Layer-skipping connections facilitate training of layered networks using equilibrium propagation.

Jimmy Gammell, Adam McCaughan

February 27, 2020

Abstract

Equilibrium propagation is a learning framework for energy-based networks that can be implemented by neurons that perform only one type of computation in the prediction and correction phases of training, and that computes parameter corrections for a given neuron using only the activation values of directly-connected neurons. This makes it an appealing candidate for implementation in neuromorphic analog hardware, and marks a step forward in the search for a biologically-plausible implementation of deep learning. However, in previous implementations of equilibrium propagation, layered networks of any depth suffered from a vanishing gradient problem that has not yet been solved in a simple or biologically-plausible way. In this paper, we demonstrate that modifying the layered topology by adding random layer-skipping connections in a manner inspired by small-world networks can counteract the vanishing gradient problem to significantly facilitate training. This approach could be conveniently implemented in neuromorphic analog hardware and is biologically-plausible.

1 Introduction

The equilibrium propagation learning framework [?] is a method for training a class of energy-based networks, the prototype for which is the continuous Hopfield network [?]. It is appealing as a framework that could be implemented in neuromorphic analog hardware because unlike in backpropagation, neurons are required to perform only one type of computation in the prediction of correction phases of training, and the parameters of a neuron can be updated using only the activations of neurons to which it is directly connected - there is no need for feedback paths through which to transmit information about the parameters and states of neurons across the entire network. Backpropagation is not biologically plausible, and a major reason is that credit assignment to a given neuron requires precise knowledge of the nonlinearities and derivatives of all neurons in the feedforward path between that neuron and the output [?]; equilibrium propagation avoids this issue. Another major reason is that it is implausible for the forward and backwards propagation phases to use different computations requiring different computational circuits [?]; equilibrium propagation also avoids this issue.

It has been demonstrated [?] that a continuous Hopfield network with a basic multilayer topology can be trained on MNIST [?] through the equilibrium propagation framework. However, previous implementations encountered a vanishing gradient problem that significantly impedes training of networks with several hidden layers. Given that network depth is critical for performance on difficult datasets [?, ?], and that convergence to a low error rate on MNIST is a low bar for a network to meet, this is a nontrivial issue. It has been demonstrated [?] that the problem can be solved by using a unique learning rate for parameters at different depths in the network, with deeper parameters trained with larger learning rates to counteract gradient attenuation with depth. This approach is unappealing because (1) it introduces additional hyperparameters that must be tuned, (2) it would be inconvenient to implement and tune unique learning rates in analog hardware, and (3) it seems unlikely that it is biologically plausible.

The purpose of this paper is to introduce a modification to the basic multilayer topology that can counteract the vanishing gradient problem in the context of energy-based networks trained using equilibrium propagation. We fully connect a network's layers, then replace a small portion of its connections with random layer-skipping connections, in a manner inspired by small-world networks [?]. We achieve 0% training error and under 2.5% test error on MNIST using a network with 3 hidden layers and no regularization term in its cost function. These error rates are competitive (**this may be too generous) in the context of similar biologically-motivated networks [?] and are the same as those achieved by a basic multilayer network with 3 hidden layers in the original paper using a basic topology and unique learning rates [?], albiet they are achieved after around 25% more epochs. Our method adds only one additional hyperparameter [1] - the number of connections to randomly replace - and could be implemented with relative ease in analog hardware. Layer-skipping connections seem biologically-plausible, and small-world networks have been documented in biological brains [?]. Similar techniques have been used with some success in convolutional [?, ?] and basic multilayer feedforward [?, ?] networks with varying degrees of success. Our findings in this paper show that layer-skipping connections are effective enough to be appealing in contexts where simplicity and biological plausibility are important, but would likely not make sense for use in a digital implementation of equilibrium propagation where performance is a primary consideration.

2 Background and Theory

2.1 Equilibrium propagation

Equilibrium propagation [?] is a learning framework for energy-based networks that trains their parameters by approximating gradient descent on some arbitrary cost function. It is applicable to any network with dynamics characterized by evolution to a fixed point of an associated energy function; in this and in the original paper, it is implemented on a continuous Hopfield network [?]. For a variety of reasons outlined in [?], backpropagation is not biologically-plausible. One of the major reasons is that to correct the parameters of a given neuron, backpropagation requires precise information about the activations and nonlinearities of all neurons in the corresponding feedforward path, and no mechanism capable of delivering this information has been observed in biological brains. Equilibrium propagation avoids this problem by approximating the gradient of an arbitrary cost function with respect to a neuron's parameters using only the activations of neurons to which it directly connects. Another major reason backpropagation is not biologically-plausible is that it requires neurons to perform distinct computations (implemented using distinct circuitry) in the forward and backward propagation phases of training, and no evidence suggests that biological neurons perform distinct computations during distinct phases of learning. In contrast, equilibrium propagation requires only one behavior of neurons in both the prediction (free) and correction (weakly-clamped) phases of training: that over time they adjust their activation functions to perform gradient descent on an associated energy function (though this is still biologically-implausible because it requires two distinct phases of training). In addition to enhancing its biological plausibility, these traits also make equilibrium propagation appealing as a framework to implement on neuromorphic analog hardware because they would limit the complexity required of neurons and the amount of infrastructure needed to operate and train them.

2.1.1 Implementation in a continuous Hopfield network

Here we summarize the dynamics of a continuous Hopfield network trained using equilibrium propagation; a more-thorough and more-general treatment, on which this summary is based, is given in the original paper [?].

Consider an arbitrary network with two subsets of its neurons designated as an input and output layer. Let $\boldsymbol{x}$, $\boldsymbol{h}$, and $\boldsymbol{y}$ denote, respectively, vectors containing the activations of its input, hidden and output layers, $\boldsymbol{s} = \{\boldsymbol{h}, \boldsymbol{y}\}$ denote its state and $\boldsymbol{u} = \{\boldsymbol{x}, \boldsymbol{h}, \boldsymbol{y}\}$ denote its full set of neurons. Let $\boldsymbol{W}$ and $\boldsymbol{b}$ denote its weights and biases and ρ denote the activation function of a neuron; here and in the original paper it is a hardened sigmoid function

$$\rho(x) = \begin{cases} 0 & x < 0 \\ x & 0 \leq x \leq 1 \\ 1 & x > 1 \end{cases} \tag{1}$$

[1] In our current implementation we tune the initial weights for added connections independently of those of preexisting connections. We have found that networks are not very sensitive to these initializations and that performance is good if they are in the same ballpark as initializations of preexisting connections (though we have no mathematical justification for this); thus, we do not consider this to be an additional hyperparameter.

with

$$\rho'(x) := \begin{cases} 0 & x < 0 \\ 1 & 0 \leq x \leq 1 \\ 0 & x > 1 \end{cases} \tag{2}$$

where the derivative is defined to be 1 at endpoints of its support to avoid saturation. Let $\boldsymbol{x}_d$ and $\boldsymbol{y}_d$ denote the input and target output from a training dataset during a given batch.

A prediction $\boldsymbol{y}$ of $\boldsymbol{y}_d$ is generated by clamping $\boldsymbol{x}$ to $\boldsymbol{x}_d$, then evolving to a local minimum of an energy function

$$E(\boldsymbol{u}) = \frac{1}{2}\sum_i u_i^2 - \frac{1}{2}\sum_{i \neq j} W_{ij}\rho(u_i)\rho(u_j) - \sum_i b_i \rho(u_i). \tag{3}$$

This initial prediction-generating evolution is denoted the free phase. Consider some cost function $C(\boldsymbol{u}, \boldsymbol{W}, \boldsymbol{b}, \boldsymbol{y}_d)$; in this and in the original paper it is a quadratic error function

$$C(\boldsymbol{y}, \boldsymbol{y}_d) = \frac{1}{2}||\boldsymbol{y} - \boldsymbol{y}_d||_2^2, \tag{4}$$

but the algorithm is applicable to an arbitrary cost function - for example, one including a regularization term. After the free phase, the network is then evolved to a local minimum of a total energy function

$$F(\boldsymbol{u}, \boldsymbol{y}_d, \beta) = E(\boldsymbol{u}) + \beta C(\boldsymbol{y}, \boldsymbol{y}_{target}) \tag{5}$$

where β, called the clamping factor, is a small constant. Note that $E(\boldsymbol{u}) = F(\boldsymbol{u}, \boldsymbol{y}_d, 0)$; therefore, the free phase can (and, henceforth, will) be interpreted as evolution to equilibrium on $F(\boldsymbol{u}, \boldsymbol{y}_d, 0)$. This second phase is denoted the weakly-clamped phase, and can be interpreted as first nudging the output neurons in the direction of the target output, then allowing remaining neurons to adjust towards a configuration that would have predicted the more-correct nudged output (though it is unclear that this interpretation would hold if a regularization term were added to the cost function). The network evolves to equilibrium by performing gradient descent on F through the equation of motion

$$\frac{d\boldsymbol{s}}{dt} = -\frac{\partial F}{\partial \boldsymbol{s}}. \tag{6}$$

The states of the network after the free and weakly-clamped phases are used to compute correction terms so must be saved; after evolution on some set of training data we will denote these states, respectively, $\boldsymbol{s}^0$ and $\boldsymbol{s}^\beta$. The process for training over a set of data is as follows:

1. Perform the free-phase evolution; evolve to equilibrium on the energy function $F(\boldsymbol{u}, \boldsymbol{y}_d, 0)$ by updating $\boldsymbol{s}$ according to equation 6. Record the equilibrium state $\boldsymbol{s}^0$.

2. Perform the weakly-clamped evolution; evolve for a short amount of time towards equilibrium on the energy function $F(\boldsymbol{u}, \boldsymbol{y}_d, \beta)$, again according to equation 6, using $\boldsymbol{s}^0$ as a starting point. Record the equilibrium state $\boldsymbol{s}^\beta$.

3. Compute the correction to each weight in the network using the function

$$\Delta W_{ij} = \alpha \frac{1}{\beta}(\rho(u_i^\beta)\rho(u_j^\beta) - \rho(u_i^0)\rho(u_j^0)) \tag{7}$$

   where α, the learning rate, is a positive constant. Adjust the weights using $W_{ij} := W_{ij} + \Delta W_{ij}$.

4. Compute the correction to each bias in the network using the function

$$\Delta b_i = \alpha \frac{1}{\beta}(\rho(u_i^\beta) - \rho(u_i^0)) \tag{8}$$

   and adjust the biases using $b_i := b_i + \Delta b_i$.

Note that the correction to a weight depends only on the activations of neurons it directly affects, and the correction to a bias depends only on the activation of the neuron it directly affects. This is in contrast to backpropagation, where to correct a weight or bias l layers from the output it would be necessary to know the activations, derivatives and weights of all neurons between 0 and $l - 1$ layers from the output.

2.1.2 Approximation of the equation of motion

In analog hardware, the dynamics described by equation 6 could be implemented efficiently using leaky integrator neurons. On digital hardware, however, it is necessary to discretize and approximate the differential equation of motion; we now describe the approximation used here and in the original paper. Let $\boldsymbol{s}[n]$ denote the state of the network after n iterations of the approximation, N denote the total number of iterations and ϵ the size of each iteration. $\boldsymbol{s}[0]$ is the initial state of the network; this may be random or, as in this and the original paper, zero. Let

$$s_i[n] = s_i[n-1] - \epsilon \frac{\partial F}{\partial s_i}(\boldsymbol{u}, \boldsymbol{y}_d, \beta),\ n = 1, \dots, N. \tag{9}$$

Then the state of the network after time $\tau = \epsilon N$ is given by

$$\int_0^\tau \frac{d\boldsymbol{s}}{dt} dt \approx \boldsymbol{s}[N]. \tag{10}$$

We denote the number of iterations in the free and weakly-clamped phases N_{free} and $N_{weakly-clamped}$, respectively. It is only necessary to run the weakly-clamped phase long enough to observe the initial direction in which the network evolves and to allow the perturbation of the output to influence all layers in the network, so typically for a network with L layers $N_{free} >> N_{weakly-clamped} > L$.

2.1.3 Applicable methods for enhanced biological-plausibility

A major reason backpropagation is not biologically-plausible is that it implies a set of feedback connections with weights and destinations identical to the feedforward connections; this is also an issue in equilibrium propagation. However, recent work [?] has shown that networks can still train if feedback weights are initialized randomly instead of symmetrically with feedforward connections, and that with training their weights will become similar to forward weights. These findings could be applied to an energy-based network trained using equilibrium propagation.

Biological neurons are known to communicate through spikes with binary values, suggesting that a spike timing based activation function for neurons is more biologically-plausible than popular functions. It has been shown [?] that an energy-based network with a spike timing-based activation function can be trained using equilibrium propagation.

The method presented in this paper allows networks with multiple layers to train effectively through a biologically-plausible tweak to their topology. Layer-skipping connections have been documented in biological brains [?].

2.2 Vanishing gradient problem

It has been observed [?] that energy-based networks with a layered topology trained using equilibrium propagation suffer from a vanishing gradient problem: the magnitude of the weight correction matrix to weights between a pair of layers attenuates exponentially with the number of layers between that pair and the output layer. This problem is familiar in the context of conventional networks trained through backpropagation, where if the magnitude of the correction matrix to weights connecting the output layer to the last hidden layer is ΔW and if the expected magnitude of the derivative of neurons' activation function is $\alpha < 1$, then the magnitude of the correction matrix to weights connecting layers l and $l-1$ from the output is expected to be $\alpha^l \Delta W$. In conventional networks, the vanishing gradient problem can be effectively addressed by initializing weights to make activation variances and backpropagated gradient variances approximately uniform with respect to depth as described in [?] and by using activation functions with unity derivatives (**find reference), such as rectified linear units $\rho(x) = \text{Max}\{0, x\}$ or hardened sigmoids (equation 1).

Neither of these approaches are effective at solving the vanishing gradient problem in the context of equilibrium propagation. In the context of energy-based networks the situation is more-complicated because there is no straightforward causal relationship between the activations of two given neurons; instead, neurons evolve as a system. The fact that conventional approaches are ineffective suggests that the cause of the problem is different than in conventional networks.

One factor that probably contributes to the vanishing gradient problem is that when the output layer of neurons is perturbed at the beginning of the weakly-clamped phase, it takes an additional iteration of the approximation of equation 6 for the perturbation to influence the last hidden layer, then an additional iteration for the second-to-last hidden layer, and so on. We do not believe this factor to be a significant contributor to the vanishing gradient problem because increasing $N_{weakly-clamped}$ does not significantly affect network performance. This is supported by the following rough theoretical formulation:

Assume that a typical neuron in the network changes by a small constant amount δ after each iteration of equation 6 (this is reasonable because we run only the early stages of evolution to the weakly-clamped equilibrium), and assume that the typical neuron is in state s_0 at the end of the free phase. After N iterations, a neuron in the output layer will have changed by $N\delta$ and one in a deep layer l layers from the output will have changed by $(N-l)\delta$. Assume further that s_0 is in the middle of the support of ρ' where ρ is the hardened sigmoid function from equation 1, so that we can replace $\rho(x)$ by x. Then it follows from equation 7 that a typical neuron in the output layer will be corrected by

$$\Delta w^{\text{output}} = \frac{\alpha}{\beta}((N\delta + s_0)((N-1)\delta + s_0) - s_0^2) \approx \frac{\alpha}{\beta}(N^2\delta^2 + 2s_0 N\delta) \tag{11}$$

and similarly a typical neuron from the deep layer will be corrected by

$$\Delta w^l \approx \frac{\alpha}{\beta}((N-l)^2\delta^2 + 2s_0(N-l)\delta) \tag{12}$$

(where we have approximated $N \approx N-1$ and $N-l+1 \approx N-l$), so the ratio of these corrections will be

$$\frac{\Delta w^l}{\Delta w^{\text{output}}} \approx \frac{(N-l)^2\delta^2 - 2s_0(N-l)\delta}{N^2\delta^2 + 2s_0 N\delta} = \frac{N-l}{N}\frac{(N-l)\delta + 2s_0}{N\delta + 2s_0} \approx 1 - \frac{l}{N} \tag{13}$$

(where we have approximated $\frac{(N-l)\delta+2s_0}{N\delta+2s_0} \approx 1$). We have observed that

$$\frac{\Delta w^l}{\Delta w^{\text{output}}} \sim e^{-l}, \tag{14}$$

so it seems unlikely that this phenomenon is a primary cause of the vanishing gradient problem.

The vanishing gradient problem causes deep networks to train very slowly, which is obviously undesirable. It also causes the magnitudes of corrections to weights between pairs of layers to differ by many orders of magnitude, which would be problematic in an analog implementation of the framework using neurons with limited numerical precision.

In the original paper the vanishing gradient problem was solved by using unique learning rates for each layer, chosen to make the magnitudes of corrections to neurons' parameters uniform regardless of depth in the network. While this method was effective, it is unappealing because it seems unlikely to take place in a biological brain, it would add complexity to an analog implementation of the framework, and it introduces more hyperparameters that must be tuned to train a network. Our topology solves the problem without these unappealing characteristics.

2.3 Small-world networks

Our topology was inspired by small-world graph topology as described in [?]. Qualitatively, small-world topology is appealing because it is characterized by a small mean shortest path between a pair of vertices (neurons) in a graph (network) with a large amount of clustering, which seems likely to reduce attenuation to the gradient of parameters of deep neurons by providing a low-attenuation path to the output layer, while largely preserving the regimented structure that has contributed to the success of deep neural networks. The topology is also appealing because it has been observed to occur in biological brains [?].

The small-worldness of a network is typically characterized by a characteristic path length L and a clustering coefficient C. A graph is small-world if L is small relative to the L of a random network with the same number of vertices and edges and C is not substantially smaller than the C of that random network. The characteristic path length is the minimal number of edges in a path joining a pair of vertices, averaged over all pairs of vertices in the network. Specifically, for a graph with N vertices where l_{ij} denotes the smallest number of edges needed to connect the two vertices,

$$L = \frac{1}{N(N-1)} \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} l_{ij}. \tag{15}$$

The neighborhood of a vertex is the set of vertices with which it shares an edge. The clustering coefficient is the proportion of pairs of vertices in the neighborhood of a given vertex that share an edge, averaged over all vertices in the network. Specifically, for a graph with N vertices, if $\mathrm{Nbhd}(v_i)$ denotes the neighborhood of vertex v_i, N_i denotes the number of vertices in $\mathrm{Nbhd}(v_i)$, and $c(v_j, v_k) = \begin{cases} 1 & v_k \in \mathrm{Nbhd}(v_j) \\ 0 & \text{else} \end{cases}$,

$$C = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{N_i} \sum_{\substack{v_j \in \\ \mathrm{Nbhd}(v_i)}} \sum_{\substack{v_k \in \\ \mathrm{Nbhd}(v_i), \\ k \neq j}} c(v_j, v_k). \tag{16}$$

An algorithm was introduced in [**?**] to convert a regimented graph into a small-world graph [2]. For each edge in the network, with probability p: randomly pick a pair of vertices that do not share an edge. Add an edge between these vertices, and remove the existing edge. As p is increased L decreases more-quickly than C, and p is typically chosen so that L has decreased substantially and C has just begun to decrease.

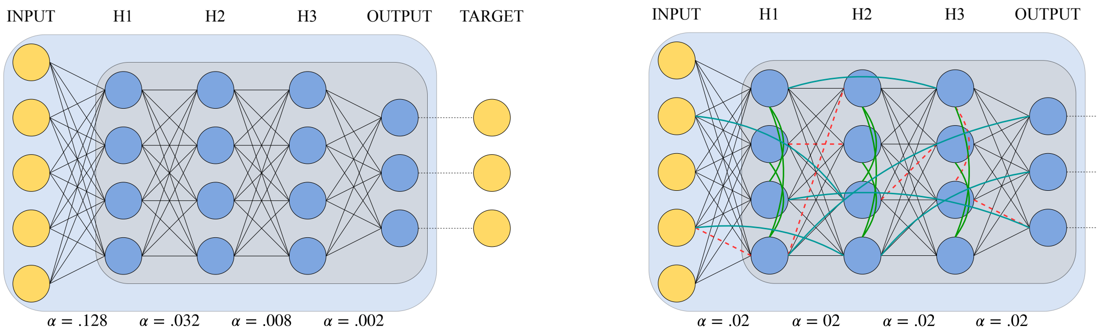
We have found that the performance of a network does not track closely with its clustering coefficient or its characteristic path length. We believe this to be mainly because the characteristic path length depends only on the path to a neuron that attenuates the gradient the least, but in actuality attenuation to the gradient also heavily depends on the number of paths it can take in parallel to a destination. We have had success using networks with $p \approx 8\%$, which is far more than necessary to make the network small-world; nonetheless, we believe that the success of this topology is for reasons similar to those listed above. It is possible that there are multiple factors influencing the performance of a network as connections are replaced - e.g. layer-skipping connections constitute a linear transformation in parallel with hidden layers constituting a nonlinear transformation (described in section 2.4), or layer-skipping connections partially counteract the trend described in section 2.2.

2.4 Nonlinearities learning residuals

In previous work layer-skipping connections have been used to alter the task of nonlinear layers from matching the desired output to matching the residual between the desired output and the output of a previous layer [**?**], [**?**]. This has proven extremely successful for training very-deep networks using backpropagation. If the underlying function describing a training dataset is $\boldsymbol{y} = f(\boldsymbol{x})$, layer-skipping connections can be run in parallel with a collection of nonlinear layers so that they need only match the function $f(\boldsymbol{x}) - \boldsymbol{x}$. The rationale behind this technique is that not all nonlinearities are useful for approximating f, and if a nonlinearity is not useful it must be trained to approximate an identity mapping, which may be difficult. By adding a parallel identity mapping or linear mapping it becomes easy to minimize the contribution by a set of nonlinearities - their weights can simply be driven towards zero.

This is a possible explanation for the effectiveness of our topology. While we do not explicitly structure layer-skipping connections as identity mappings, they could potentially provide a means for the output of a deep layer to at-least partially pass an unhelpful nonlinear transformation.

[2] Our algorithm is actually slightly different: instead of replacing each edge with a probability p, we replace a specific number n of the edges currently in the network. Therefore, for a network with N edges our algorithm is equivalent to the one from [**?**] with $p = 1 - (\frac{N-1}{N})^n$. For ease of comparison with other work and to contextualize how many edges have been replaced, we will henceforth describe networks using our topology in terms of p and not n.



(a) Topology of the basic multilayer network. There is a connection between every pair of neurons in adjacent layers. There are no connections between neurons in the same layer. There are no layer-skipping connections. To combat the vanishing gradient problem, the learning rate is higher for weights deeper in the network.

(b) Changes in our topology with respect to the basic topology. Green connections were added to connect every pair of neurons within a layer. Red connections (8% of existing connections) were chosen at random and removed. Blue connections were chosen at random and added to replace removed connections. These changes allow a uniform learning rate regardless of a weight's depth in the network.

Figure 1

3 Related work

Much research has been done in pursuit of a biologically-plausible deep learning algorithm. [?], [?] and [?] are other algorithms that address the weight transport problem. [?] addresses the need for symmetric feedback weights, a problem not addressed by equilibrium propagation. [?] implements equilibrium propagation using spiking neurons like are present in a biological brain. [?] and [?] discuss the criteria such an algorithm would need to satisfy. [?] surveys promising biologically-motivated algorithms and evaluates their effectiveness on hard algorithms.

[?], [?] and [?] discuss neuromorphic architecture that could potentially implement equilibrium propagation as an analog computer.

Layer-skipping connections have been explored in other contexts. [?] and [?] use layer-skipping connections as a linear transformation in parallel with nonlinear layers to great effect in very-deep convolutional networks. [?] and [?] use a small-world topology in conventional multilayer feedforward networks.

The vanishing gradient problem was a big obstacle to the training of conventional deep networks through backpropagation, and [?], [?] and [?] provide effective means for solving it in that context.

4 Implementation

We implemented the equilibrium propagation learning framework [?] using the Pytorch library [?], and verified that we could recreate the experiments run in [?]. All of our networks use a hardened sigmoid activation function

$$\rho(x) = \text{Max}\{0, \text{Min}\{x, 1\}\} \tag{17}$$

with

$$\rho'(x) := \begin{cases} 1 & 0 \leq x \leq 1 \\ 0 & \text{else} \end{cases} \tag{18}$$

(the derivative at $x = 0$ and $x = 1$ is chosen to be 1 to avoid saturation). All of our networks use a squared-error cost function with no regularization term. The bulk of our testing was done on the MNIST dataset [?] with the following 3 networks.

4.1 Basic topology with unique learning rates

We recreated the 5-layer network used in [?]. Its topology is illustrated in figure 1a.

The network consists of an input layer with 28^2 neurons, 5 hidden layers with 500 neurons, and an output layer with 10 neurons. There are connections between every pair of neurons in adjacent layers, no connections between neurons in the same layer, and no layer-skipping connections.

The weight matrix was initialized using the Glorot-Bengio initialization scheme [?]: a weight connecting layer j with n_j neurons to layer $j+1$ with n_{j+1} neurons is drawn from

$$U[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}]. \tag{19}$$

For effective training it is necessary to use unique learning rates to train the weights connecting each pair of layers. We use the same learning rates that were used for experiments in [?]: .128, .032, .008, .002, in order of the deepest to the shallowest layer. Each neuron has a bias term that is trained using the learning rate corresponding to the weights of the neuron's input connections, e.g. the biases of the first hidden layer are trained with a learning rate of .128.

4.2 Basic topology with single learning rate

To provide a reference point for our topology we ran tests on a network that is identical to that in section 4.1 except that it has a single learning rate of .02 across the entire network.

4.3 Our topology

To generate a network with our topology we use as a starting point the topology described in section 4.1. We then connect every pair of neurons in a given layer. Finally, for a given number of connections, we remove an existing connection at random and replace it with a random connection between two unconnected neurons in the network. [3] We do not add connections within the input or output layers and we do not allow neurons to connect to themselves. We have seen good results with $p \approx 8\%$ of a network's connections in this manner.

This topology allows us to train the network with a uniform learning rate of .02 across the entire network. Weights of connections between pairs of neurons in adjacent layers are still initialized based on (19). Weights of intralayer connections and layer-skipping connections are drawn from $U[-.05, .05]$,where the value .05 was chosen empirically. While we do not theoretically justify this initialization, it allows for good performance.

4.4 Tracking the training rates of individual pairs of layers

To observe the nature and extent of the vanishing gradient on different networks we periodically measured the root-mean-square correction to the weights between individual pairs of layers. Specifically, if $\Delta w_{ij}^l(b)$ denotes the correction to the connection weight between the i^{th} neuron in layer l and the j^{th} neuron in layer $l+1$ in response to batch b and N^l denotes the number of neurons in layer l, we define

$$\Delta w^l(b) := \sqrt{\frac{\sum_{i=1}^{N^l} \sum_{j=1}^{N^{l+1}} (\Delta w_{ij}^l(b))^2}{N^l N^{l+1}}} \tag{20}$$

as our metric of the extent to which the weights between layers l and $l+1$ trained in response to batch b. Note that this measurement ignores the training of the weights of intralayer and layer-skipping connections.

Since this measurement tends to be volatile, for clarity we plot the average of $\Delta w^l(b')$ for b' in a neighborhood of batch b. Specifically, for some n we define

$$\Delta \hat{w}^l(b) := \frac{1}{2n+1} \sum_{b'=b-n}^{b+n} \Delta w^l(b'). \tag{21}$$

[3] We have also tried adding new connections without replacing existing ones, and observed roughly the same performance. In this paper we replace connections to limit the number of free parameters added to the network and avoid the possibility of conflating improvement due to the changed topology with that due to additional parameters.

Network topology	Learning rate(s)	ϵ	β	N_{free}	$N_{weakly-clamped}$
Basic, unique learning rates	.128, .032, .008, .002	.5	1.0	500	8
Basic, one learning rate	.02	.5	1.0	500	8
Our topology, $p = 7.56\%$	.02	.5	1.0	500	8

Table 1: Hyperparameters of networks tested on MNIST dataset

Then by plotting the traces $\Delta\hat{w}^l(b)$ for each layer l with respect to b we can compare the extent to which each layer trained over a given span of batches.

4.5 Tracking the spread of training rates as a scalar

We observe that in networks with our topology, weights connecting to the output layer tend to train with a rate around an order of magnitude higher than deeper weights, independently of the number of replaced connections. We also note that with few replaced connections it is common for training rates to vary by several orders of magnitude. These observations informed our definition of a scalar metric of the spread of the training rates of a network's layers. For a network trained for N batches with $\Delta w^l(b)$ measured every n batches and where $\Delta w^l(b)$ is defined by (20), we first define

$$\Delta w^l := \sum_{b=1}^{\lfloor N/n \rfloor} \Delta w^l(nb) \tag{22}$$

as a metric of how much the weights between layers l and $l+1$ were corrected over the course of training. For a network with $2L+2$ layers where $l_{min}(i)$ denotes the pair of layers with the i^{th} smallest Δw^l, we finally define

$$\text{Spread} := \frac{\sum_{i=L+1}^{2L} \Delta w^{l_{min}(i)}}{\sum_{i=1}^{L} \Delta w^{l_{min}(i)}}. \tag{23}$$

Note that this metric ignores the difference between deep pairs and the pair connecting to the output layer, as it is assumed to be independent of the number of replaced connections.

5 Results

5.1 MNIST dataset

We compared the three network topologies described in sections 4.1, 4.2, and 4.3 on the MNIST dataset [?]. The hyperparameters for the networks are shown in table **??**. All networks were trained for 250 epochs with 50,000 training examples, 10,000 test examples and a batch size of 20.

5.1.1 Network performance comparison

We tracked the error rates of the three networks as they were trained and found that the network with our topology significantly outperforms the basic network with one learning rate, and achieves the same error rates as the basic network with unique learning rates in around 25% more epochs. These results are shown in figure **??**.

Notice that the basic network with unique learning rates converges to 0% training error and 2.5% test error in around 150 epochs and the network with our topology converges to 0% training error and 2.5% test error in around 190 epochs, whereas the basic network with one learning rate fails to converge in 250 epochs and has training and test error around .5% higher than the other two networks. While it is possible that the basic network with one learning rate would converge given enough time, it is clearly inferior to the other two networks. It is also apparent that in the context of the MNIST dataset, our network with a single learning rate is practically interchangeable with the basic network with unique learning rates.

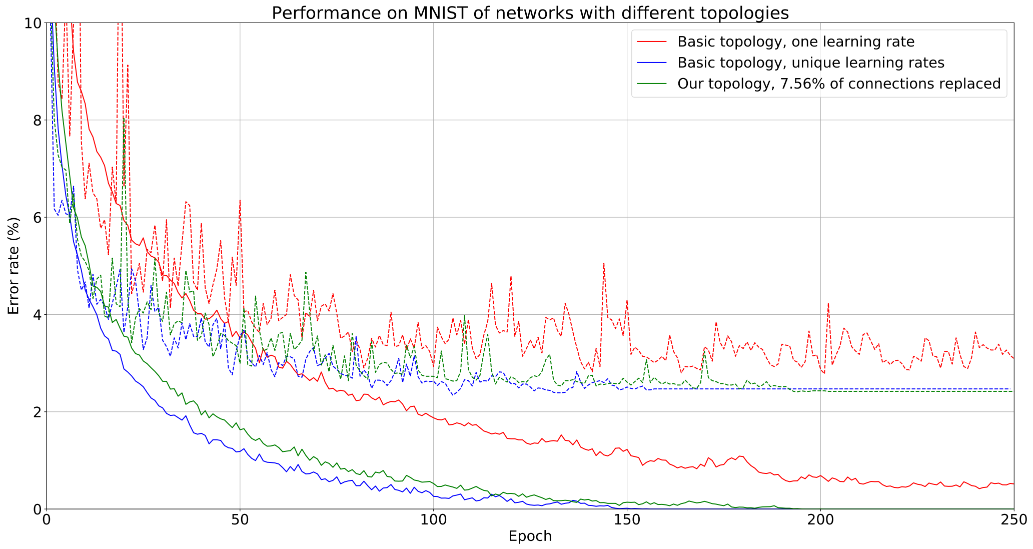


Figure 2: Comparison of network topologies on MNIST dataset. Dotted lines show test error and solid lines show training error. In red is a network with a standard multilayer feedforward topology and a single learning rate. In blue is a network with the same multilayer feedforward topology and unique learning rates for each layer, tuned to promote uniformity in the extent to which each pair of layers trains, as described in [**?**]. In green is a network with a multilayer feedforward topology with fully connected layers and $p = 7.56\%$.

5.1.2 Training rates of individual pairs of layers

We tracked the training rates of individual pairs of layers as described in (20) and (21) and found evidence that the extent of the vanishing gradient problem is the primary cause of the performance disparity seen in section **??**. These results are shown in figure **??**.

Notice in the basic network with one learning rate that there is a significant spread in the training rates of pairs of layers, with the deepest pair training at around 1% of the rate of the shallowest pair. This problem is solved very effectively in the basic network with unique learning rates. The problem appears to be solved effectively for the deepest 3 pairs in the network with our topology, but the output layer still trains significantly faster than the deeper 3 layers. This makes sense if we assume that the important factor in a layer's training rate is its expected path length to the target layer, because every neuron in the output layer connects to the target layer through a single connection, whereas paths starting at deeper neurons must first pass through the output layer before connecting to the target layer.

5.1.3 Error rate after one epoch as connections are added

We tracked the training error after one epoch of a network with our topology, with varying numbers of layer-skipping connections and found that the error rate decays approximately exponentially. These results are shown in figure **??**.

The error rate drops quickly as connections are replaced early on. We believe this is because when a forward connection is added, in addition to providing a path via. one connection between the two connected neurons, it also provides paths via. at most 3 jumps to all pairs of neurons in the two layers, as a result of the intralayer connectivity. This benefit is exhausted when all pairs of layers are connected, leading to the more-gradual improvement later on. It appears to take around 25,000 replaced connections to reach this regime, likely because the extent of attenuation of a gradient depends less on the minimum path length between two neurons and more on the number of low-attenuation paths it has available.

We found that our topology performs significantly worse than the basic topology with one learning rate

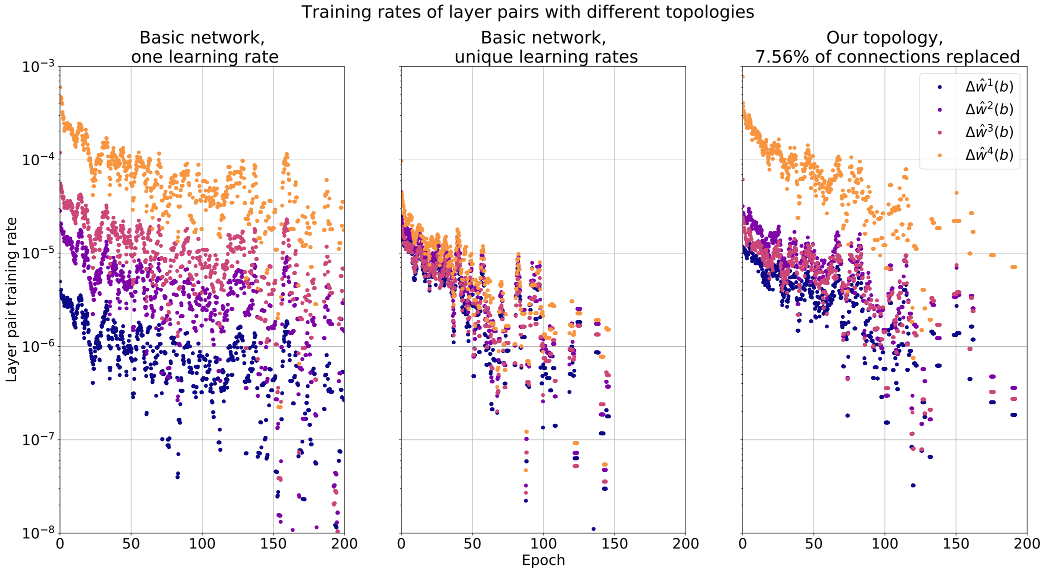


Figure 3: Observation of extent of training of individual layers for different network topologies. Measurements were taken while running trials shown in figure **??**, and have been averaged as described in (**??**). To the left is a network with a standard multilayer feedforward topology and a single learning rate. In the center is a network with the same multilayer feedforward topology and unique learning rates for each layer, as described in [**?**]. To the right is a network with a multilayer feedforward topology with fully-connected layers and $p = 7.56\%$.

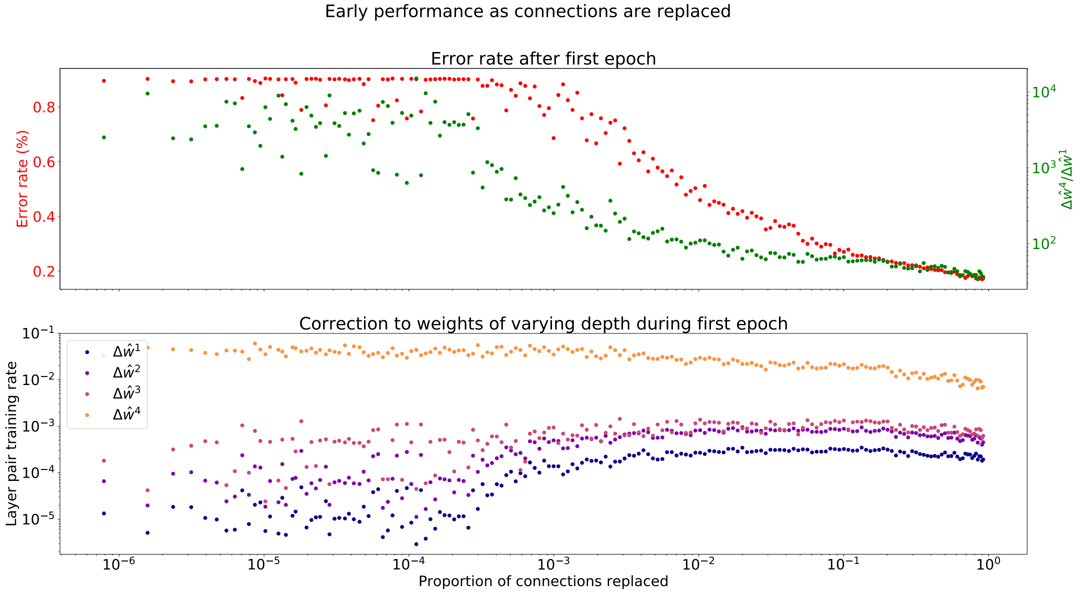


Figure 4: The training error after one epoch of a network with our topology, as connections are replaced.

when few connections are replaced, possibly due to inappropriate hyperparameter choices for intraconnected layers. We have seen some evidence that replacement of connections makes a network more-forgiving of poor weight matrix initialization, but have not thoroughly probed the issue.

6 Conclusion

We believe that our topology is a suitable substitute for unique learning rates as a solution to the vanishing gradient problem. While it is not as effective, it is simpler, more biologically-plausible, and would be easier to implement in an analog computer.

There are several directions in which future research could be taken. It would be useful to find and mathematically-justify an effective weight initialization scheme for a network with our topology. It would be interesting to explore the effect of adding (instead of replacing) layer-skipping connections, training a network, then removing the connections and training it further, to see if they provide a residual benefit even after removal; doing so could allow a network in an analog computer to be trained quickly, and the added connections removed to reduce the power consumption and heat load of the network.

References

[1] my github repository for eqp code.

[2] Sergey Bartunov, Adam Santoro, Blake A. Richards, Geoffrey E. Hinton, and Timothy P. Lillicrap. Assessing the scalability of biologically-motivated deep learning algorithms and architectures. *CoRR*, abs/1807.04587, 2018.

[3] Yoshua Bengio, Dong-Hyun Lee, Jörg Bornschein, and Zhouhan Lin. Towards biologically plausible deep learning. *CoRR*, abs/1502.04156, 2015.

[4] E. Bullmore and O. Sporns. Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature*, 2009.

[5] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Prasad Joshi, Andrew Lines, Andreas Wild, and Hong Wang. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, PP:1–1, 01 2018.

[6] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[8] John Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences of the United States of America*, 81:3088–92, 06 1984.

[9] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[10] Gokul Krishnan, Xiaocong Du, and Yu Cao. Structural pruning in deep neural networks: A small-world approach, 2019.

[11] Y. LeCun and C. Cortes. The mnist database of handwritten digits, 1998.

[12] Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Y. Bengio. Difference target propagation. pages 498–515, 08 2015.

[13] Timothy P. Lillicrap, Daniel Cownden, Douglas B. Tweed, and Colin J. Akerman. Random feedback weights support learning in deep neural networks, 2014.

[14] Mitchell Nahmias, Bhavin Shastri, A.N. Tait, and P.R. Prucnal. A leaky integrate-and-fire laser neuron for ultrafast cognitive computing. *Selected Topics in Quantum Electronics, IEEE Journal of*, 19:1–12, 09 2013.

[15] Peter O'Connor, Efstratios Gavves, and Max Welling. Training a network of spiking neurons with equilibrium propagation. 2018.

[16] Fernando J Pineda. Generalization of back-propagation to recurrent neural networks. *Physical Review Letters*, 59(19):2229–2232, 1987.

[17] Benjamin Scellier and Yoshua Bengio. Towards a biologically plausible backprop. *CoRR*, abs/1602.05179, 2016.

[18] Jeffrey M. Shainline, Sonia M. Buckley, Adam N. McCaughan, Jeffrey T. Chiles, Amir Jafari Salim, Manuel Castellanos-Beltran, Christine A. Donnelly, Michael L. Schneider, Richard P. Mirin, and Sae Woo Nam. Superconducting optoelectronic loop neurons. *Journal of Applied Physics*, 126(4):044902, Jul 2019.

[19] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[20] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks, 2015.

[21] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *CoRR*, abs/1505.00387, 2015.

[22] D. Watts and S. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 1998.

[23] L. Xiaohu, L. Xiaoling, Z. Jinhua, Z. Yulin, and L. Maolin. A new multilayer feedforward small-world neural network with its performances on function approximation. In *2011 IEEE International Conference on Computer Science and Automation Engineering*, 2011.

[24] Xiaohui Xie and Hyunjune Seung. Equivalence of backpropagation and contrastive hebbian learning in a layered network. *Neural computation*, 15:441–54, 03 2003.