INITIALIZED EQUILIBRIUM PROPAGATION FOR BACKPROP-FREE TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep neural networks are almost universally trained with reverse-mode automatic differentiation (a.k.a. backpropagation). Biological networks, on the other hand, appear to lack any mechanism for sending gradients back to their input neurons, and thus cannot be learning in this way. In response to this, Scellier & Bengio (2017) proposed *Equilibrium Propagation* - a method for gradient-based training of neural networks which uses only local learning rules and, crucially, does not rely on neurons having a mechanism for back-propagating an error gradient. Equilibrium propagation, however, has a major practical limitation: inference involves doing an iterative optimization of neural activations to find a fixed-point, and the number of steps required to closely approximate this fixed point scales poorly with the depth of the network. In response to this problem, we propose *Initialized Equilibrium Propagation*, which trains a feedforward network to initialize the iterative inference procedure for Equilibrium propagation. This feed-forward network learns to approximate the state of the fixed-point using a local learning rule. After training, we can simply use this initializing network for inference, resulting in a learned feedforward network. Our experiments show that this network appears to work as well or better than the original version of Equilibrium propagation. This shows how we might go about training deep networks without using backpropagation.

1 INTRODUCTION

Deep neural networks are almost always trained with gradient descent, and gradients are almost always computed with backpropagation. For those interested in understanding the working of the brain in the context of machine learning, it is therefore distressing that biological neurons appear not to send signals backwards.

Biological neurons communicate by sending a sequence of pulses to downstream neurons along a one-way signaling pathway called an "axon". If neurons were doing backpropagation, one would expect a secondary signalling pathway wherein gradient signals travel backwards along axons. This appears not to exist, so it seems that biological neurons cannot be doing backpropagation.

Moreover, backpropagation may not be the ideal learning algorithm for efficient implementation in hardware, because it involves buffering activations for each layer until an error gradient returns. This requirement becomes especially onerous when we wish to backpropagate through many steps of time, or through many layers of depth. For these reasons, researchers are looking into other means of neural *credit assignment* - mechanisms for generating useful learning signals without doing backpropagation.

Recently, Scellier & Bengio (2017) proposed a novel algorithm called *Equilibrium Propagation*, which enables the computation of parameter gradients in a deep neural network without backpropagation. Equilibrium Propagation defines a neural network as a dynamical system, whose dynamics follow the negative-gradient of an energy function. The "prediction" of this network is the fixed-point of the dynamics - the point at which the system settles to a local minimum energy given the input, and ceases to change. Because of this inference scheme, Equilibrium Propagation is impractically slow for large networks - the network has to iteratively converge to a fixed point at every training iteration.

In this work, we take inspiration from Hinton et al. (2015) and distill knowledge from a slow, energy based *equilibrating network* into a fast *feedforward network* by training the feedforward network to predict the fixed-points of the equilibrating network with a local loss. At the end of training, we can then discard the equilibrating network and simply use our feedforward network for test-time inference. We thus have a way to train a feedforward network without backpropagation. The resulting architecture loosely resembles a Conditional Generative Adversarial Network (Mirza & Osindero, 2014), where the feedforward network produces a network state which is evaluated by the energy-based equilibrating network.

To aid the reader, this paper contains a glossary of symbols in Appendix A.

1.1 EQUILIBRIUM PROPAGATION

Equilibrium Propagation (Scellier & Bengio, 2017) is a method for training a Continuous Hopfield Network Hopfield (1984) for classification. The network performs inference by iteratively converging to a fixed-point, conditioned on the input data, and taking the state of the output neurons at the fixed point to be the output of the network. The network's dynamics are defined by an energy function over neuron states s and parameters $\theta = (w, b)$:

$$E_\theta(s, x) = \frac{1}{2}\sum_{i\in\mathcal{S}} s_i^2 - \sum_{i\in S} b_i\rho(s_i) - \sum_{j\in\mathcal{S}, i\in\alpha_j\cap\mathcal{S}} w_{ij}\rho(s_i)\rho(s_j) - \sum_{j\in\mathcal{S}, i\in\alpha_j\cap\mathcal{I}} x_i w_{ij}\rho(s_j) \quad (1)$$

Where $\mathcal{I}$ is the set of input neurons, $\mathcal{S}$ is the set non-input neurons; $s \in \mathbb{R}^{|\mathcal{S}|}$ is the vector of neuron states; $\alpha = \{\alpha_j : j \in \mathcal{S}\}$ denotes the network architecture, where $\alpha_j \subset \{i : i \in \mathcal{I} \cup \mathcal{S}, i \neq j, (j \in \alpha_i \forall i \in \mathcal{S})\}$ is the set of neurons connected to neuron j with the no self-connections and symmetry constraints; x denotes the input vector; and ρ is some nonlinearity; w is a weight matrix with a symmetric constraint: $w_{ij} = w_{ji}$[1], and entries only defined for $\{w_{ij} : i \in \alpha_j\}$ The state-dynamics for non-input neurons, derived from Equation 1, are:

$$\frac{\partial s_j}{\partial t} = -\frac{\partial E_\theta(s, x)}{\partial s_j} = -s_j + \rho'(s_j)\left(b_j + \sum_{j\in\mathcal{S}, i\in\alpha_j\cap\mathcal{S}} w_{ij}\rho(s_i) + \sum_{j\in\mathcal{S}, i\in\alpha_j\cap\mathcal{I}} w_{ij}x_i\right) \forall j \in \mathcal{S} \quad (2)$$

The network is trained using a two-phase procedure, with a negative and then a positive phase. In the negative phase, the network is allowed to settle to an energy minimum $s^- := \arg\min_s E_\theta(s, x)$ conditioned on a minibatch of input data x. In the positive phase, a target y is introduced, and the energy function is augmented to "perturb" the fixed-point of the state towards the target with a "clamping factor" β: $E_\theta^\beta(s, x, y) = E_\theta(s, x) + \beta\frac{\partial C(s_\mathcal{O}, y)}{\partial s}$, where β is a small positive number. The network is allowed to settle to the perturbed state $s^+ := \arg\min_s(E^\beta(s, x, y))$.

Finally, the parameters of the network are learned based on a contrastive loss between the negative-phase and positive-phase energy, which can be shown to be proportional to the gradient of the output loss in the limit of $\beta \to 0$:

$$\Delta\theta = -\frac{\eta}{\beta}\left(\frac{\partial E_\theta(s^+, x)}{\partial\theta} - \frac{\partial E_\theta(s^-, x)}{\partial\theta}\right) \propto -\frac{\partial C(s^-_\mathcal{O}, y)}{\partial\theta} \quad (3)$$

Where η is some learning rate; $\mathcal{O} \subseteq \mathcal{S}$ is the subset of output neurons. This results in a local learning rule, where parameter changes only depend on the activities of the pre- and post-synaptic neurons:

[1] However as described in Scellier et al. (2018), the symmetry requirement can be relaxed without significantly impacting performance. In this case there is no energy function, but one simply defines the network in terms of the state dynamics. The network nevertheless seems to learn to settle to fixed-points rather than falling into limit cycles or chaotic dynamics. The reason for this phenomenon is not well understood.

$$\Delta w_{ij} = \frac{\eta}{\beta}\left(\rho(s_i^+)\rho(s_j^+) - \rho(s_i^-)\rho(s_j^-)\right) \tag{4}$$

$$\Delta b_i = \frac{\eta}{\beta}\left(\rho(s_i^+) - \rho(s_i^-)\right) \tag{5}$$

Intuitively, the algorithm works by adjusting θ to pull $\arg\min_s E_\theta(s,x)$ closer to $\arg\min_s E_\theta^\beta(s,x,y)$ so that the network will gradually learn to naturally minimize the output loss.

Because inference involves an iterative settling process, it is an undesirably slow process in Equilibrium propagation. In their experiments, Scellier & Bengio (2017) indicate that the number of settling steps required scales super-linearly with the number of layers. This points to an obvious need for a fast inference scheme.

2 METHOD

We propose training a feedforward network $f_\phi(x) \to s^f \in \mathbb{R}^{|\mathcal{S}|}$ to predict the fixed-point of the equilibrating network. This allows the feedforward network to achieve two things: First, it initializes the state of the equilibrating network, so that the settling process starts in the right regime. Second, the feedforward network can be used to perform inference at test-time, since it learns to approximate the minimal-energy state of the equilibrating network, which corresponds to the prediction. $f_\phi(x)$ is defined as follows:

$$\begin{aligned} f_\phi(x) &:= (f_{\phi_j}(x) : j \in \mathcal{S}) && \in \mathbb{R}^{|\mathcal{S}|} \\ f_{\phi_j}(x) &:= \begin{cases} x_j & \text{if } j \in \mathcal{I} \\ h\left(\left(\sum_{i\in\alpha_j^f} \omega_{ij} f_{\phi_i}(x)\right) + c_j\right) & \text{if } j \in \mathcal{S} \end{cases} && \in \mathbb{R} \end{aligned} \tag{6}$$

Where h is some nonlinearity; $\alpha_j^f = (i : (i \in \alpha_j) \wedge (i < j))$ is the set of *feedforward* connections to neuron j (which is a subset of α_j - the full set of connections to neuron j from the equilibrium network from Equation 1); $\phi_j := (\omega_{\alpha_j,j} \in \mathbb{R}^{|\alpha_j^f|}, c_j \in \mathbb{R})$ is the set of parameters connected to neuron j. This feedforward network produces the initial state of the negative phase of equilibrium propagation network, given the input data - i.e., instead of starting at a zero-state, the equilibrium-propagation network is initialized in a state $s^f := f_\phi(x)$. We train the parameters ϕ to approximate the minimal energy state s^- of the equilibrating network. In other words, we seek:

$$\phi^* := \arg\min_\phi \mathcal{L}(f_\phi(x), s^-) \tag{7}$$

$$\mathcal{L}(f_\phi(x), s^-) := \sum_{i\in\mathcal{S}} \mathcal{L}_i(f_{\phi_i}(x), s_i^-) := \sum_{i\in\mathcal{S}} (f_{\phi_i}(x) - s_i^-)^2 \tag{8}$$

The derivative of the forward parameters of the i'th neuron, ϕ_i, can be expanded as:

$$\frac{\partial \mathcal{L}}{\partial \phi_i} := \sum_{i\in\mathcal{S}} \frac{\partial \mathcal{L}_i(f_{\phi_i}(x), s_i^-)}{\partial \phi_i} = \overbrace{\frac{\partial \mathcal{L}_i}{\partial s_i^f}\frac{\partial s_i^f}{\partial \phi_i}}^{local} + \overbrace{\sum_{j>i} \frac{\partial \mathcal{L}_j}{\partial s_j^f}\frac{\partial s_j^f}{\partial s_i^f}\frac{\partial s_i^f}{\partial \phi_i}}^{distant} \tag{9}$$

The *distant* term is problematic, because computing $\frac{\partial s_j^f}{\partial s_i^f}$ would require backpropagation, and the entire purpose of this exercise is to train a neural network without backpropagation. However, we find that only optimizing the local term $\frac{\partial \mathcal{L}_i}{\partial \phi_i}$ does not noticeably harm performance. In Section 2.2 we go into more detail on why it appears to be sufficient to minimize local losses.

Over the course of training, parameters ϕ will learn until our feedforward network is a good predictor of the minimal-energy state of the equilibrating network. This feedforward network can then be used

to do inference: we simply take the state of the output neurons to be our prediction of the target data. The full training procedure is outlined in Algorithm 1. At the end of training, inference can be done either by taking the output activations from the forward pass of the inference network f_ϕ (Algorithm 2), or by initializing with a forward pass and then iteratively minimizing the energy (Algorithm 3). Experiments in Section 3 indicate that the forward pass performs just as well as the full energy minimization.

Algorithm 1 Training

Input: Dataset (x, y), Step Size ϵ, Learning Rate η, Network Architecture α, Number of negative-phase steps T^-, Number of positive-phase steps T^+
$\phi \leftarrow$ InitializeFeedforwardParameters(α)
$\theta \leftarrow$ InitializeEquilibriumParameters(α)
while not converged **do**
  $x_m, y_m \rightarrow$ SampleMinibatch(x, y)
  $s \leftarrow s^f \leftarrow f_\phi(x_m)$
  for $t \in 1..T^-$ **do** # Neg. Phase
    $s \leftarrow s - \epsilon \frac{\partial E_\theta(s, x_m)}{\partial s}$
  $s^- \leftarrow s$
  for $t \in 1..T^+$ **do** # Pos. Phase
    $s \leftarrow s - \epsilon \frac{\partial E_\theta^\beta(s, x_m, y_m)}{\partial s}$
  $s^+ \leftarrow s$
  $\theta \leftarrow \theta - \frac{\eta}{\beta}\left(\frac{\partial E_\theta(s^+, x)}{\partial \theta} - \frac{\partial E_\theta(s^-, x)}{\partial \theta}\right)$
  $\phi_i \leftarrow \phi_i - \eta \frac{\partial \mathcal{L}_i(s_i^f, s_i^-)}{\partial \phi_i} \forall i$
Return: ϕ, θ # Parameters

Algorithm 2 Feedforward Inference

Input: Input Data x, Inference Parameters ϕ
$s \leftarrow f_\phi(x)$
return $(s_i : i \in \mathcal{O})$ # Output unit states

Algorithm 3 Iterative Inference

Input: Input Data x, Initialization Parameters ϕ, Equilibriating Parameters θ, Number of Negative Steps T^-
$s \leftarrow f_\phi(x)$
for $t \in 1..T^-$ **do** # Neg. Phase
  $s \leftarrow s - \epsilon \frac{\partial E_\theta(s, x_m)}{\partial s}$
return $(s_i : i \in \mathcal{O})$ # Output unit states

2.1 Including the forward states in the energy function

Because the fixed point of a neuron of the equilibrating network is a nonlinear function of all the parameters of the network, it is possible for the equilibrating network to create targets that are not *achievable* by the neurons in the feedforward network. Neurons in the feedforward network simply learn a linear mapping from the previous layer's activations to the targets provided by the equilibrating network. In order to encourage the equilibrating network to stay in the regime that is reachable by the forward network, we can add a loss encouraging the fixed points to stay in the regime of the forward pass.

$$E_\theta^\lambda(s, x) = E_\theta(s, x) + \lambda \sum_{j \in \mathcal{S}} (s_j^f - s_j)^2 \tag{10}$$

Where λ is a hyperparameter which brings the fixed-points of the equilibrating network closer to the states of the forward pass. This trick has a secondary benefit: It allows faster convergence in the negative phase by pulling the minimum of $E_\theta^\lambda(s, x)$ closer to the feedforward prediction, so we can learn with fewer convergence steps.

2.2 Why the local loss is sufficient: Gradient Alignment

In Equation 9 we decompose the loss-gradient of parameters ϕ into a local and a global component. Empirically (see Appendix B), we find that using the local loss and simply ignoring the global loss led to equally good convergence.

To understand why this is the case, let use consider a problem where we learn the mapping from an input x to a set of neuron-wise targets: s^*. Assume these targets are generated by some (unknown)

set of ideal parameters ϕ^*, so that $s^* = f_{\phi^*}(x)$. To illustrate, we consider a two layer network with $\phi = (w_1, w_2)$ and $\phi^* = (w_1^*, w_2^*)$:

$$\begin{aligned} s_1 &= h(x \cdot w_1) \quad & s_1^* &= h(x \cdot w_1^*) \quad & \mathcal{L}_1 &= \|s_1 - s_1^*\| \\ s_2 &= h(s_1 \cdot w_2) \quad & s_2^* &= h(s_1^* \cdot w_2^*) \quad & \mathcal{L}_2 &= \|s_2 - s_2^*\| \end{aligned} \tag{11}$$

It may come as a surprise that, given random parameter initializations, the term $\left\langle \frac{\partial \mathcal{L}_1}{\partial w_1}, \frac{\partial \mathcal{L}_2}{\partial w_1} \right\rangle > 0$ is almost always true (where $\langle \cdot, \cdot \rangle$ denotes an inner product), i.e. the local and distant gradients tend to be aligned. This is a pleasant surprise because it means the local loss will at least initially guide us in the right direction. The reason becomes apparent when we define $\Delta w := w - w^*$, and write out the expression for the gradient in the limit of $\Delta w \to 0$

$$\begin{aligned} \frac{\partial \mathcal{L}_1}{\partial w_1}^T &\underset{\Delta w \to 0}{=} (x \cdot \Delta w_1 \odot f'(x \cdot w_1) \odot f'(x \cdot w_1))^T \cdot x \\ \frac{\partial \mathcal{L}_2}{\partial w_1}^T &\underset{\Delta w \to 0}{=} \overbrace{\Big(x \cdot \Delta w_1 \odot f'(x \cdot w_1) \cdot w_2 \odot f'(s_1 \cdot w_2)^2 \cdot w_2^T \odot f'(x \cdot w_1)\Big)^T \cdot x}^{G_1} \\ & \qquad \overbrace{-\Big(s_1 \cdot \Delta w_2 \odot f'(s_1 \cdot w_2)^2 \cdot w_2^T \odot f'(x \cdot w_1)\Big)^T \cdot x}^{G_2} \end{aligned} \tag{12}$$

When the term $w_2 \odot f'(s_1 \cdot w_2)^2 \cdot w_2^T$ is proportional to an identity matrix, we can see that $\frac{\partial \mathcal{L}_1}{\partial w_1}$ and G_1 are perfectly aligned. This will be the case when w_2 is orthogonal and layer 2 has a linear activation. However, even for randomly sampled parameters, $w_2 \odot f'(s_1 \cdot w_2)^2 \cdot w_2^T$ has a strong diagonal component and the terms thus tend to be positively aligned. Figure 1 demonstrates this effect.

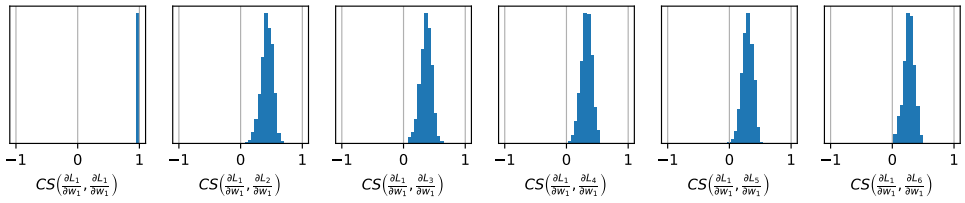

Figure 1: In a randomly initialized 6-layer network, with randomly generated targets s^*, we observe that the local parameter gradient tends to align to the distant components of the parameter gradient. Here, the $i'th$ panel shows the histogram of gradient alignments between local loss $\mathcal{L}_1$ and the loss of layer i, $\mathcal{L}_i$, w.r.t. the first layer's parameters w_1, over random network initializations. $CS(a, b)$ is the cosine similarity between vectors a and b.

This explains the empirical observation in Appendix 2.2 that optimizing the local, as opposed to the global, loss for the feedforward network does not appear to slow down convergence: Layer layers do not have to "wait" for earlier layers to converge before they themselves converge - earlier layers optimize the loss of later layers right from the beginning of training.

3 Experiments

We base our experiments off of those of Scellier & Bengio (2017): We use $\rho(x) = max(0, min(1, x))$ as our nonlinearity. We clip the state of s_i to the range (0, 1) because, since $\rho'(x) = 0 : x < 0 \vee x > 1$, if the system in Equation 2 were run in continuous time, it should never reach states outside this range. Like Scellier & Bengio (2017), to avoid instability problems arising from incomplete negative-phase convergence, we randomly sample $\beta \sim \mathcal{U}(\{-\beta_{base}, +\beta_{base}\})$, where β_{base} is a small positive number, for each minibatch and use this for both the positive phase

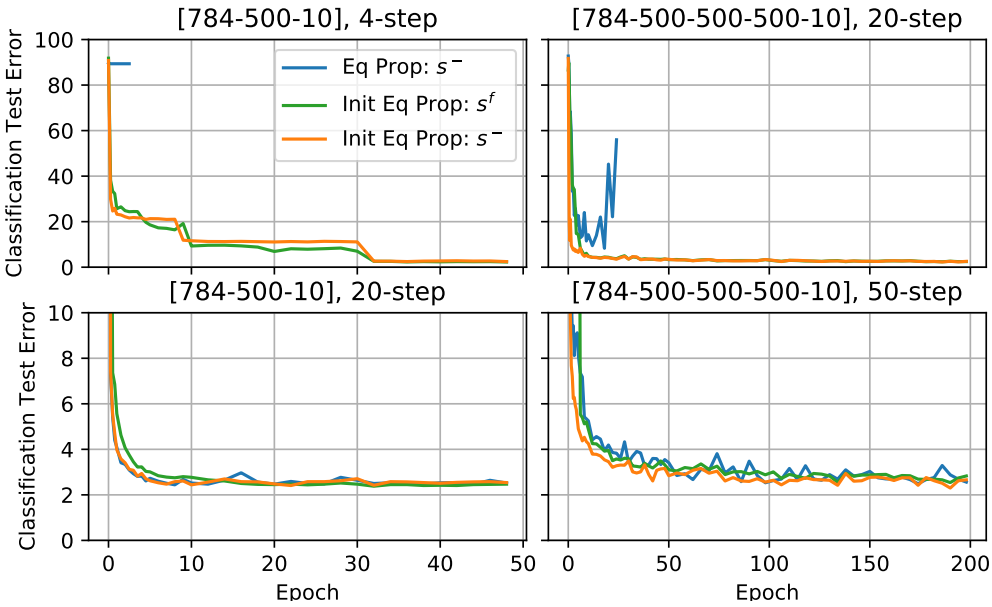

Figure 2: Learning Curves on MNIST comparing the performance of Equilibrium Propagation (Eq Prop: s^-), the Forward-Pass in Initialized Equilibrium Propagation (Fwd Eq Prop: s^f) (Algorithm 2) and the Negative Phase in Initialized Equilibrium Propagation (Fwd Eq Prop: s^-) (Algorithm 3). **Left Column:** A shallow network with a single hidden layer of 500 units. **Right Column:** A deeper network with 3 layers of [500, 500, 500] hidden units. **Top Row:** Training with a small-number of negative-phase steps (4 for the shallow network, 20 for the deeper) shows that feedfoward initialization makes training more stable by providing a good starting point for the negative phase optimization. The *Eq Prop* s^- lines on the upper plots are shortened because we terminate training when the network fails to converge. **Bottom Row:** Training with more negative-phase steps shows that when the baseline Equilibrium Propagation network is given sufficient time to converge, it performs comparably with our feedforward network (Note that the y-axis scale differs from the top).

and for multiplying the learning rate in Equation 3 (for simplicity, this is not shown in Algorithm 1). Unlike Scellier & Bengio (2017), we do not use the trick of caching and reusing converged states for each data point between epochs. Our feedforward network (described in Equation 6) has activation function $h(x) = \rho(x) + 0.01x$, where the 0.01 "leak" is added to prevent the feed-forward neurons from getting "stuck" due to zero-gradients in the saturated regions.

3.1 MNIST

We verify that the our algorithm works on the MNIST dataset. The learning curves can be seen in Figure 2. We find, somewhat surprisingly, that the forward pass of our network performs almost indistinguishably from the performance of the negative-phase of Equilibrium Propagation. This encouraging result shows that this approach for training a feedforward network without backprop does indeed work.

We also see from from the top-two panels of Figure 2 that our approach can stabilize Equilibrium-Prop learning when we run the network for fewer steps than are needed for full convergence. By initializing the negative phase in a close-to-optimal regime, the network is able to learn when the number of steps is so low that plain Equilibrium Propagation cannot converge.

4 RELATED WORK

4.1 RELATION TO ADVERSARIAL LEARNING

Several authors (Kim & Bengio, 2016), (Finn et al., 2016), (Zhai et al., 2016) have pointed out the connection between Energy Based Models and Generative Adversarial Networks (GANs). In these works, a feedforward *generator* network proposes synthetic samples to be evaluated by an energy-based *discriminator*, which learns to push down the energy of real samples and push up the energy of synthetic ones. In these models, both the *generator/sample proposer* and the *discriminator/energy-based-model* are deep feedforward networks trained with backpropagation.

In our approach, we have a similar scenerio. The inference network f_ϕ can be thought of as a conditional generator which produces a network state s^f given a randomly sampled input datum x: $s^f = f_\phi(x)$. Parameters ϕ are trained to approximate the minimal-energy states of the energy function: $\min_\phi \|f_\phi(x) - \arg\min_s E_\theta(s, x)\|$. However, in our model, the Energy-Based network $E_\theta(s, x)$ does not directly evaluate the energy of the generated data s^f, but of the minimizing state $s^- = \arg\min_s E_\theta(s, x)$ which is produced by performing T^- energy-minimization steps on s^f (see Algorithm 1). Like a discriminator, the energy-based model parameters θ learn based on a contrastive loss which pushes up the energy of the of the "synthetic" network state s^- while pushing down the energy of the "real" state s^+.

If we were to set $T^- = 0$ so that $s^- = s^f$ (see Algorithm 1) and train our forward parameters ϕ to minimize $E_\theta(s^f)$, we would have a model that is very similar to a conditional GAN. However doing this naively does not work empirically (the network fails to converge), and the reason for this failure is clear: As stated in Equation 3, the Equilibrium Propagation only accurately computes the output loss gradient $\frac{\partial C(s_\mathcal{O}, y)}{\partial \theta}$ if s is at an minimum of $E_\theta(s, x)$. Although the feed-forward network does attempt to find this minimizing state, we empirically find that this is not close enough to the true minimum to provide for stable learning, and the network fails to converge. This approach - using a recurrent, energy-based model as a discriminator which provides local losses to train a feedforward inference network - seems like an interesting direction of future research.

4.2 RELATION TO DISTILLATION

Hinton et al. (2015) proposed the idea of "distilling" the knowledge of a large neural network or ensemble of neural networks into a smaller network which is designed to run efficiently at inference time. In this work, we use this same idea of distillation to turn a slow, training-time network in to a fast inference network, but in a different way: Our equilibrating network has a slow, iterative inference phase, but is able to provide layerwise targets for a fast feedforward network. We "distill" the knowledge out of the equilibrating network into the feedforward network. In addition, we use our feedforward network to speed the equilibrating network by initializing its inference at a good starting point.

4.3 RELATION TO OTHER WORK IN CREDIT-ASSIGNMENT

The idea of initializing an iterative settling process with a forward pass was proposed by Bengio et al. (2016). In this work, they propose using the parameters of the Equilibriating network to do a forward pass, and describe the conditions under which this provides a good approximation the the energy-minimizing state. Their conclusion is that this criterion is met when consecutive layers of the energy-based model form a good autoencoder. Their work differs from ours in that the parameters of the forward model are tied to the parameters of the energy-based model. The effects of this assumption are unclear, and the authors do not demonstrate a training algorithm using this idea.

Another interesting approach to shortening the inference phase in Equilibrium propagation was proposed by Kohan et al. (2018). The authors propose a model that is *almost* a feedforward network, except that the output layer projects back to the input layer. In the negative phase, the authors make several feedforward passes through the network, reprojecting the output back to the input with each pass. Although the resulting inference model is not a feedforward network, the authors claim that this approach allows them to dramatically shorten convergence time of the negative phase.

More broadly, other approaches to backprop-free credit assignment have been tried. Difference-Target propagation (Lee et al., 2015) proposes a mechanism to send back targets to each layer, such that locally optimizing targets also optimizes the true objective. Feedback-Alignment (Lillicrap et al., 2014) shows that, surprisingly, it is possible to train while using random weights for the backwards pass in backpropagation, because the forward pass parameters tends to "align" to the backwards-pass parameters so that the pseudogradients tend to be within 90° of the true gradients. A similar phenomenon was observed in Equilibrium Propagation by Scellier et al. (2018), who showed that when one removed the constraint of symmetric weight in Equilibrium propagation, the weights would evolve towards symmetry through training.

5 DISCUSSION

In this paper we describe how to use a recurrent, energy-based model to provide layerwise targets with which to train a feedforward network without backpropagation.

This work helps us understand how the brain might be training fast inference networks. In this view, neurons in the inference network learn to predict local targets, which correspond to the minimal energy states, which are found by the iterative settling of a separate, recurrently connected *equilibrating network*.

An interesting direction for future work would be to more closely connect this to adversarial learning. We can think of the feedforward network as generating "synthetic" network states, conditioned on input data, and the equilibrating network as being a discriminator, which aims to distinguish these "synthetic" states from the "real" energy minima which arise when knowledge of the target is present.

REFERENCES

Yoshua Bengio, Benjamin Scellier, Olexa Bilaniuk, Joao Sacramento, and Walter Senn. Feedforward initialization for fast inference of deep generative networks is biologically plausible. *arXiv preprint arXiv:1606.01651*, 2016.

Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*, 2016.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

John J Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the national academy of sciences*, 81(10):3088–3092, 1984.

Taesup Kim and Yoshua Bengio. Deep directed generative models with energy-based probability estimation. *arXiv preprint arXiv:1606.03439*, 2016.

Adam A Kohan, Edward A Rietman, and Hava T Siegelmann. Error forward-propagation: Reusing feedforward connections to propagate errors in deep learning. *arXiv preprint arXiv:1808.03357*, 2018.

Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio. Difference target propagation. In *Machine Learning and Knowledge Discovery in Databases*, pp. 498–515. Springer, 2015.

Timothy P Lillicrap, Daniel Cownden, Douglas B Tweed, and Colin J Akerman. Random feedback weights support learning in deep neural networks. *arXiv preprint arXiv:1411.0247*, 2014.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Benjamin Scellier and Yoshua Bengio. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11:24, 2017.

Benjamin Scellier, Anirudh Goyal, Jonathan Binas, Thomas Mesnard, and Yoshua Bengio. Extending the framework of equilibrium propagation to general dynamics, 2018. URL `https://openreview.net/forum?id=SJTB5GZCb`.

Shuangfei Zhai, Yu Cheng, Rogerio Feris, and Zhongfei Zhang. Generative adversarial networks as variational training of energy based models. *arXiv preprint arXiv:1611.01799*, 2016.

A GLOSSARY

Here we we have a reference of symbols used in the paper, in (Greek, Latin) alphabetical order.

$\alpha_i \subset \{j : j \in \mathcal{S}, j \neq i\}$: The set of neurons in the Equilibrating Network that connect to neuron i

$\alpha_i^f = \{j : j \in \alpha_i, j < i\}$: The set of neurons in the Feedforward Network connected to neuron i.

$\beta \in \mathbb{R}$: The perturbation factor, which modulates how much the *augmented energy* E_θ^β is affected by the output loss.

$\eta \in \mathbb{R}^+$: The learning rate.

θ: The set of parameters (all w_{ij}'s and b_j's, in the Equilibrating network)

ρ: a neuron nonlinearity. In all experiments it is $\rho(x) = max(0, min(1, x))$

$\phi = (\omega, c)$: The set of parameters (all ω_{ij}'s and c_j's), in the feedforward network

ϕ_j: The set of parameters (all $\omega_{\cdot j}$'s and c_j') belonging to a neuron j

(ω, c): The (weights, biases) of the feedforward network. Collectively called ϕ

$C(s_\mathcal{O}, y) \in \mathbb{R}$: The output loss function, defined on the states of the output units.

$E_\theta(s, x) \in \mathbb{R}$: The energy function of the Equilibrating network (Equation 1) produces a scalar energy given a set of states s and input x

$E_\theta^\beta(s, x, y) = E_\theta(s, x) + \beta \frac{\partial C(s_\mathcal{O}, y)}{\partial s} \in \mathbb{R}$: The *augmented* energy function of the Equilibrating network, when it has been perturbed by a factor β by target data y

$f_\phi(x) \mapsto s^f$: The *initialization* function: A feedforward network which initializes the state of the Equilibrating network.

$\mathcal{I}$: The set of input neurons.

h: The nonlinearity of the Feedforward network.

$\mathcal{L}$: The total loss in the Feedforward network's prediction. Defined in Equation 8

$\mathcal{L}_i$: The local loss on the $i'th$ neuron in the feedforward network. Defined in Equation 8

$\mathcal{O}$: The set of output neurons, a subset of $\mathcal{S}$

$\mathcal{S}$: The set of non-input neurons.

s: The set of neuron states. $s := \{s_i : i \in \mathcal{S}\} \in \mathbb{R}^{\dim(\mathcal{S})}$

$s^- := \arg\min_s E(s, x) \in \mathbb{R}^{\dim(\mathcal{S})}$: The minimizing state of the Energy function.

$s^+ := \arg\min_s E_\theta^\beta(s, x, y) \in \mathbb{R}^{\dim(\mathcal{S})}$ The minimizing state of the augmented energy function.

$s^f := f_\phi(x)z \in \mathbb{R}^{\dim(\mathcal{S})}$: The state output by the feedforward network.

$s_\mathcal{O} \in \mathbb{R}^{\dim(\mathcal{O})}$: the states of the output units

T^-, T^+: Hyperparameters for Equilibrium Prop defining the number of steps of convergence of the negative/positive phase.

w, b: the parameters of the Equilibrating network (collectively called θ)

$x \in \mathbb{R}^{\dim \mathcal{I}}$: The input data

$y \in \mathbb{R}^{\dim \mathcal{O}}$: The target data

B LOCAL VS GLOBAL LOSSES

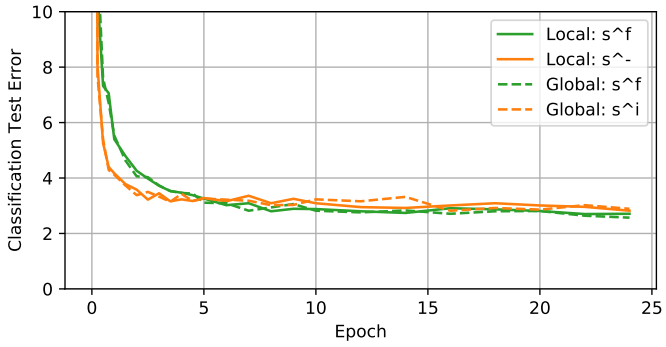

Figure 3: We find that only optimizing the Local Loss term (**solid lines**) performs about as well as when we optimize the global loss term (**dashed lines**) in Equation 9. Green lines show the initialization pass error and Orange lines show the error after the end of the negative phase.