the feedback weights. In our initial simulation we used non-sparse, random (i.e. normally distributed) feedback weights. We were interested in whether learning could still work with sparse weights. As well, were wondered whether symmetric weights would *improve* learning, which would be expected given previous findings (Lillicrap et al., 2014; Lee et al., 2015). Hence, we trained our one hidden layer network using both sparse feedback weights (only 20% non-zero) and symmetric weights ($Y = W^{1^T}$) (**Fig. 5C-F**). We found that learning was actually *improved* with sparse weights (**Fig. 5D**, red line), achieving an error rate of 3.6% by the 60th epoch. Similarly, symmetric weights improved learning, leading to a more rapid decrease in the test error and an error rate of 3.6% by the 60th epoch (**Fig. 5F**, red line). However, when we added noise to the symmetric weights this advantage was eliminated and learning was, in fact, slightly impaired (**Fig. 5F**, blue line). Together, these data show that learning can work to some degree with sparse feedback weights, but symmetric feedback weights are, in fact, the better situation for learning.

Discussion

Deep learning has radically altered the field of artificial intelligence, demonstrating that parallel, distributed processing across multiple layers can produce human/animal-level capabilities in image classification, pattern recognition and reinforcement learning (Hinton et al., 2006; LeCun et al., 2015; Mnih et al., 2015; Silver et al., 2016; Krizhevsky et al., 2012; He et al., 2015). Deep learning was ultimately motivated by analogies to the real brain (LeCun et al., 2015; Cox and Dean, 2014), so it is tantalizing that recent studies have shown that deep neural networks develop representations that strongly resemble the representations observed in the mammalian neocortex (Khaligh-Razavi and Kriegeskorte, 2014; Yamins and DiCarlo, 2016; Cadieu et al., 2014; Kubilius et al., 2016), even more so than some models that explicitly attempt to mimic the real brain (Khaligh-Razavi and Kriegeskorte, 2014). Hence, at a phenomenological level, it appears that deep learning, defined as multilayer, coordinated, cost function reduction, may be key to the remarkable computational prowess of the mammalian brain (Marblestone et al., 2016). However, the lack of biological feasibility in deep learning algorithms, most notably back-propagation of error (Rumelhart et al., 1986), has left neuroscientists with a mystery. How can the real brain implement deep learning using local synaptic weight updates, discrete spikes for communication, and continuous voltage fluctuations (Bengio et al., 2015)? Here, we demonstrated that segregating the feedback and feedforward inputs to neurons, much as the real neocortex does (Larkum et al., 1999, 2007, 2009), can enable the construction of local targets to guide synaptic updates, even when using spikes to communicate constantly fluctuating voltages. We showed that we could use segregated dendritic compartments to coordinate learning across layers (**Fig. 1 & 2**). This enabled our network to take advantage of multiple layers to develop abstract, category-based representations of hand-written digits that enabled high levels of classification accuracy on the MNIST dataset (**Fig. 3**). Furthermore, we found that our algorithm actually approximates the weight updates the would be prescribed by back-propagation, and produces similar low-level feature detectors (**Fig. 4**). Therefore, our work demonstrates that deep learning is in fact possible in a biologically feasible framework, provided that feedforward and feedback signals are sufficiently segregated.

One of the principle aspects of biological infeasibility in back-propagation is the "weight-transport" problem: in the back-propagation algorithm, synaptic weight-updates in the hidden layers depend on downstream synaptic weights, which requires non-local transmission of synaptic weight information across layers, or alternatively, feedback synaptic weight matrices that are symmetric with feedforward weight matrices (Grossberg, 1987). Our model builds on recent neural networks research demonstrating that fully symmetric feedback weights are not, in fact, required for deep learning to proceed (Lillicrap et al., 2014; Liao et al., 2015; Lee et al., 2015). By demonstrating that the weight transport problem is solvable, these studies have provided a major breakthrough in our understanding of how deep learning could work in the real brain. However, this previous research did not explore whether these same principles could be implemented in leaky neurons with continuously fluctuating voltages that communicate via spikes. In our study, we adopted the strategy taken by Lee et al.'s (2015) difference target propagation algorithm,

wherein the feedback from higher layers is used to construct local activity targets at the hidden layers. One of the reasons that we adopted this strategy is that it is appealing to think that feedback from upper layers may not simply be providing a signal for plasticity, but also providing information that could be used to push the hidden layer neurons towards a "better" activity pattern in *real-time*. This sort of local teaching signal could be used by the brain to improve sensory processing in different contexts and engage in inference (Bengio et al., 2015). However, we did not actually implement a system that pushed the hidden layer towards its target in real-time—we simply used the local target to drive learning. In this respect, a potential future advance on our model would be to implement a system wherein the feedback actively "nudges" the hidden layer towards its target during learning. This proposal is reminiscent of the approach taken by Urbanczik and Senn (2014) in their paper on dendritic prediction, wherein "nudging conductances" at the somatic compartments ensured that the voltages in their model neurons were pushed towards the target voltages specified by a teaching signal. Perhaps it would be possible to coordinate learning across layers as we have while simultaneously pushing the somatic compartments in the hidden layer towards their local targets to improve performance *on-line*.

It is important to note that there are many aspects of our model that are not biologically realistic. These include (but are not limited to) synaptic inputs that are not conductance based, synaptic weights that can switch from positive to negative (and vice-versa), and "burst" activity that is considered instantaneous and is not an accurate reflection of burst or calcium spike dynamics in real pyramidal neurons (Larkum et al., 1999, 2009). However, the intention with this study was not to achieve total biological realism, but rather to demonstrate biologically *feasible* mechanisms that could permit deep learning. In particular, what we wish to highlight about our work is that it shows how feedback signals that are electrotonically segregated from the soma, except at select moments in time, can permit deep learning. Although we found that deep learning can still work with realistic levels of passive conductance from the apical compartment to the somatic compartment (Larkum et al., 1999), we also found that learning was impaired if the basal and apical compartments had equal conductance to the soma (**Fig. 5 A,B**). This is interesting, because it suggests that the unique physiology of neocortical pyramidal neurons may in fact reflect nature's solution to deep learning. Perhaps the relegation of feedback from higher-order brain regions to the electrotonically distant apical dendrites (Budd, 1998; Spratling, 2002), and the presence of active calcium spike mechanisms in the apical shaft (Larkum et al., 2009), are mechanisms for coordinating local synaptic weight updates. If this is correct, then the SST positive inhibitory interneurons that target apical dendrites and limit active communication to the soma (Murayama et al., 2009) may be used by the neocortex to switch between "feedforward" transmission and "feedback" phases for learning. Although this is highly speculative, it is worth noting that current evidence supports the idea that neuromodulatory inputs carrying temporally precise salience information (Hangya et al., 2015) can shut off SST interneurons via disinhibition (Pi et al., 2013; Karnani et al., 2016; Pfeffer et al., 2013), and presumably, promote apical communication to the soma. Future research should examine whether these inhibitory and neuromodulatory mechanisms do, in fact, open the window for plasticity in the basal dendrites of pyramidal neurons, as our model predicts.

An additional issue that should be recognized is that the error rates which our network achieved were by no means as low as can be achieved with artificial neural networks, nor at human levels of performance (LeCun et al., 1998; Li et al., 2016). As well, our algorithm was not able to take advantage of very deep structures (beyond two hidden layers, the error rate did not improve). In contrast, increasing the depth of networks trained with back-propagation can lead to major performance improvements (Li et al., 2016). But, these observations do not mean that our network was not engaged in deep learning. First, we stress that our network was not designed to match the state-of-the-art in machine learning, nor human capabilities. To test our basic hypothesis we kept the network small, and we did not implement any add-ons to the learning to improve the error rates, such as mini-batch training, weight decay, drop-out, momentum or RMSProp (Tieleman and Hinton, 2012; Srivastava et al., 2014). Indeed, it would be quite surprising if a relatively vanilla, small network like ours could come close to matching current performance benchmarks in machine learning. Second, although our network was able to take advantage of multiple layers to improve performance, there may be a variety of reasons that ever increasing depth didn't improve performance significantly. For example, our use of direct connections from the output

layer to the hidden layers may have impaired the network's ability to coordinate synaptic updates between *hidden* layers. As well, given our finding that the use of spikes produced weight updates that were less well-aligned to back-propagation (**Fig. 4A**) it is possible that deeper architectures require mechanisms to overcome the inherent noisiness of spikes.

One aspect of our model that we did not develop was the potential for learning at the feedback synapses. Although we used random synaptic weights for feedback, we also demonstrated that symmetric weights can actually improve learning (**Fig. 5 C & D**). This suggests that it would be beneficial to develop a unique synaptic weight update rule for the feedback synapses that pushed them towards symmetry with the feedforward synapses. Indeed, Lee et al. (2015) implemented an "inverse loss function" for their feedback synapses which promoted the development of symmetric weights and auto-encoder functions in the network. In light of this, it is interesting to note that there is evidence for unique, "reverse" spike-timing dependent synaptic plasticity rules in the distal apical dendrites of pyramidal neurons (Sjöström and Häusser, 2006; Letzkus et al., 2006), which have been shown to produce symmetric feedback weights and auto-encoder functions in spiking networks (Burbank and Kreiman, 2012; Burbank, 2015). Thus, it is possible that the neocortex actually learns symmetric feedback weights over the course of development. Our work would suggests that experimental evidence for increasing weight symmetry over development could be considered evidence for deep learning in the neocortex.

Materials and Methods

For notational simplicity, we describe our model in the case of a network with only one hidden layer. We describe how this is extended to a network with multiple layers at the end of this section. As well, at the end of this section we provide a table listing the parameter values we used for all of the simulations presented in this paper.

Neuronal dynamics

Neurons in the hidden layer are modeled using three functional compartments – basal dendrites with voltages $\boldsymbol{B}$, apical dendrites with voltages $\boldsymbol{A}$, and somatic compartments with voltages $\boldsymbol{C}$. Feedforward and feedback spiking inputs arrive at basal and apical synapses, respectively. At each synapse, presynaptic spikes are translated into post-synaptic potentials given by:

$$\boldsymbol{PSP}(t) = \sum_{s \in X} \boldsymbol{\kappa}(t - s) \tag{5}$$

...where X is the set of pre-synaptic spike times. $\boldsymbol{\kappa}(t) = (e^{-t/\tau_L} - e^{-t/\tau_s})\Theta(t)/(\tau_L - \tau_s)$ is the response kernel, where τ_s and τ_L are short and long time constants, and Θ is the Heaviside step function. The basal and apical dendritic potentials are given by weighted sums of the post-synaptic potentials at their synapses:

$$\boldsymbol{B}(t) = W^0 \boldsymbol{PSP^B}(t) + \boldsymbol{b}^0$$
$$\boldsymbol{A}(t) = Y \boldsymbol{PSP^A}(t) \tag{6}$$

...where $\boldsymbol{b}^0$ is a bias term, and W^0 and Y are the feedforward and feedback synaptic weight matrices, respectively, for neurons in the hidden layer. The somatic voltages $\boldsymbol{C}$ evolve with leak as:

$$\dot{\boldsymbol{C}} = -g_L \boldsymbol{C} + g_B(\boldsymbol{B} - \boldsymbol{C}) \tag{7}$$

...where g_L is the leak conductance and g_B is the conductance from the basal dendrite to the soma. Note that for simplicity's sake we are assuming a resting potential of 0 V and a membrane capacitance

of 1 F, but these values are not important for the results. In **Figure 5B** we introduce a conductance from the apical dendrite to the soma, g_A, so that:

$$\dot{\boldsymbol{C}} = -g_L\boldsymbol{C} + g_B(\boldsymbol{B} - \boldsymbol{C}) + g_A(\boldsymbol{A} - \boldsymbol{C}) \tag{8}$$

The instantaneous firing rate of neurons in the hidden layer, $\boldsymbol{\lambda}_C$, is given by a nonlinearity $\phi(\cdot)$ applied to $\boldsymbol{C}$, which we chose to be a simple sigmoid function:

$$\boldsymbol{\lambda}_C(t) = \phi(\boldsymbol{C}(t)) = \frac{\phi_{max}}{1 + e^{-\boldsymbol{C}(t)}} \tag{9}$$

...where ϕ_{max} is the maximum possible spike rate, which we set to 200 Hz. Spikes are then generated using Poisson processes with these firing rates. We note that although the maximum rate was 200 Hz, the neurons rarely achieved anything close to this rate, and the average rate of fire in the neurons during our simulations was 24 Hz, in-line with neocortical firing rates (Steriade et al., 2001).

Units in the output layer are modeled using only two compartments, dendrites with voltages $\boldsymbol{V}$ and soma with voltages $\boldsymbol{U}$. $\boldsymbol{V}$ is given by:

$$\boldsymbol{V}(t) = W^1\boldsymbol{PSP^V}(t) + \boldsymbol{b}^1 \tag{10}$$

...and $\boldsymbol{U}$ evolves as:

$$\dot{\boldsymbol{U}} = -g_L\boldsymbol{U} + g_D(\boldsymbol{V} - \boldsymbol{U}) + \boldsymbol{I} \tag{11}$$

...where $\boldsymbol{I}$ is a somatic current that can drive output neurons toward a desired somatic voltage, and is given by:

$$\boldsymbol{I}(t) = \boldsymbol{g_E}(t)(E_E - \boldsymbol{U}) + \boldsymbol{g_I}(t)(E_I - \boldsymbol{U}) \tag{12}$$

...where E_E and E_I are the excitatory & inhibitory reversal potentials, and $\boldsymbol{g_E}(t)$ and $\boldsymbol{g_I}(t)$ are time-varying excitatory & inhibitory nudging conductances. In our simulations, we set $E_E = 8$ and $E_I = -8$. During the target phase only, we set $\boldsymbol{g_I}_i = 1$ and $\boldsymbol{g_E}_i = 0$ for all units i whose output should be minimal, and $\boldsymbol{g_E}_i = 1$ and $\boldsymbol{g_I}_i = 0$ for the unit whose output should be maximal. In this way, all units other than the "target" unit are silenced, while the "target" unit receives a strong excitatory drive. In the forward phase, $\boldsymbol{I}$ is set to 0. The Poisson spike rates $\boldsymbol{\lambda}_U$ are calculated as in Equation (9).

Weight updates

All feedforward synaptic weights are updated at the end of each target phase. Output layer units update their synaptic weights W^1 in order to minimize the loss function L^1 given in Equation (2). All average voltages are calculated after a delay Δt_s from the start of a phase, which allows for the network to reach a steady state before averaging begins. In practice this means that

$$\overline{\boldsymbol{U}}^f \approx k_D\overline{\boldsymbol{V}}^f = k_DW^1\overline{\boldsymbol{PSP^V}}^f \tag{13}$$

...where $k_D = g_D/(g_L + g_D)$. Thus,

$$\begin{aligned}
\frac{\partial L^1}{\partial W^1} &\approx -k_D(\hat{\boldsymbol{\lambda}}_U - \phi(\overline{\boldsymbol{U}}^f))\phi'(\overline{\boldsymbol{U}}^f) \circ \overline{\boldsymbol{PSP^V}}^f \\
\frac{\partial L^1}{\partial \boldsymbol{b}^1} &\approx -k_D(\hat{\boldsymbol{\lambda}}_U - \phi(\overline{\boldsymbol{U}}^f))\phi'(\overline{\boldsymbol{U}}^f)
\end{aligned} \tag{14}$$

The dendrites in the output layer use this approximation of the gradient in order to update their weights using gradient descent:

$$W^1 \rightarrow W^1 - \eta^1 P^1 \frac{\partial L^1}{\partial W^1}$$
$$\boldsymbol{b}^1 \rightarrow \boldsymbol{b}^1 - \eta^1 P^1 \frac{\partial L^1}{\partial \boldsymbol{b}^1} \tag{15}$$

...where η^1 is a learning rate constant, and P^1 is a scaling factor used to normalize the scale of the firing rate function ϕ.

In the hidden layer, basal dendrites update their synaptic weights W^0 by minimizing the loss function L^0 given in Equation (2). We define the target firing rates $\hat{\boldsymbol{\lambda}}_C = \phi(\overline{\boldsymbol{C}}^f) + \boldsymbol{\alpha}^t - \boldsymbol{\alpha}^f$ similar to the targets used in difference target propagation (Lee et al., 2015). Burst activity, $\boldsymbol{\alpha}$, was given by a non-linearity applied to the average apical dendritic potentials during the transmission modes, where α was chosen to be:

$$\alpha(\overline{\boldsymbol{A}}) = \frac{1}{(1 + e^{\overline{\boldsymbol{A}}})} \tag{16}$$

Although in theory any differentiable function is suitable. Then, using the fact that $\hat{\boldsymbol{\lambda}}_C - \phi(\overline{\boldsymbol{C}}^f) = \boldsymbol{\alpha}^t - \boldsymbol{\alpha}^f$,

$$\frac{\partial L^0}{\partial W^0} \approx -k_B(\boldsymbol{\alpha}^t - \boldsymbol{\alpha}^f)\phi'(\overline{\boldsymbol{C}}^f) \circ \overline{\boldsymbol{PSP^B}}^f$$
$$\frac{\partial L^0}{\partial \boldsymbol{b}^0} \approx -k_B(\boldsymbol{\alpha}^t - \boldsymbol{\alpha}^f)\phi'(\overline{\boldsymbol{C}}^f) \tag{17}$$

...where $k_B = g_B/(g_L + g_B + g_A)$. Basal weights are updated in order to descend this gradient:

$$W^0 \rightarrow W^0 - \eta^0 P^0 \frac{\partial L^0}{\partial W^0}$$
$$\boldsymbol{b}^0 \rightarrow \boldsymbol{b}^0 - \eta^0 P^0 \frac{\partial L^0}{\partial \boldsymbol{b}^0} \tag{18}$$

Importantly, this update rule is fully local for the hidden layer neurons. It consists essentially of three terms, (1) the difference in the burst activity during target and forward phases, (2) the derivative of the spike rate function, and (3) the postsynaptic potentials. All three of these terms are values that a real neuron could theoretically calculate using some combination of molecular synaptic tags, calcium currents, and back-propagating action potentials. Hence, the learning algorithm is biologically feasible.

Hidden layer targets

Theorem 1 shows that if we use a target $\hat{\boldsymbol{\lambda}}_C^* = \overline{\boldsymbol{\lambda}_C}^f + \alpha(Y\overline{\boldsymbol{\lambda}_U}^t) - \alpha(Y\phi(kW^1\overline{\boldsymbol{\lambda}_C}^f))$ for the hidden layer, there is a guarantee that the hidden layer approaching this target will also push the upper layer closer to its target $\hat{\boldsymbol{\lambda}}_U$, if certain other conditions are met. Our choice of $\hat{\boldsymbol{\lambda}}_C$ approximates this target rate vector using variables that are accessible to the hidden layer units.

Because neuronal units calculate averages after the network has reached a steady state, $\phi(\overline{\boldsymbol{U}}^f) \approx \overline{\boldsymbol{\lambda}_U}^f$. Using **Lemma 1** and Equation (13), $E[\overline{\boldsymbol{U}}^f] \approx kW^1\overline{\boldsymbol{\lambda}_C}^f$. If we assume that $\overline{\boldsymbol{U}}^f \approx E[\overline{\boldsymbol{U}}^f]$, which is true on average, then:

$$\boldsymbol{\alpha}^f = \alpha(\overline{\boldsymbol{A}}^f) \approx \alpha(Y\overline{\boldsymbol{\lambda}_U}^f) \approx \alpha(Y\phi(\overline{U}^f)) \approx \alpha(Y\phi(kW^1\overline{\boldsymbol{\lambda}_C}^f)) \tag{19}$$

...and:

$$\boldsymbol{\alpha}^t = \alpha(\overline{\boldsymbol{A}}^t) \approx \alpha(Y\overline{\boldsymbol{\lambda}_U}^t) \tag{20}$$

Therefore, $\hat{\boldsymbol{\lambda}}_C \approx \hat{\boldsymbol{\lambda}}_C^*$.

Thus, our hidden layer targets ensure that our model employs a learning rule similar to difference target propagation that approximates the conditions to guarantee error convergence.

Multiple hidden layers

In order to extend our algorithm to deeper networks with multiple hidden layers, our model incorporates direct synaptic connections from the output layer to each hidden layer. Thus, each hidden layer receives the same feedback from the output layer through its own separate set of fixed, random backwards weights. For example, in a network with two hidden layers, both layers receive the same feedback from the output layer at their apical dendrites through backward weights Y^0 and Y^1. The local targets at each layer are then given by:

$$\hat{\boldsymbol{\lambda}}_U = \overline{\boldsymbol{\lambda}}_U^t \tag{21}$$

$$\hat{\boldsymbol{\lambda}}_C^1 = \phi(\overline{\boldsymbol{C}^1}^f) + \boldsymbol{\alpha}^{1t} - \boldsymbol{\alpha}^{1f} \tag{22}$$

$$\hat{\boldsymbol{\lambda}}_C^0 = \phi(\overline{\boldsymbol{C}^0}^f) + \boldsymbol{\alpha}^{0t} - \boldsymbol{\alpha}^{0f} \tag{23}$$

where the superscripts ^{0} and ^{1} denote the first and second hidden layers, respectively.

The local loss functions at each layer are:

$$\begin{aligned}
L^2 &= ||\hat{\boldsymbol{\lambda}}_U - \phi(\overline{\boldsymbol{U}}^f)||_2^2 \\
L^1 &= ||\hat{\boldsymbol{\lambda}}_C^1 - \phi(\overline{\boldsymbol{C}^1}^f)||_2^2 \\
L^0 &= ||\hat{\boldsymbol{\lambda}}_C^0 - \phi(\overline{\boldsymbol{C}^0}^f)||_2^2
\end{aligned} \tag{24}$$

where L^2 is the loss at the output layer.

The learning rules used by the hidden layers in this scenario are the same as in the case with one hidden layer.

Learning rate optimization

For each of the three network sizes that we present in this paper, a grid search was performed in order to find optimal learning rates. We set the learning rate for each layer by stepping through the range $[0.1, 0.3]$ with a step size of 0.02. For each combination of learning rates, a neural network was trained for one epoch on the 60, 000 training examples, after which the network was tested on 10,000 test images. The learning rates that gave the best performance on the test set after an epoch of training were used as a basis for a second grid search around these learning rates that used a smaller step size of 0.01. From this, the learning rates that gave the best test performance after 20 epochs were chosen as our optimal learning rates for that network size.

In all of our simulations, we used a learning rate of 0.19 for a network with no hidden layers, learning rates of 0.21 (output and hidden) for a network with one hidden layer, and learning rates of 0.23 (output and second hidden layer) and 0.12 (first hidden layer) for a network with two hidden layers. All networks with one hidden layer had 500 hidden layer units, and all networks with two hidden layers had 500 units in the first hidden layer and 100 units in the second hidden layer.

Training paradigm

For all simulations described in this paper, the neural networks were trained on classifying handwritten digits using the MNIST database of 28 pixel x 28 pixel images. Initial feedforward and feedback weights were chosen randomly from a uniform distribution.

Prior to training, we tested a network's initial performance on a set of 10,000 test examples. This set of images was shuffled at the beginning of testing, and each example was shown to the network in sequence. Each input image was encoded into Poisson spiking activity of 784 input units representing each pixel of the image. The firing rate of an input unit was proportional to the brightness of the pixel that it represents. The spiking activity of each of the 784 input units was received by the units in the first hidden layer. For each test image, the network underwent only a forward phase. At the end of this phase, the network's classification of the input image was given by the unit in the output layer with the greatest somatic potential (and therefore the greatest spike rate). The network's classification was compared to the target classification. After classifying all 10,000 testing examples, the network's classification error was given by the percentage of examples that it did not classify correctly.

Following the initial test, training of the neural network was done in an on-line fashion. All 60,000 training images were randomly shuffled at the start of each training epoch. The network was then shown each training image in sequence, undergoing a forward phase ending with a burst, and a target phase ending with a burst. All feedforward weights were then updated at the end of the target phase. At the end of the epoch (after all 60,000 images were shown to the network), the network was again tested on the 10,000 test examples. The network was trained for up to 60 epochs.

Simulation details

For each training example, a minimum length of 50ms was used for each of the forward & training phases. The lengths of the forward & target transmit phases were determined by adding their minimum length to an extra length term, which was chosen randomly from a Wald distribution with a mean of 2ms and scale factor of 1. During testing, a fixed length of 500ms was used for the forward transmit phase. Average forward & target phase voltages were calculated after a settle duration of 30ms from the start of the phase.

The time step used for simulations was 1ms. At each time step, the network's state was updated bottom-to-top beginning with the first hidden layer and ending with the output layer. For each layer, dendritic potentials were updated, followed by somatic potentials, and finally their spiking activity.

Table 1 lists the simulation parameters and the values that were used in the figures presented.

All code was written using Python version 2.7 and is available at https://github.com/jordan-g/Segregated-Dendrite-Deep-Learning.

Parameter	Units	Value	Description
dt	ms	1	Time step resolution
ϕ_{max}	Hz	200	Maximum spike rate
τ_s	ms	3	Short synaptic time constant
τ_L	ms	10	Long synaptic time constant
g_B	S	0.6	Hidden layer conductance from basal dendrite to the soma
g_A	S	0, 0.05, 0.6	Hidden layer conductance from apical dendrite to the soma
g_D	S	0.6	Output layer conductance from dendrite to the soma
g_L	S	0.1	Leak conductance
P_0	–	$20/\phi_{max}$	Hidden layer error signal scaling factor
P_1	–	$20/\phi_{max}^2$	Output layer error signal scaling factor

Table 1. List of parameter values used in our simulations.

Proofs

Lemma 1. *Let X be a set of presynaptic spike times during the time interval $\Delta t = t_1 - t_0$, distributed according to an inhomogeneous Poisson process. Let $N = |X|$ denote the number of presynaptic spikes during this time window, and let $s_i \in X$ denote the ith presynaptic spike, where $0 < i \leq N$. Finally, let $\lambda(t)$ denote the time-varying presynaptic firing rate (i.e. the time-varying mean of the Poisson process), and $PSP(t)$ be the postsynaptic potential at time t given by Equation (5). Then, during the time window Δt, as long as $\Delta t \gg 2\tau_L^2 \tau_s^2 \overline{\lambda^2}/(\tau_L - \tau_s)^2(\tau_L + \tau_s)$,*

$$E[\overline{PSP(t)}] \approx \overline{\lambda}$$

Proof. The average of $PSP(t)$ over the time window Δt is

$$\overline{PSP} = \frac{1}{\Delta t} \int_{t_0}^{t_1} PSP(t)dt$$

$$= \frac{1}{\Delta t} \sum_{s \in X} \int_{t_0}^{t_1} \frac{e^{-(t-s)/\tau_L} - e^{-(t-s)/\tau_s}}{\tau_L - \tau_s} \Theta(t-s)dt$$

Since $\Theta(t-s) = 0$ for all $t < s$,

$$\overline{PSP} = \frac{1}{\Delta t} \sum_{s \in X} \int_{s}^{t_1} \frac{e^{-(t-s)/\tau_L} - e^{-(t-s)/\tau_s}}{\tau_L - \tau_s} dt$$

$$= \frac{1}{\Delta t} \left(N - \sum_{s \in X} \frac{\tau_L e^{-(t_1-s)/\tau_L} - \tau_s e^{-(t_1-s)/\tau_s}}{\tau_L - \tau_s} \right)$$

The expected value of $\overline{PSP}$ with respect to X is given by

$$E_X[\overline{PSP}] = E_X \left[\frac{1}{\Delta t} \left(N - \sum_{s \in X} \frac{\tau_L e^{-(t_1-s)/\tau_L} - \tau_s e^{-(t_1-s)/\tau_s}}{\tau_L - \tau_s} \right) \right]$$

$$= \frac{E_X[N]}{\Delta t} - \frac{1}{\Delta t} E_X \left[\sum_{i=0}^{N} \left(\frac{\tau_L e^{-(t_1-s_i)/\tau_L} - \tau_s e^{-(t_1-s_i)/\tau_s}}{\tau_L - \tau_s} \right) \right]$$

Since the presynaptic spikes are a inhomogeneous Poisson process with a rate λ, $E_X[N] = \int_{t_0}^{t_1} \lambda dt$. Thus,

$$E_X[\overline{PSP}] = \frac{1}{\Delta t} \int_{t_0}^{t_1} \lambda dt - \frac{1}{\Delta t} E_X \left[\sum_{i=0}^{N} g(s_i) \right]$$

$$= \overline{\lambda} - \frac{1}{\Delta t} E_X \left[\sum_{i=0}^{N} g(s_i) \right]$$

where we let $g(s_i) \equiv (\tau_L e^{-(t_1-s_i)/\tau_L} - \tau_s e^{-(t_1-s_i)/\tau_s})/(\tau_L - \tau_s)$. Then, the law of total expectation gives

$$E_X \left[\sum_{i=0}^{N} g(s_i) \right] = E_N \left[E_X \left[\sum_{i=0}^{N} g(s_i) \bigg| N \right] \right]$$

$$= \sum_{n=0}^{\infty} \left(E_X \left[\sum_{i=0}^{N} g(s_i) \bigg| N = n \right] \cdot P(N = n) \right)$$

Letting $f_{s_i}(s)$ denote $P(s_i = s)$, we have that

$$E_X\left[\sum_{i=0}^{N} g(s_i)\middle| N = n\right] = \sum_{i=0}^{n} E_X[g(s_i)]$$

$$= \sum_{i=0}^{n} \int_{t_0}^{t_1} g(s)f_{s_i}(s)ds$$

Since Poisson spike times are independent, for an inhomogeneous Poisson process:

$$f_{s_i}(s) = \frac{\lambda(s)}{\int_{t_0}^{t_1} \lambda(t)dt}$$

$$= \frac{\lambda(s)}{\bar{\lambda}\Delta t}$$

for all $s \in [t_0, t_1]$. Since Poisson spike times are independent, this is true for all i. Thus,

$$E_X\left[\sum_{i=0}^{N} g(s_i)\middle| N = n\right] = \frac{1}{\bar{\lambda}\Delta t} \sum_{i=0}^{n} \int_{t_0}^{t_1} g(s)\lambda(s)ds$$

$$= \frac{n}{\bar{\lambda}\Delta t} \int_{t_0}^{t_1} g(s)\lambda(s)ds$$

Then,

$$E_X\left[\sum_{i=0}^{N} g(s_i)\right] = \sum_{n=0}^{\infty} \left(\frac{n}{\bar{\lambda}\Delta t}\left(\int_{t_0}^{t_1} g(s)\lambda(s)ds\right) \cdot P(N = n)\right)$$

$$= \frac{1}{\bar{\lambda}\Delta t}\left(\int_{t_0}^{t_1} g(s)\lambda(s)ds\right)\left(\sum_{n=0}^{\infty} n \cdot P(N = n)\right)$$

Now, for an inhomogeneous Poisson process with time-varying rate $\lambda(t)$,

$$P(N = n) = \frac{[\int_{t_0}^{t_1} \lambda(t)dt]^n e^{-\int_{t_0}^{t_1} \lambda(t)dt}}{n!}$$

$$= \frac{[\bar{\lambda}\Delta t]^n e^{-(\bar{\lambda}\Delta t)}}{n!}$$

Thus,

$$E_X\left[\sum_{i=0}^{N} g(s_i)\right] = \frac{e^{-(\bar{\lambda}\Delta t)}}{\bar{\lambda}\Delta t}\left(\int_{t_0}^{t_1} g(s)\lambda(s)ds\right)\left(\sum_{n=0}^{\infty} n\frac{[\bar{\lambda}\Delta t]^n}{n!}\right)$$

$$= \frac{e^{-(\bar{\lambda}\Delta t)}}{\bar{\lambda}\Delta t}\left(\int_{t_0}^{t_1} g(s)\lambda(s)ds\right)(\bar{\lambda}\Delta t)e^{\bar{\lambda}\Delta t}$$

$$= \int_{t_0}^{t_1} g(s)\lambda(s)ds$$

Then,

$$E_X[\overline{PSP}] = \overline{\lambda} - \frac{1}{\Delta t}\left(\int_{t_0}^{t_1} g(s)\lambda(s)ds\right)$$

The second term of this equation is always greater than or equal to 0, since $g(s) \geq 0$ and $\lambda(s) \geq 0$ for all s. Thus, $E_X[\overline{PSP}] \leq \overline{\lambda}$. As well, the Cauchy-Schwarz inequality states that

$$\int_{t_0}^{t_1} g(s)\lambda(s)ds \leq \sqrt{\int_{t_0}^{t_1} g(s)^2 ds}\sqrt{\int_{t_0}^{t_1} \lambda(s)^2 ds}$$

$$= \sqrt{\int_{t_0}^{t_1} g(s)^2 ds}\sqrt{\overline{\lambda^2}\Delta t}$$

where

$$\int_{t_0}^{t_1} g(s)^2 ds = \int_{t_0}^{t_1}\left(\frac{\tau_L e^{-(t_1-s)/\tau_L} - \tau_s e^{-(t_1-s)/\tau_s}}{\tau_L - \tau_s}\right)^2 ds$$

$$\leq \frac{1}{2(\tau_L - \tau_s)^2}\left(4\frac{\tau_L^2\tau_s^2}{\tau_L + \tau_s}\right)$$

$$= \frac{2\tau_L^2\tau_s^2}{(\tau_L - \tau_s)^2(\tau_L + \tau_s)}$$

Thus,

$$\int_{t_0}^{t_1} g(s)\lambda(s)ds \leq \sqrt{\frac{2\tau_L^2\tau_s^2}{(\tau_L - \tau_s)^2(\tau_L + \tau_s)}}\sqrt{\overline{\lambda^2}\Delta t}$$

$$= \sqrt{\Delta t}\sqrt{\frac{2\tau_L^2\tau_s^2\overline{\lambda^2}}{(\tau_L - \tau_s)^2(\tau_L + \tau_s)}}$$

Therefore,

$$E_X[\overline{PSP}] \geq \overline{\lambda} - \frac{1}{\Delta t}\sqrt{\Delta t}\sqrt{\frac{2\tau_L^2\tau_s^2\overline{\lambda^2}}{(\tau_L - \tau_s)^2(\tau_L + \tau_s)}}$$

$$= \overline{\lambda} - \sqrt{\frac{2\tau_L^2\tau_s^2\overline{\lambda^2}}{\Delta t(\tau_L - \tau_s)^2(\tau_L + \tau_s)}}$$

Then,

$$\overline{\lambda} - \sqrt{\frac{2\tau_L^2\tau_s^2\overline{\lambda^2}}{\Delta t(\tau_L - \tau_s)^2(\tau_L + \tau_s)}} \leq E_X[\overline{PSP}] \leq \overline{\lambda}$$

Thus, as long as $\Delta t \gg 2\tau_L^2\tau_s^2\overline{\lambda^2}/(\tau_L - \tau_s)^2(\tau_L + \tau_s)$, $E_X[\overline{PSP}] \approx \overline{\lambda}$. $\qquad\square$

What this lemma says, effectively, is that the expected value of PSP is going to be roughly the average presynaptic rate of fire as long as the time over which the average is taken is sufficiently long in comparison to the postsynaptic time constants and the average rate-of-fire is sufficiently small. In our simulations, Δt is always greater than or equal to 50 ms, the average rate-of-fire is approximately 20 Hz, and our time constants τ_L and τ_s are 10 ms and 3 ms, respectively. Hence, in general:

$$2\tau_L^2\tau_s^2\overline{\lambda^2}/(\tau_L - \tau_s)^2(\tau_L + \tau_s) = 2(10)^2(3)^2(0.02)^2/(10-3)^2(10+3)$$
$$\approx 0.001$$
$$\ll 50$$

Thus, in the subsequent proof, we assume $E_X[\overline{PSP}] = \overline{\lambda}$.

Theorem 1. *Consider a neural network with a hidden layer and an output layer. Let $\hat{\boldsymbol{\lambda}}_C^* = \overline{\boldsymbol{\lambda}_C}^f + \alpha(Y\overline{\boldsymbol{\lambda}_U}^t) - \alpha(Y\phi(E[\overline{\boldsymbol{U}}^f]))$ be the target firing rates for neurons in the hidden layer, where $\alpha(\cdot)$ and $\phi(\cdot)$ are differentiable functions. Assume that $\overline{\boldsymbol{U}}^f \approx k_D\overline{\boldsymbol{V}}^f$. Let $\hat{\boldsymbol{\lambda}}_U = \overline{\boldsymbol{\lambda}_U}^t$ be the target firing rates for the output layer. Also, for notational simplicity, let $\beta(\boldsymbol{x}) \equiv \phi(W^1k\boldsymbol{x})$ and $\gamma(\boldsymbol{x}) \equiv \alpha(Y\boldsymbol{x})$. Theorem 1 states that if $\hat{\boldsymbol{\lambda}}_U - \phi(E[\overline{\boldsymbol{U}}^f])$ is sufficiently small, and the Jacobian matrices J_β and J_γ satisfy that the largest eigenvalue of $(I - J_\beta J_\gamma)^T(I - J_\beta J_\gamma)$ is less than 1, then*

$$||\hat{\boldsymbol{\lambda}}_U - \phi(W^1k_D\hat{\boldsymbol{\lambda}}_C^*)||_2^2 < ||\hat{\boldsymbol{\lambda}}_U - \phi(E[\overline{\boldsymbol{U}}^f])||_2^2$$

We note that the proof for this theorem is essentially a modification of the proof provided in Lee et al. (2015) that incorporates our lemma regarding the expected value of PSP.

Proof.

$$\begin{aligned}
\hat{\boldsymbol{\lambda}}_U - \phi(W^1k_D\hat{\boldsymbol{\lambda}}_C^*) &= \hat{\boldsymbol{\lambda}}_U - \beta(\hat{\boldsymbol{\lambda}}_C^*) \\
&= \hat{\boldsymbol{\lambda}}_U - \beta(\overline{\boldsymbol{\lambda}_C}^f + \gamma(\overline{\boldsymbol{\lambda}_U}^t) - \gamma(\phi(E[\overline{\boldsymbol{U}}^f])))
\end{aligned}$$

Lemma 1 shows that $\phi(E[\overline{\boldsymbol{U}}^f]) = \phi(E[W^1k_D\overline{\boldsymbol{PSP^V}}^f]) \approx \phi(W^1k_D\overline{\boldsymbol{\lambda}_C}^f)$ given a sufficiently large averaging time window. Let $\boldsymbol{e} = \overline{\boldsymbol{\lambda}_U}^t - \phi(W^1k_D\overline{\boldsymbol{\lambda}_C}^f) = \overline{\boldsymbol{\lambda}_U}^t - \beta(\overline{\boldsymbol{\lambda}_C}^f)$. Applying Taylor's theorem,

$$\hat{\boldsymbol{\lambda}}_U - \beta(\hat{\boldsymbol{\lambda}}_C^*) = \hat{\boldsymbol{\lambda}}_U - \beta(\overline{\boldsymbol{\lambda}_C}^f + J_\gamma\boldsymbol{e} + \boldsymbol{o}(||\boldsymbol{e}||_2))$$

where $\boldsymbol{o}(||\boldsymbol{e}||_2)$ is the remainder term that satisfies $\lim_{\boldsymbol{e}\to 0}\boldsymbol{o}(||\boldsymbol{e}||_2)/||\boldsymbol{e}||_2 = 0$. Applying Taylor's theorem again,

$$\begin{aligned}
\hat{\boldsymbol{\lambda}}_U - \beta(\hat{\boldsymbol{\lambda}}_C^*) &= \hat{\boldsymbol{\lambda}}_U - \beta(\overline{\boldsymbol{\lambda}_C}^f) - J_\beta(J_\gamma\boldsymbol{e} + \boldsymbol{o}(||\boldsymbol{e}||_2)) \\
&\quad - \boldsymbol{o}(||(J_\gamma\boldsymbol{e} + \boldsymbol{o}(||\boldsymbol{e}||_2)||_2) \\
&= \hat{\boldsymbol{\lambda}}_U - \beta(\overline{\boldsymbol{\lambda}_C}^f) + J_\beta J_\gamma\boldsymbol{e} - \boldsymbol{o}(||\boldsymbol{e}||_2) \\
&= (I - J_\beta J_\gamma)\boldsymbol{e} - \boldsymbol{o}(||\boldsymbol{e}||_2)
\end{aligned}$$

Then,

$$\begin{aligned}
||\hat{\boldsymbol{\lambda}}_U - \beta(\hat{\boldsymbol{\lambda}}_C^*)||_2^2 &= ((I - J_\beta J_\gamma)\boldsymbol{e} - \boldsymbol{o}(||\boldsymbol{e}||_2))^T((I - J_\beta J_\gamma)\boldsymbol{e} - \boldsymbol{o}(||\boldsymbol{e}||_2)) \\
&= \boldsymbol{e}^T(I - J_\beta J_\gamma)^T(I - J_\beta J_\gamma)\boldsymbol{e} - \boldsymbol{o}(||\boldsymbol{e}||_2)^T(I - J_\beta J_\gamma)\boldsymbol{e} \\
&\quad - \boldsymbol{e}^T(I - J_\beta J_\gamma)^T\boldsymbol{o}(||\boldsymbol{e}||_2) + \boldsymbol{o}(||\boldsymbol{e}||_2)^T\boldsymbol{o}(||\boldsymbol{e}||_2) \\
&= \boldsymbol{e}^T(I - J_\beta J_\gamma)^T(I - J_\beta J_\gamma)\boldsymbol{e} + \boldsymbol{o}(||\boldsymbol{e}||_2^2) \\
&\leq \mu||\boldsymbol{e}||_2^2 + |o(||\boldsymbol{e}||_2^2)|
\end{aligned}$$

where μ is the largest eigenvalue of $(I-J_\beta J_\gamma)^T(I-J_\beta J_\gamma)$. If $\boldsymbol{e}$ is sufficiently small so that $|o(||\boldsymbol{e}||_2^2))| < (1 - \mu)||\boldsymbol{e}||_2^2$, then

$$||\hat{\boldsymbol{\lambda}}_U - \phi(W^1\hat{\boldsymbol{\lambda}}_C^*)||_2^2 \leq ||\boldsymbol{e}||_2^2 = ||\hat{\boldsymbol{\lambda}}_U - \phi(E[\overline{\boldsymbol{U}}^f])||_2^2$$

$\square$

Acknowledgments

References

Bengio, Y., Lee, D.-H., Bornschein, J., and Lin, Z. (2015). Towards biologically plausible deep learning. *arXiv preprint arXiv:1502.04156*.

Budd, J. M. (1998). Extrastriate feedback to primary visual cortex in primates: a quantitative analysis of connectivity. *Proceedings of the Royal Society of London B: Biological Sciences*, 265(1400):1037–1044.

Burbank, K. S. (2015). Mirrored STDP Implements Autoencoder Learning in a Network of Spiking Neurons. *PLoS Comput Biol*, 11(12):e1004566.

Burbank, K. S. and Kreiman, G. (2012). Depression-Biased Reverse Plasticity Rule Is Required for Stable Learning at Top-Down Connections. *PLoS Comput Biol*, 8(3):e1002393.

Cadieu, C. F., Hong, H., Yamins, D. L. K., Pinto, N., Ardila, D., Solomon, E. A., Majaj, N. J., and DiCarlo, J. J. (2014). Deep neural networks rival the representation of primate IT cortex for core visual object recognition. *PLoS Comput Biol*, 10(12):e1003963.

Cox, D. and Dean, T. (2014). Neural Networks and Neuroscience-Inspired Computer Vision. *Current Biology*, 24(18):R921–R929.

Crick, F. (1989). The recent excitement about neural networks. *Nature*, 337(6203):129–132.

Grossberg, S. (1987). Competitive learning: From interactive activation to adaptive resonance. *Cognitive science*, 11(1):23–63.

Hangya, B., Ranade, S., Lorenc, M., and Kepecs, A. (2015). Central cholinergic neurons are rapidly recruited by reinforcement feedback. *Cell*, 162(5):1155–1168.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554.

Karnani, M. M., Jackson, J., Ayzenshtat, I., Hamzehei Sichani, A., Manoocheri, K., Kim, S., and Yuste, R. (2016). Opening holes in the blanket of inhibition: Localized lateral disinhibition by VIP interneurons. *The Journal of Neuroscience*, 36(12):3471–3480.

Khaligh-Razavi, S.-M. and Kriegeskorte, N. (2014). Deep supervised, but not unsupervised, models may explain IT cortical representation. *PLoS Comput Biol*, 10(11):e1003915.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Kubilius, J., Bracci, S., and Op de Beeck, H. P. (2016). Deep neural networks as a computational model for human shape sensitivity. *PLoS Comput Biol*, 12(4):e1004896.

Körding, K. P. and König, P. (2000). Learning with two sites of synaptic integration. *Network: Computation in Neural Systems*, 11(1):25–39.

Larkum, M. E., Nevian, T., Sandler, M., Polsky, A., and Schiller, J. (2009). Synaptic Integration in Tuft Dendrites of Layer 5 Pyramidal Neurons: A New Unifying Principle. *Science*, 325(5941):756–760.

Larkum, M. E., Waters, J., Sakmann, B., and Helmchen, F. (2007). Dendritic spikes in apical dendrites of neocortical layer 2/3 pyramidal neurons. *The Journal of Neuroscience*, 27(34):8999–9008.

Larkum, M. E., Zhu, J. J., and Sakmann, B. (1999). A new cellular mechanism for coupling inputs arriving at different cortical layers. *Nature*, 398(6725):338–341.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lee, D.-H., Zhang, S., Fischer, A., and Bengio, Y. (2015). Difference target propagation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 498–515. Springer.

Letzkus, J. J., Kampa, B. M., and Stuart, G. J. (2006). Learning rules for spike timing-dependent plasticity depend on dendritic synapse location. *Journal of Neuroscience*, 26(41):10420–10429.

Li, Y., Li, H., Xu, Y., Wang, J., and Zhang, Y. (2016). Very deep neural network for handwritten digit recognition. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 174–182. Springer.

Liao, Q., Leibo, J. Z., and Poggio, T. (2015). How Important is Weight Symmetry in Backpropagation? *arXiv preprint arXiv:1510.05067*.

Lillicrap, T. P., Cownden, D., Tweed, D. B., and Akerman, C. J. (2014). Random feedback weights support learning in deep neural networks. *arXiv preprint arXiv:1411.0247*.

Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

Manita, S., Suzuki, T., Homma, C., Matsumoto, T., Odagawa, M., Yamada, K., Ota, K., Matsubara, C., Inutsuka, A., Sato, M., Ohkura, M., Yamanaka, A., Yanagawa, Y., Nakai, J., Hayashi, Y., Larkum, M., and Murayama, M. (2015). A top-down cortical circuit for accurate sensory perception. *Neuron*, 86(5):1304–1316.

Marblestone, A., Wayne, G., and Kording, K. (2016). Towards an integration of deep learning and neuroscience. *arXiv preprint arXiv:1606.03813*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Murayama, M., Perez-Garci, E., Nevian, T., Bock, T., Senn, W., and Larkum, M. E. (2009). Dendritic encoding of sensory stimuli controlled by deep cortical interneurons. *Nature*, 457(7233):1137–1141.

Pfeffer, C. K., Xue, M., He, M., Huang, Z. J., and Scanziani, M. (2013). Inhibition of inhibition in visual cortex: the logic of connections between molecularly distinct interneurons. *Nat Neurosci*, 16(8):1068–1076.

Pi, H.-J., Hangya, B., Kvitsiani, D., Sanders, J. I., Huang, Z. J., and Kepecs, A. (2013). Cortical interneurons that specialize in disinhibitory control. *Nature*, 503(7477):521–524.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323:9.

Scellier, B. and Bengio, Y. (2016). Towards a biologically plausible backprop. *arXiv preprint arXiv:1602.05179*.

Schiller, J., Major, G., Koester, H. J., and Schiller, Y. (2000). NMDA spikes in basal dendrites of cortical pyramidal neurons. *Nature*, 404(6775):285–289.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Sjöström, P. J. and Häusser, M. (2006). A Cooperative Switch Determines the Sign of Synaptic Plasticity in Distal Dendrites of Neocortical Pyramidal Neurons. *Neuron*, 51(2):227–238.

Spratling, M. W. (2002). Cortical Region Interactions and the Functional Role of Apical Dendrites. *Behavioral and Cognitive Neuroscience Reviews*, 1(3):219–228.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Steriade, M., Timofeev, I., and Grenier, F. (2001). Natural Waking and Sleep States: A View From Inside Neocortical Neurons. *Journal of Neurophysiology*, 85(5):1969–1985.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4(2).

Urbanczik, R. and Senn, W. (2014). Learning by the dendritic prediction of somatic spiking. *Neuron*, 81(3):521–528.

Yamins, D. L. K. and DiCarlo, J. J. (2016). Using goal-driven deep learning models to understand sensory cortex. *Nat Neurosci*, 19(3):356–365.