frontiers in Neuroscience

ORIGINAL RESEARCH
published: 18 February 2021
doi: 10.3389/fnins.2021.633674

Scaling Equilibrium Propagation to Deep ConvNets by Drastically Reducing Its Gradient Estimator Bias

Axel Laborieux[1]*, *Maxence Ernoult*[1,2,3]*, *Benjamin Scellier*[3]†, *Yoshua Bengio*[3,4], *Julie Grollier*[2] *and Damien Querlioz*[1]

[1] Université Paris-Saclay, CNRS, Centre de Nanosciences et de Nanotechnologies, Palaiseau, France, [2] Unité Mixte de Physique, CNRS, Thales, Université Paris-Saclay, Palaiseau, France, [3] Mila, Université de Montréal, Montreal, QC, Canada, [4] Canadian Institute for Advanced Research, Toronto, ON, Canada

Equilibrium Propagation is a biologically-inspired algorithm that trains convergent recurrent neural networks with a local learning rule. This approach constitutes a major lead to allow learning-capable neuromophic systems and comes with strong theoretical guarantees. Equilibrium propagation operates in two phases, during which the network is let to evolve freely and then "nudged" toward a target; the weights of the network are then updated based solely on the states of the neurons that they connect. The weight updates of Equilibrium Propagation have been shown mathematically to approach those provided by Backpropagation Through Time (BPTT), the mainstream approach to train recurrent neural networks, when nudging is performed with infinitely small strength. In practice, however, the standard implementation of Equilibrium Propagation does not scale to visual tasks harder than MNIST. In this work, we show that a bias in the gradient estimate of equilibrium propagation, inherent in the use of finite nudging, is responsible for this phenomenon and that canceling it allows training deep convolutional neural networks. We show that this bias can be greatly reduced by using symmetric nudging (a positive nudging and a negative one). We also generalize Equilibrium Propagation to the case of cross-entropy loss (by opposition to squared error). As a result of these advances, we are able to achieve a test error of 11.7% on CIFAR-10, which approaches the one achieved by BPTT and provides a major improvement with respect to the standard Equilibrium Propagation that gives 86% test error. We also apply these techniques to train an architecture with unidirectional forward and backward connections, yielding a 13.2% test error. These results highlight equilibrium propagation as a compelling biologically-plausible approach to compute error gradients in deep neuromorphic systems.

Keywords: equilibrium propagation, energy based models, biologically plausible deep learning, neuromorphic computing, on-chip learning, deep convolutional neural network, learning algorithms

OPEN ACCESS

Edited by:
Melika Payvand,
ETH Zurich, Switzerland

Reviewed by:
Christian Pehle,
Heidelberg University, Germany
Bruno Umbria Pedroni,
University of California, San Diego, United States

****Correspondence:***
Axel Laborieux
axel.laborieux@c2n.upsaclay.fr
Maxence Ernoult
ernoultm@mila.quebec

†Present address:
Benjamin Scellier,
Google, Zurich, Switzerland

Specialty section:
This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience

Received: *25 November 2020*
Accepted: *26 January 2021*
Published: *18 February 2021*

Citation:
Laborieux A, Ernoult M, Scellier B, Bengio Y, Grollier J and Querlioz D (2021) Scaling Equilibrium Propagation to Deep ConvNets by Drastically Reducing Its Gradient Estimator Bias. Front. Neurosci. 15:633674. doi: 10.3389/fnins.2021.633674

1. INTRODUCTION

How synapses in hierarchical neural circuits are adjusted throughout learning a task remains a challenging question called the credit assignment problem (Richards et al., 2019). Equilibrium Propagation (EP) (Scellier and Bengio, 2017) provides a biologically plausible solution to this problem in artificial neural networks. EP is an algorithm for convergent recurrent neural networks

(RNNs) which, by definition, are given a static input and whose recurrent dynamics converge to a steady state corresponding to the prediction of the network. EP proceeds in two phases, bringing the network to a first steady state, then nudging the output layer of the network toward a ground-truth target until reaching a second steady state. During the second phase of EP, the perturbation originating from the output layer propagates forward in time to upstream layers, creating local error signals that match exactly those that are computed by Backpropagation Through Time (BPTT), the canonical approach for training RNNs (Ernoult et al., 2019). We refer to Scellier and Bengio (2019) for a comparison between EP and recurrent backpropagation (Almeida, 1987; Pineda, 1987). Owing to this strong theoretical guarantee, EP can provide leads for understanding biological learning (Lillicrap et al., 2020). Moreover, the spatial locality of the learning rule prescribed by EP and the possibility to make it also local in time (Ernoult et al., 2020) is highly attractive for designing energy-efficient neuromorphic hardware implementations of gradient-based learning algorithms (Ernoult et al., 2020; Foroushani et al., 2020; Ji and Gross, 2020; Kendall et al., 2020; Martin et al., 2020; Zoppo et al., 2020).

To meet these expectations, however, EP should be able to scale to complex tasks. Until now, works on EP (Scellier and Bengio, 2017; O'Connor et al., 2018, 2019; Ernoult et al., 2019, 2020) limited their experiments to the MNIST classification task and shallow network architectures. Despite the theoretical guarantees of EP, the literature suggests that no implementation of EP has thus far succeeded to match the performance of standard deep learning approaches to train deep networks on hard visual tasks. This problem is even more challenging when using a more bio-plausible topology where the synaptic connections of the network are unidirectional: existing proposals of EP in this situation (Scellier et al., 2018; Ernoult et al., 2020) lead to a degradation of accuracy on MNIST compared to standard EP. In this work, we show that performing the second phase of EP with nudging strength of constant sign induces a systematic first order bias in the EP gradient estimate which, once canceled, unlocks the training of deep convolutional neural networks (ConvNets), with bidirectional or unidirectional connections and with performance closely matching that of BPTT on CIFAR-10. We also propose to implement the neural network predictor as an external softmax readout. This modification preserves the local nature of EP and allows us to use the cross-entropy loss, contrary to previous approaches using the squared error loss, and where the predictor takes part in the free dynamics of the system.

Other biologically plausible alternatives to backpropagation (BP) have attempted to scale to hard vision tasks. Bartunov et al. (2018) investigated the use of feedback alignment (Lillicrap et al., 2016) and variants of target propagation (Lecun, 1987; Bengio, 2014) on CIFAR-10 and ImageNet, showing that they perform significantly worse than backpropagation. When the alignment between forward and backward weights is enhanced with extra mechanisms (Akrout et al., 2019), feedback alignment performs better on ImageNet than sign-symmetry (Xiao et al., 2018), where feedback weights are taken to be the sign of the forward weights, and almost as well as backpropagation. However, in feedback alignment and target propagation, the error feedback does not affect the forward neural activity and is instead routed through a distinct backward pathway, an issue that EP avoids. Payeur et al. (2020) proposed a burst-dependent learning rule that also addresses this problem and whose rate-based equivalent, relying on the use of specialized synapses and complex network topology, has been benchmarked against CIFAR-10 and ImageNet. Related works on implicit models (Bai et al., 2019) have shown that training deep networks can be framed as solving a fixed point (steady state) equation, leading to an analytical backward pass. This framework was shown to solve challenging vision tasks (Bai et al., 2020). While the use of a steady state is common with EP, the process to reach the steady state as well as the learning rule are different. In comparison with these approaches, EP offers a minimalistic circuit requirement to handle both inference and gradient computation, which makes it an outstanding candidate for energy-efficient neuromorphic learning hardware design.

More specifically, the contributions of this work are the following:

- We introduce a new method to estimate the gradient of the loss based on three steady states instead of two (section 3.1). This approach enables us to achieve 11.68% test error on CIFAR-10, with 0.6% performance degradation only with respect to BPTT. Conversely, we show that using a nudging strength of constant sign yields 86.64% test error.
- We propose to implement the output layer of the neural network as a softmax readout, which subsequently allows us to optimize the cross-entropy loss function with EP. This method improves the classification performance on CIFAR-10 with respect to the use of the squared error loss and is also closer to the one achieved with BPTT (section 3.2).
- Finally, based on ideas of Scellier et al. (2018) and Kolen and Pollack (1994), we adapt the learning rule of EP for architectures with distinct (unidirectional) forward and backward connections, yielding only 1.5% performance degradation on CIFAR-10 compared to bidirectional connections (section 2.4).

2. BACKGROUND

2.1. Convergent RNNs With Static Input

We consider the setting of supervised learning where we are given an input x (e.g., an image) and want to predict a target y (e.g., the class label of that image). To solve this type of task, Equilibrium Propagation (EP) relies on convergent RNNs, where the input of the RNN at each time step is static and equal to x, and the state s of the neural network converges to a steady-state s_*. EP applies to a wide class of convergent RNNs, where the transition function derives from a scalar primitive[1] Φ (Ernoult et al., 2019). In this situation, the dynamics of a network with parameters θ, usually

[1]In the original version of EP for real-time dynamical systems (Scellier and Bengio, 2017), the dynamics derive from an energy function E, which plays a similar role to the primitive function Φ in the discrete-time setting studied here.

synaptic weights, is given by

$$s_{t+1} = \frac{\partial \Phi}{\partial s}(x, s_t, \theta), \tag{1}$$

where s_t is the state of the RNN at time step t. After the dynamics have converged at some time step T, the network reaches the steady state $s_T = s_*$, which, by definition, satisfies:

$$s_* = \frac{\partial \Phi}{\partial s}(x, s_*, \theta). \tag{2}$$

Formally, the goal of learning is to optimize θ to minimize the loss at the steady state $\mathcal{L}^* = \ell(s_*, y)$, where ℓ is a differentiable cost function. While we did not investigate theoretical guarantees ensuring the convergence of the dynamics, we refer the reader to Scarselli et al. (2008) for sufficient conditions on the transition function to ensure convergence. In practice, we always observe the convergence to a steady-state.

2.2. Training Procedures for Convergent RNNs

2.2.1. Equilibrium Propagation (EP)

Scellier and Bengio (2017) introduced Equilibrium Propagation in the case of real time dynamics. Subsequent work adapted it to discrete-time dynamics, bringing it closer to conventional deep learning (Ernoult et al., 2019). EP consists of two distinct phases. During the first ("free") phase, the RNN evolves according to Equation (1) for T time steps to ensure convergence to a first steady state s_*. During the second ("nudged") phase of EP, a nudging term $-\beta \frac{\partial \ell}{\partial s}$ is added to the dynamics, with β a small scaling factor. Denoting s_0^β, s_1^β, s_2^β... the states during the second phase, the dynamics reads

$$s_0^\beta = s_*, \quad \text{and} \quad \forall t > 0, \quad s_{t+1}^\beta = \frac{\partial \Phi}{\partial s}(x, s_t^\beta, \theta) - \beta \frac{\partial \ell}{\partial s}(s_t^\beta, y). \tag{3}$$

The RNN then reaches a new steady state denoted s_*^β. Scellier and Bengio (2017) proposed the EP learning rule, denoting η the learning rate applied:

$$\Delta\theta = \eta \widehat{\nabla}^{\mathrm{EP}}(\beta), \qquad \text{where}$$
$$\widehat{\nabla}^{\mathrm{EP}}(\beta) \triangleq \frac{1}{\beta}\left(\frac{\partial \Phi}{\partial \theta}(x, s_*^\beta, \theta) - \frac{\partial \Phi}{\partial \theta}(x, s_*, \theta)\right). \tag{4}$$

They proved that this learning rule performs stochastic gradient descent in the limit $\beta \to 0$:

$$\lim_{\beta \to 0} \widehat{\nabla}^{\mathrm{EP}}(\beta) = -\frac{\partial \mathcal{L}^*}{\partial \theta}. \tag{5}$$

2.2.2. Equivalence of Equilibrium Propagation and Backpropagation Through Time (BPTT)

The convergent RNNs considered by EP can also be trained by Backpropagation Through Time (BPTT). At each BPTT training iteration, the first phase is performed for T time steps until the network reaches the steady state $s_T = s_*$. The loss at the final time step is computed and the gradients are subsequently backpropagated through the computational graph of the first phase, backward in time.

Let us denote $\nabla^{\mathrm{BPTT}}(t)$ the gradient computed by BPTT truncated to the last t time steps $(T - t, \ldots, T)$, which we define formally in **Supplementary Material** (section 1).

A theorem derived by Ernoult et al. (2019), inspired from Scellier and Bengio (2019), shows that, provided convergence in the first phase has been reached after $T - K$ time steps (i.e., $s_{T-K} = s_{T-K+1} = \ldots = s_T = s_*$), the gradients of EP match those computed by BPTT in the limit $\beta \to 0$, in the first K time steps of the second phase for fully connected and convolutional architectures including pooling operations:

$$\forall t = 1, 2, \ldots, K,$$
$$\widehat{\nabla}^{\mathrm{EP}}(\beta, t) \triangleq \frac{1}{\beta}\left(\frac{\partial \Phi}{\partial \theta}(x, s_t^\beta, \theta) - \frac{\partial \Phi}{\partial \theta}(x, s_*, \theta)\right) \xrightarrow[\beta \to 0]{} \nabla^{\mathrm{BPTT}}(t). \tag{6}$$

2.3. Convolutional Architectures for Convergent RNNs

A convolutional architecture for convergent RNNs with static input was introduced by Ernoult et al. (2019) and successfully trained with EP on the MNIST dataset. In this architecture, presented in **Figure 1**, we define N^{conv} and N^{fc} the number of convolutional and fully connected layers respectively, and $N^{\mathrm{tot}} \triangleq N^{\mathrm{conv}} + N^{\mathrm{fc}}$. w_{n+1} denotes the weights connecting s^n to s^{n+1}, with $s_0 = x$. To simplify notations, we use distinct operators to differentiate whether w_n is a convolutional layer or a fully connected layer: respectively $\star$ for convolutions and $\cdot$ for linear layers. The primitive function can therefore be defined as:

$$\Phi(x, \{s^n\}) = \sum_{n=0}^{N^{\mathrm{conv}}-1} s^{n+1} \bullet \mathcal{P}\left(w_{n+1} \star s^n\right) + \sum_{n=N^{\mathrm{conv}}}^{N^{\mathrm{tot}}-1} s^{n+1\top} \cdot w_{n+1} \cdot s^n, \tag{7}$$

where $\bullet$ is the Euclidean scalar product generalized to pairs of tensors with same arbitrary dimension, and $\mathcal{P}$ is a pooling operation. Combining Equations (1) and (7), and restricting the space of the state variables to $[0, 1]$, yield the dynamics:

$$\begin{cases} s_{t+1}^n = \sigma\left(\mathcal{P}\left(w_n \star s_t^{n-1}\right) + \tilde{w}_{n+1} \star \mathcal{P}^{-1}\left(s_t^{n+1}\right)\right), 1 \le n \le N^{\mathrm{conv}} \\ s_{t+1}^n = \sigma\left(w_n \cdot s_t^{n-1} + w_{n+1}^\top \cdot s_t^{n+1}\right), N^{\mathrm{conv}} < n < N^{\mathrm{tot}} \end{cases} \tag{8}$$

where σ is an activation function bounded between 0 and 1. Transpose convolution and inverse pooling are respectively defined through the convolution by the flipped kernel $\tilde{w}$ and $\mathcal{P}^{-1}$. Plugging Equation (7) into Equation (4) yields the local learning rule $\Delta\theta_{ij} = \eta(s_{i,*}^\beta s_{j,*}^\beta - s_{i,*} s_{j,*})/\beta$ for a parameter θ_{ij} linking neurons i and j. **Supplementary Material** (section 4) provides the implementation details of this model.

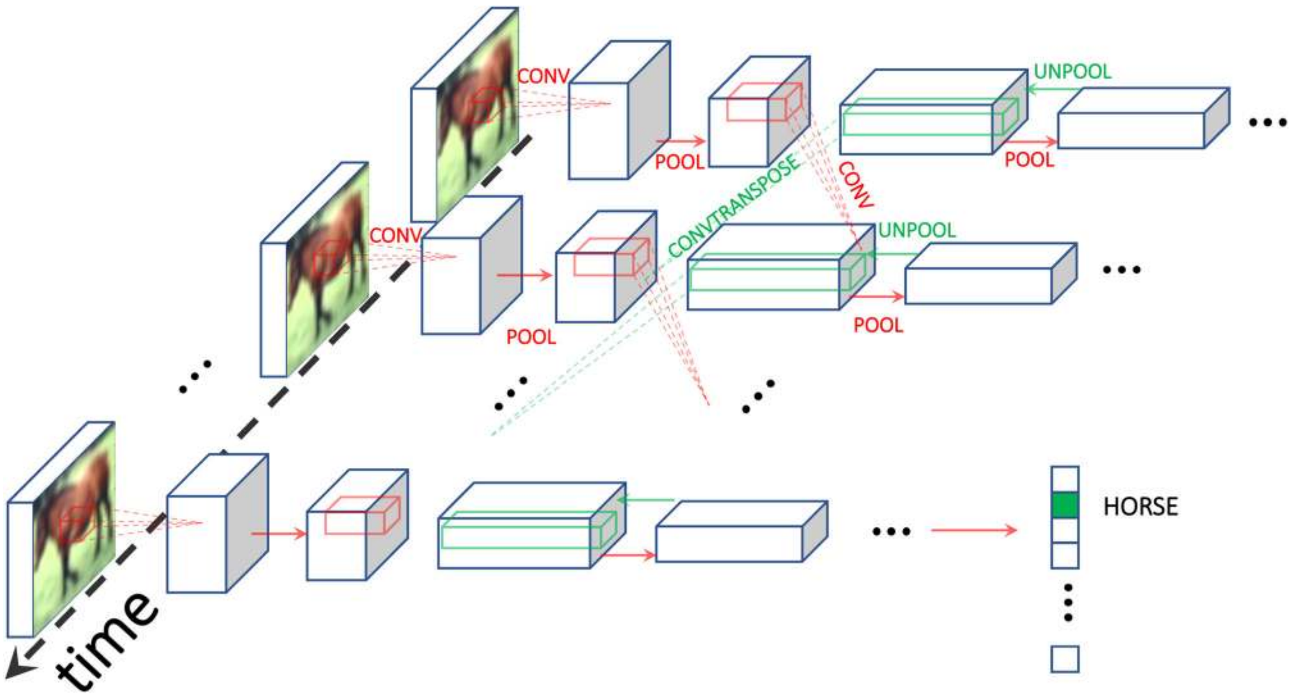

FIGURE 1 | Schematic of the architecture used. We use Equilibrium Propagation (EP) to train a recurrent ConvNet receiving a static input. Red (resp. green) arrows depict forward (resp. backward) operations, with convolutions and transpose convolutions happening through time. At the final time step, the class prediction is carried out. The use of RNNs is inherent in the credit assignment of EP which uses of the temporal variations of the system as error signals for the gradient computation.

2.4. Equilibrium Propagation With Unidirectional Synaptic Connections

We have seen that in the standard formulation of EP, the dynamics of the neural network derive from a function Φ (Equation 1) called the primitive function. This formulation implies the existence of bidirectional synaptic connections between neurons. For better biological plausibility, a more general formulation of EP circumvents this requirement and allows training networks with distinct (unidirectional) forward and backward connections (Scellier et al., 2018; Ernoult et al., 2020). This feature is also desirable for hardware implementations of EP. Although some analog implementations of EP naturally lead to symmetric weights (Kendall et al., 2020), neural networks with unidirectional weights are in general easier to implement in neuromorphic hardware.

In this setting, the dynamics of Equation (1) is changed into the more general form:

$$s_{t+1} = F(x, s_t, \theta), \tag{9}$$

and the conventionally proposed learning rule reads:

$$\Delta\theta = \eta \widehat{\nabla}^{\mathrm{VF}}(\beta), \qquad \text{where}$$
$$\widehat{\nabla}^{\mathrm{VF}}(\beta) \triangleq \frac{1}{\beta}\frac{\partial F}{\partial \theta}(x, s_*, \theta)^\top \cdot \left(s_*^\beta - s_*\right), \tag{10}$$

where VF stands for Vector Field (Scellier et al., 2018). If the transition function F derives from a primitive function Φ (i.e., if $F = \frac{\partial \Phi}{\partial s}$), then $\widehat{\nabla}^{\mathrm{VF}}(\beta)$ is equal to $\widehat{\nabla}^{\mathrm{EP}}(\beta)$ in the limit $\beta \to 0$ (i.e., $\lim_{\beta\to 0} \widehat{\nabla}^{\mathrm{VF}}(\beta) = \lim_{\beta \to 0} \widehat{\nabla}^{\mathrm{EP}}(\beta)$).

3. IMPROVING EP TRAINING

We have seen in Equation (6) that the temporal variations of the network over the second phase of EP exactly compute BPTT gradients in the limit $\beta \to 0$. This result appears to underpin the use of two phases as a fundamental element of EP, but is it really the case? In this section, we revisit EP as a gradient estimation procedure and propose an implementation in three phases instead of two. Moreover, we show how to optimize the cross-entropy loss function with EP. Combining these two new techniques enabled us to achieve the best performance on CIFAR-10 by EP, on architectures with bidirectional and unidirectional forward and backward connections (section 4).

3.1. Reducing Bias and Variance in the Gradient Estimate of the Loss Function

In the foundational work on EP, Scellier and Bengio (2017) demonstrate that:

$$\left.\frac{d}{d\beta}\right|_{\beta=0} \frac{\partial \Phi}{\partial \theta}(x, s_*^\beta, \theta) = -\frac{\partial \mathcal{L}^*}{\partial \theta}. \tag{11}$$

The traditional implementation of EP evaluates the left-hand side of Equation (11) using the estimate $\widehat{\nabla}^{\mathrm{EP}}(\beta)$ with two points $\beta = 0$ and $\beta > 0$, thereby calling for the need of two

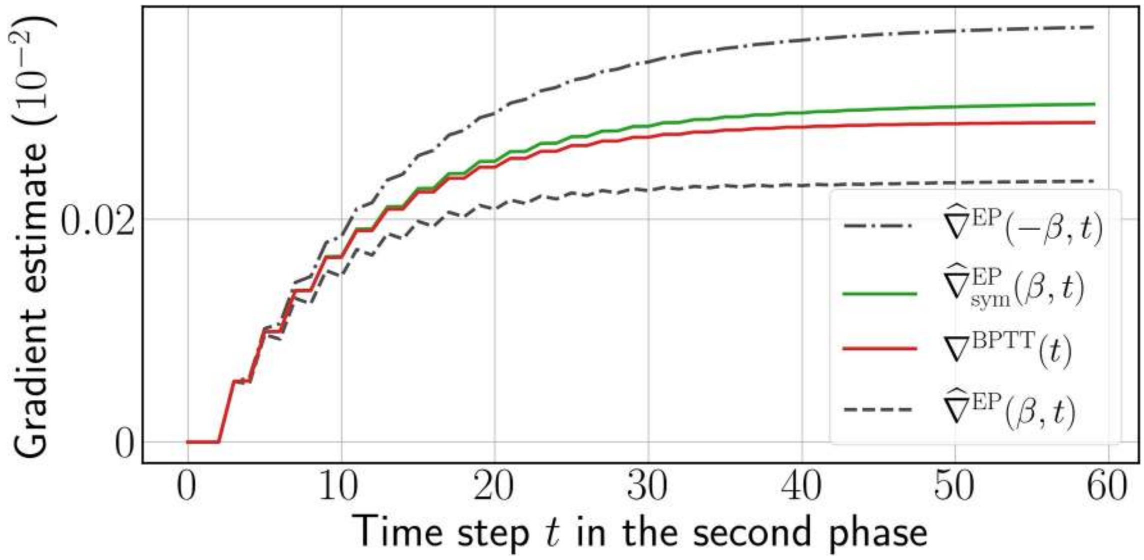

FIGURE 2 | One-sided EP gradient estimate for opposite values of $\beta = 0.1$ (black dashed curves), symmetric EP gradient estimate (green curve), and reference gradients computed by BPTT (red curve) computed over the second phase, for a single weight chosen at random. The time step t is defined for BPTT and EP according to Equation (6). More instances can be found in **Supplementary Material** (section 6).

phases—the free phase and the nudged phase. However, the use of $\beta > 0$ in practice induces a systematic first order bias in the gradient estimation provided by EP. In order to eliminate this bias, we propose to perform a third phase with $-\beta$ as the nudging factor, keeping the first and second phases unchanged. We then estimate the gradient of the loss using the following symmetric difference estimate:

$$\widehat{\nabla}^{\mathrm{EP}}_{\mathrm{sym}}(\beta) \triangleq \frac{1}{2\beta}\left(\frac{\partial \Phi}{\partial \theta}(x, s_*^{\beta}, \theta) - \frac{\partial \Phi}{\partial \theta}(x, s_*^{-\beta}, \theta)\right). \tag{12}$$

Indeed, under mild assumptions on the function $\beta \mapsto \frac{\partial \Phi}{\partial \theta}(x, s_*^{\beta}, \theta)$, we can show that, as $\beta \to 0$:

$$\frac{\widehat{\nabla}^{\mathrm{EP}}(\beta) + \widehat{\nabla}^{\mathrm{EP}}(-\beta)}{2} = -\frac{\partial \mathcal{L}^*}{\partial \theta} + O(\beta^2), \tag{13}$$

$$\widehat{\nabla}^{\mathrm{EP}}_{\mathrm{sym}}(\beta) = -\frac{\partial \mathcal{L}^*}{\partial \theta} + O(\beta^2). \tag{14}$$

This result is proved in Lemma 2 of the **Supplementary Material** (section 2). Equation (13) shows that the estimate $\widehat{\nabla}^{\mathrm{EP}}(\beta)$ possesses a first-order error term in β which the symmetric estimate $\widehat{\nabla}^{\mathrm{EP}}_{\mathrm{sym}}(\beta)$ eliminates (Equation 14). Note that the first-order term of $\widehat{\nabla}^{\mathrm{EP}}(\beta)$ could also be canceled out on average by choosing the sign of β at random with even probability (so that $\mathbb{E}(\beta) = 0$, see Algorithm 1 of the **Supplementary Material**, section 3.1). Although not explicitly stated in this purpose, the use of such randomization has been reported in some earlier publications on the MNIST task (Scellier and Bengio, 2017; Ernoult et al., 2020). However, in this work, we show that this method exhibits high variance in the training procedure.

We call $\widehat{\nabla}^{\mathrm{EP}}(\beta)$ and $\widehat{\nabla}^{\mathrm{EP}}_{\mathrm{sym}}(\beta)$ the one-sided and symmetric EP gradient estimates, respectively. The qualitative difference between these estimates is depicted in **Figure 2**, and the full training procedure is depicted in Algorithm 2 of the **Supplementary Material** (section 3.2).

Finally, this technique can also be applied to the Vector Field setting introduced in section 2.4 and we denote $\widehat{\nabla}^{\mathrm{VF}}_{\mathrm{sym}}(\beta)$ the resulting symmetric estimate—see the **Supplementary Material** (section 4.3) for details.

3.2. Changing the Loss Function

We also introduce a novel architecture to optimize the cross-entropy loss with EP, narrowing the gap with conventional deep learning architectures for classification tasks. In the next paragraph, we denote $\widehat{y}$ the set of neurons that carries out the prediction of the neural network.

3.2.1. Squared Error Loss Function

Previous implementations of EP used the squared error loss. Using this loss function for EP is natural, as in this setting, the output $\widehat{y}$ is viewed as a part of s (the state variable of the network), which can influence the state of the network through bidirectional synaptic connections (see **Figure 3**). Moreover, the nudging term in this case can be physically interpreted since it reads as an elastic force. The state of the network is of the form $s = (s^1, \ldots, s^N, \widehat{y})$ where $h = (s^1, \ldots, s^N)$ represent the "hidden layers," and the corresponding cost function is

$$\ell(\widehat{y}, y) = \frac{1}{2}\left\|\widehat{y} - y\right\|^2. \tag{15}$$

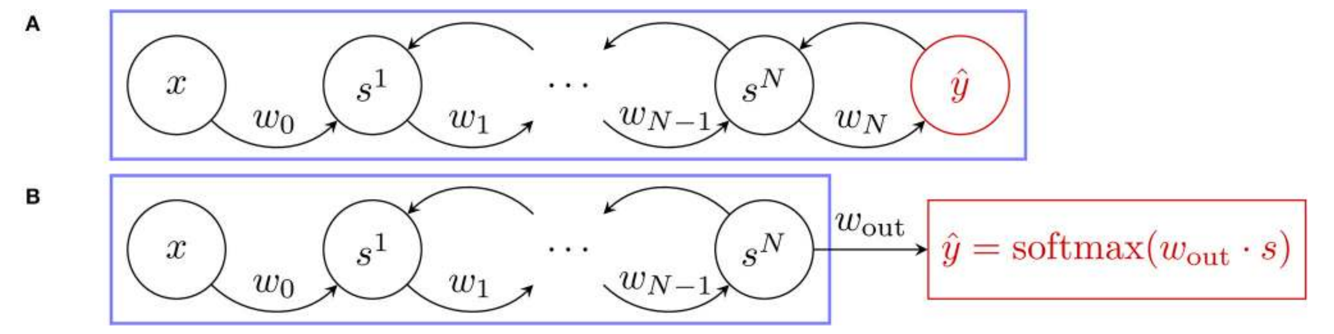

FIGURE 3 | Free dynamics of the architectures used for the two loss functions where the blue frame delimits the system. **(A)** Squared Error loss function. The usual setting where the predictor $\hat{y}$ (in red) takes part in the free dynamics of the neural network through bidirectional synaptic connections. **(B)** Cross-entropy loss function. The new approach proposed in this work where the predictor $\hat{y}$ (also in red) is no longer involved in the system free dynamics and is implemented as a softmax readout.

The second phase dynamics of the hidden state and output layer given by Equation (3) read, in this context:

$$
\begin{aligned}
h^\beta_{t+1} &= \frac{\partial \Phi}{\partial h}(x, h^\beta_t, \widehat{y}^\beta_t, \theta), \\
\widehat{y}^\beta_{t+1} &= \frac{\partial \Phi}{\partial \widehat{y}}(x, h^\beta_t, \widehat{y}^\beta_t, \theta) + \beta\, (y - \widehat{y}^\beta_t).
\end{aligned}
\tag{16}
$$

3.2.2. Softmax Readout, Cross-Entropy Loss Function

In this paper, we propose an alternative approach, where the output $\widehat{y}$ is not a part of the state variable s but is instead implemented as a read-out (see **Figure 3**), which is a function of s and of a weight matrix w_{out} of size $\dim(y) \times \dim(s)$. In practice, w_{out} reads out the last convolutional layer. At each time step t we define:

$$
\widehat{y}_t = \text{softmax}(w_{\text{out}} \cdot s_t). \tag{17}
$$

The cross-entropy cost function associated with the softmax readout is then:

$$
\ell(s, y, w_{\text{out}}) = -\sum_{c=1}^{C} y_c \log(\text{softmax}_c(w_{\text{out}} \cdot s)). \tag{18}
$$

Using $\frac{\partial \ell}{\partial s}(s, y, w_{\text{out}}) = w_{\text{out}}^\top \cdot \left(\text{softmax}(w_{\text{out}} \cdot s) - y\right)$, the second phase dynamics given by Equation (3) read in this context:

$$
s^\beta_{t+1} = \frac{\partial \Phi}{\partial s}(x, s^\beta_t, \theta) + \beta\, w_{\text{out}}^\top \cdot \left(y - \widehat{y}^\beta_t\right). \tag{19}
$$

Note here that the loss $\mathcal{L}^* = \ell(s_*, y, w_{\text{out}})$ also depends on the parameter w_{out}. The **Supplementary Material** (section 4.2.2) provides the learning rule applied to w_{out}.

3.3. Changing the Learning Rule of EP With Unidirectional Synaptic Connections

In the case of architectures with unidirectional connections, applying the traditional EP learning rule directly, as given by Equation (10), prescribes different forward and backward weights updates, resulting in significantly different forward and backward weights throughout learning. However, the theoretical equivalence between EP and BPTT only holds for bidirectional connections. Until now, training experiments of unidirectional weights EP have performed worse than bidirectional weights EP (Ernoult et al., 2020). In this work, therefore, we tailor a new learning rule for unidirectional weights, described in detail the **Supplementary Material** (section 4.3), where the forward and backward weights undergo the same weight updates, incorporating an equal leakage term. This way, forward and backward weights, although they are independently initialized, naturally converge to identical values throughout the learning process. A similar methodology, adapted from Kolen and Pollack (1994), has been shown to improve the performance of Feedback Alignment in Deep ConvNets (Akrout et al., 2019).

Assuming general dynamics of the form of Equation (9), we distinguish forward connections θ_{f} from backward connections θ_{b} so that $\theta = \{\theta_{\text{f}}, \theta_{\text{b}}\}$, with θ_{f} and θ_{b} having same dimension. Assuming a first phase, a second phase with $\beta > 0$ and a third phase with $-\beta$, we define:

$$
\forall \text{i} \in \{\text{f}, \text{b}\},
$$
$$
\overline{\nabla}^{\text{VF}}_{\theta_{\text{i}}}(\beta) = \frac{1}{2\beta}\left(\frac{\partial F}{\partial \theta_{\text{i}}}^\top(x, s^\beta_*, \theta) \cdot s^\beta_* - \frac{\partial F}{\partial \theta_{\text{i}}}^\top(x, s^{-\beta}_*, \theta) \cdot s^{-\beta}_*\right) \tag{20}
$$

and we propose the following update rules:

$$
\begin{cases}
\Delta\theta_{\text{f}} = \eta\left(\widehat{\nabla}^{\text{KP-VF}}_{\text{sym}}(\beta) - \lambda\theta_{\text{f}}\right) \\
\Delta\theta_{\text{b}} = \eta\left(\widehat{\nabla}^{\text{KP-VF}}_{\text{sym}}(\beta) - \lambda\theta_{\text{b}}\right)
\end{cases},
$$
$$
\text{with} \quad \widehat{\nabla}^{\text{KP-VF}}_{\text{sym}}(\beta) = \frac{1}{2}(\overline{\nabla}^{\text{VF}}_{\theta_{\text{f}}}(\beta) + \overline{\nabla}^{\text{VF}}_{\theta_{\text{b}}}(\beta)) \tag{21}
$$

where η is the learning rate and λ a leakage parameter. The estimate $\widehat{\nabla}^{\text{KP-VF}}_{\text{sym}}(\beta)$ can be thought of a generalization of Equation (12), as highlighted in the **Supplementary Material** (section 4.3) with an explicit application of Equation (21) to a ConvNet. In the case of a fully connected layer, both terms in

TABLE 1 | Hyper-parameters used for the CIFAR-10 experiments.

Hyper-parameter	Squared error	Cross-entropy
T	250	250
K	30	25
β	0.5	1.0
Batch size	128	128
Initial learning rates (Layer-wise)	0.25 - 0.15 - 0.1 - 0.08 - 0.05	0.25 - 0.15 - 0.1 - 0.08 - 0.05
Final learning rates	10^{-5}	10^{-5}
Weight decay (All layers)	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$
Momentum	0.9	0.9
Epoch	120	120
Cosine annealing Decay time (epochs)	100	100

TABLE 2 | Performance comparison on CIFAR-10 between BPTT and EP with several gradient estimation schemes.

Loss function	EP gradient	EP error (%)		BPTT error (%)	
	estimate	Test	Train	Test	Train
	2-Phase/$\hat{\nabla}^{\mathrm{EP}}$	86.64 (5.82)	84.90		
Squared error	Random Sign	21.55 (20.00)	20.01	11.10 (0.21)	3.69
	3-Phase/$\hat{\nabla}^{\mathrm{EP}}_{\mathrm{sym}}$	12.45 (0.18)	7.83		
Cross-Ent.	3-Phase/$\hat{\nabla}^{\mathrm{EP}}_{\mathrm{sym}}$	**11.68** (**0.17**)	**4.98**	11.12 (0.21)	2.19
Cross-Ent. (Dropout)	3-Phase/$\hat{\nabla}^{\mathrm{EP}}_{\mathrm{sym}}$	11.87 (0.29)	6.46	10.72 (0.06)	2.99
Cross-Ent.	3-Phase/$\hat{\nabla}^{\mathrm{VF}}_{\mathrm{sym}}$	75.47 (4.72)	78.04	9.46 (0.17)	0.80
	3-Phase/$\hat{\nabla}^{\mathrm{KP-VF}}_{\mathrm{sym}}$	**13.15** (**0.49**)	8.87		

Note that the different gradient estimates only apply to EP. We indicate over five trials the mean and standard deviation in parenthesis for the test error, and the mean train error.

the sum in the right hand side of Equation (21) are equal: $\partial F/\partial \theta_i$ only depends on the neuron activations and not on θ_i, in the same way, as seen at the end of section 2.3, that Equation (8) yields a fully local learning rule. The case of convolutional layers is a little more subtle, due to presence of the maximum pooling operations. The forward weights are involved in a pooling operation while the backward weights are involved in an unpooling operation. However, for the parameter update to be the same, the pooling and unpooling operations need to share information regarding the indices of maxima. Therefore, there is indeed a need for information transfer between backward and forward parameters, but this exchange is limited to the index of the maximum identified in the maximum pooling operation (this can be seen from Equation 24).

4. RESULTS

In this section, we implement EP with the modifications described in section 3 and successfully train deep ConvNets on the CIFAR-10 vision task (Krizhevsky et al., 2009). The convolutional architecture used consists of four 3×3 convolutional layers of respective feature maps 128–256–512–512. We use a stride of one for each convolutional layer, and zero-padding of one for each layer except for the last layer. Each layer is followed by a 2×2 Max Pooling operation with a stride of two. The resulting flattened feature vector is of size 512. The weights are initialized using the default initialization of PyTorch, which is the uniform Kaiming initialization of He et al. (2015). The data is normalized and augmented with random horizontal flips and random crops. The training is performed with stochastic gradient descent with momentum and weight decay. We use the learning rate scheduler introduced by Loshchilov and Hutter (2016) to speed up convergence.

The hyper-parameters are reported in **Table 1**. All experiments are performed using PyTorch 1.4.0. (Paszke et al., 2017). The simulations were carried across several servers consisting of 14 Nvidia GeForce RTX 2080 TI GPUs in total. Each run was performed on a single GPU for an average run time of 2 days.

4.1. ConvNets With Bidirectional Connections

We first consider the bidirectional weight setting of section 2.3. In **Table 2**, we compare the performance achieved by the ConvNet for each EP gradient estimate introduced in section 3.1 with the performance achieved by BPTT.

The one-sided gradient estimate leads to unstable training behavior where the network is unable to fit the data, as shown by the purple curve of **Figure 4A**, with 86.64% test error on CIFAR-10. When the bias in the gradient estimate is averaged out by choosing at random the sign of β during the second phase, the average test error over five runs goes down to 21.55% (see **Table 2**). However, one run among the five yielded instability similar to the one-sided estimate, whereas the four remaining runs lead to 12.61% test error and 8.64% train error. This method for estimating the loss gradient thus presents high variance—further experiments shown in the **Supplementary Material** (section 4.4) confirm this trend.

Conversely, the three-phase symmetric estimate enables EP to consistently reach 12.45% test error, with only 1.35% degradation with respect to BPTT (see **Figure 4A**). Therefore, removing the first-order error term in the gradient estimate is critical for scaling to deeper architectures. Proceeding to this end deterministically (with three phases) rather than stochastically (with a randomized nudging sign) appears more reliable.

The results of **Table 2** also show that the readout scheme introduced in section 3.2 to optimize the cross-entropy loss function enables EP to narrow the performance gap with BPTT down to 0.56% while outperforming the Squared Error setting by 0.77%. However, we observe that the test errors reached by BPTT are similar for the squared error and the cross-entropy loss. The fact that only EP benefits from the cross-entropy loss is due to the output not being part of the dynamics, which reduces the number of layers following the dynamics by one.

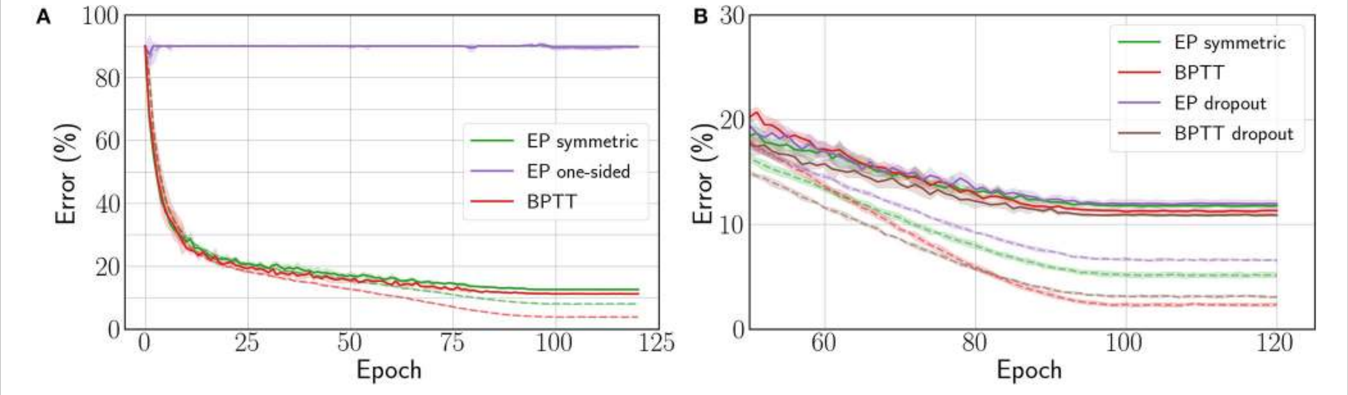

FIGURE 4 | (A) Train (dashed) and test (solid) errors on CIFAR-10 with the Squared Error loss function. **(B)** Train (dashed) and test (solid) errors on CIFAR-10 with the Cross-Entropy loss function. The curves are averaged over 5 runs and shadows stand for $\pm 1 \times$ standard deviation. The change in error rate around epochs 85–90 is due to the end of the learning rate scheduler decay phase (Cosine annealing).

We also adapted dropout (Srivastava et al., 2014) to convergent RNNs (see the **Supplementary Material**, section 4.5 for implementation details) to see if the performance could be improved further. However, we can observe from **Table 2** and **Figure 4B** that contrary to BPTT, the EP test error is not improved by adding a 0.1 dropout probability in the neuron layer after the convolutions.

4.2. ConvNets With Unidirectional Connections

We now present the accuracy achieved by EP when the architecture uses distinct forward and backward weights, using a softmax readout. For this architecture, the backward weights are defined for all convolutional layers, except the first convolutional layer connected to the static input. The forward and backward weights are initialized randomly and independently at the beginning of training. The backward weights have no bias contrary to their forward counterparts. The hyper-parameters such as learning rate, weight decay and momentum are shared between forward and backward weights.

As seen in **Table 2**, we find that the estimate $\widehat{\nabla}^{\mathrm{VF}}_{\mathrm{sym}}(\beta)$ leads to a poor performance with 75.47% test-error. We concomitantly observed that forward and backward weight did not align well, as shown by the dashed curves in **Figure 5**. Conversely, when using our new estimate $\widehat{\nabla}^{\mathrm{KP-VF}}_{\mathrm{sym}}(\beta)$ defined in section 3.3, a good performance is recovered with only 1.5% performance degradation with respect to the architecture with bidirectional connections, and a 3% degradation with respect to BPTT (see **Table 2**). The discrepancy between the BPTT test error achieved by the architecture with bidirectional (11.12%) and unidirectional (9.46%) connections comes from the increase in parameters provided by backward weights. As observed in the weight alignment curves in **Figure 5**, forward and backward weights are well-aligned by epoch 50 when using the new estimate. These results suggest that enhancing forward and backward weights alignment can help EP training in deep ConvNets.

5. DISCUSSION

Our results unveil the necessity, in order to scale EP to deep convolutional neural networks on hard visual tasks, to compute better gradient estimates than the conventional implementation of EP. This traditional implementation incorporates a first order gradient estimate bias, which severely impedes the training of deep architectures. Conversely, we saw that the three-phase EP proposed here removes this bias and brings EP performance on CIFAR-10 close to the one achieved by BPTT. Additionally, our new technique to train EP with softmax readout reduces the gap between EP and BPTT further down to 0.56%, while maintaining the locality of the learning rule of all parameters.

While the test accuracy of BPTT and our adapted EP are very close, we can notice in **Table 2** that BPTT fits the training data better than EP by at least 2.8%. Also, the introduction of dropout improves BPTT performance, while it has no significant effect on the test accuracy of EP. These two insights combined suggest that EP training may have a self-regularizing effect applied throughout the network, similar to the effects of dropout. We hypothesize this effect to be not only due to the residual estimation bias of the BPTT gradients by EP, but also to an additional inherent error term due to the fact that in practice, the fixed point is approached with a precision that depends on the number of time steps at inference. While the exactness of the fixed point is crucial for EP, BPTT computes exact gradients regardless of whether the fixed point is not exactly reached.

We also saw that employing a new training technique that still preserves the spatial locality of EP computations—and therefore its suitability for neuromorphic implementations—our results extend to the case of an architecture with distinct forward and backward synaptic connections.

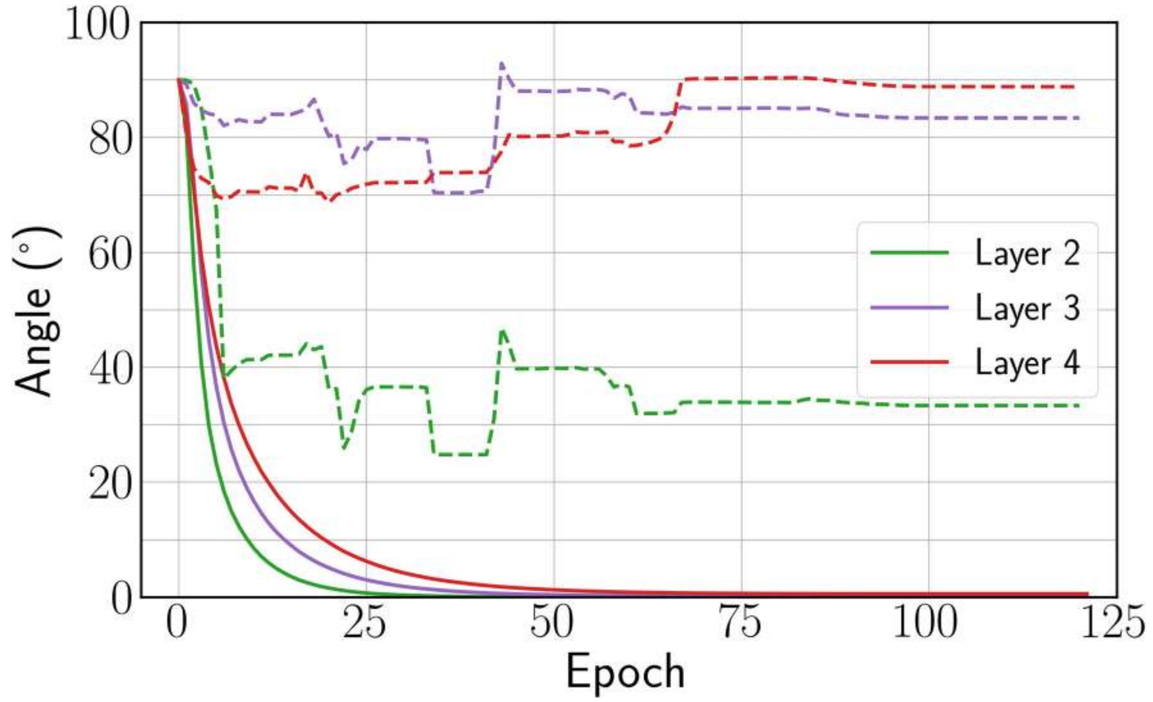

FIGURE 5 | Angle between forward and backward weights for the new estimate $\widehat{\nabla}_{\text{sym}}^{\text{KP-VF}}$ introduced (solid) and $\widehat{\nabla}_{\text{sym}}^{\text{VF}}$ (dashed). The angle is not defined for the first layer because the input layer is clamped.

We only observe a 1.5% performance degradation with respect to the bidirectional architecture. This result demonstrates the scalability of EP without the biologically implausible requirement of a bidirectional connectivity pattern.

Our three steady states-based gradient estimate comes at a computational cost with regards to the conventional EP implementation, as an additional phase is needed. Even though the steady state of the free phase s_* is not used to compute the gradient estimate in Equation (12), we experimentally found that s_* is needed as a starting point for the second and third phases. In terms of simulation time, EP is 20% slower than BPTT due to the dynamics performed in second and third phases. However, the memory requirement to store the computational graph unfolded in time in the case of BPTT far outweighs the memory needed by EP, which consists only of the steady states reached by the neurons.

The full potential of EP will be best envisioned on neuromorphic hardware. Multiple works have investigated the implementation of EP on such systems (Ernoult et al., 2019, 2020; Foroushani et al., 2020; Ji and Gross, 2020; Zoppo et al., 2020), in both rate based (Kendall et al., 2020) and spiking approaches (Martin et al., 2020). Most of these approaches employ analog circuits that exploit device physics to implement the dynamics of EP intrinsically. The spatially local nature of EP computations, on top of its connection with physical equations, make this mapping between EP and neuromorphic hardware natural. Our prescription to run two nudging phases with opposite nudging strengths could be implemented naturally in neuromorphic systems. In fact, the use of differential operation to cancel inherent biases is a technique widely used in electronics, and in neuromorphic computing in particular (Hirtzlin et al., 2019). Overall, our work provides evidence that EP is a compelling approach to scale neuromorphic on-chip training to real-world tasks in a fully local fashion.

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are publicly available. This data can be found here: https://github.com/Laborieux-Axel/Equilibrium-Propagation.

AUTHOR CONTRIBUTIONS

AL developed the PyTorch code for the project and performed the simulations. ME supervised the work, helped debug the code, guided hyperparameter search, and designed the experiments with unidirectional connections. BS proposed the ideas of unbiasing the gradient estimate and of using a softmax readout. DQ, JG, and YB provided additional guidance and support. All authors participated in data analysis, discussed the results, and co-edited the manuscript.

FUNDING

This work was supported by European Research Council Starting Grant NANOINFER (reference: 715872), European Research Council Grant bioSPINspired (reference: 682955), CIFAR, NSERC, and Samsung.

ACKNOWLEDGMENTS

The authors would like to thank Thomas Fischbacher for useful feedback and discussions.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fnins.2021.633674/full#supplementary-material

REFERENCES

Akrout, M., Wilson, C., Humphreys, P., Lillicrap, T., and Tweed, D. B. (2019). "Deep learning without weight transport," in *Advances in Neural Information Processing Systems* (Vancouver, BC), 974–982.

Almeida, L. B. (1987). "A learning rule for asynchronous perceptrons with feedback in a combinatorial environment," in *Proceedings of the IEEE First International Conference on Neural Networks (San Diego, CA), Vol. II* (Piscataway, NJ: IEEE), 609–618.

Bai, S., Kolter, J. Z., and Koltun, V. (2019). "Deep equilibrium models," in *Advances in Neural Information Processing Systems* (Vancouver, BC), 690–701.

Bai, S., Koltun, V., and Kolter, J. Z. (2020). Multiscale deep equilibrium models. *arXiv preprint arXiv:2006.08656*.

Bartunov, S., Santoro, A., Richards, B., Marris, L., Hinton, G. E., and Lillicrap, T. (2018). "Assessing the scalability of biologically-motivated deep learning algorithms and architectures," in *Advances in Neural Information Processing Systems* (Vancouver, BC), 9368–9378.

Bengio, Y. (2014). How auto-encoders could provide credit assignment in deep networks via target propagation. *arXiv preprint arXiv:1407.7906*.

Ernoult, M., Grollier, J., Querlioz, D., Bengio, Y., and Scellier, B. (2019). "Updates of equilibrium prop match gradients of backprop through time in an RNN with static input," in *Advances in Neural Information Processing Systems* (Vancouver, BC), 7081–7091.

Ernoult, M., Grollier, J., Querlioz, D., Bengio, Y., and Scellier, B. (2020). Equilibrium propagation with continual weight updates. *arXiv preprint arXiv:2005.04168*.

Foroushani, A. N., Assaf, H., Noshahr, F. H., Savaria, Y., and Sawan, M. (2020). "Analog circuits to accelerate the relaxation process in the equilibrium propagation algorithm," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)* (Séville), 1–5. doi: 10.1109/ISCAS45731.2020.9181250

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* (Santiago), 1026–1034. doi: 10.1109/ICCV.2015.123

Hirtzlin, T., Bocquet, M., Penkovsky, B., Klein, J.-O., Nowak, E., Vianello, E., et al. (2019). Digital biologically plausible implementation of binarized neural networks with differential hafnium oxide resistive memory arrays. *Front. Neurosci*. 13:1383. doi: 10.3389/fnins.2019.01383

Ji, Z., and Gross, W. (2020). "Towards efficient on-chip learning using equilibrium propagation," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)* (Séville), 1–5. doi: 10.1109/ISCAS45731.2020.9180548

Kendall, J., Pantone, R., Manickavasagam, K., Bengio, Y., and Scellier, B. (2020). Training end-to-end analog neural networks with equilibrium propagation. *arXiv preprint arXiv:2006.01981*.

Kolen, J. F., and Pollack, J. B. (1994). "Backpropagation without weight transport," in *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94), Vol. 3* (Orlando, FL), 1375–1380. doi: 10.1109/ICNN.1994.374486

Krizhevsky, A., Hinton, G., et al. (2009). *Learning Multiple Layers of Features From Tiny Images*. Available online at: https://www.semanticscholar.org/paper/Learning-Multiple-Layers-of-Features-from-Tiny-Krizhevsky/5d90f06bb70a0a3dced62413346235c02b1aa086

Lecun, Y. (1987). *Modeles connexionnistes de l'apprentissage (connectionist learning models)* (Ph.D. thesis). IAAI Laboratory, Paris, France.

Lillicrap, T. P., Cownden, D., Tweed, D. B., and Akerman, C. J. (2016). Random synaptic feedback weights support error backpropagation for deep learning. *Nat. Commun*. 7, 1–10. doi: 10.1038/ncomms13276

Lillicrap, T. P., Santoro, A., Marris, L., Akerman, C. J., and Hinton, G. (2020). Backpropagation and the brain. *Nat. Rev. Neurosci*. 21, 335–346. doi: 10.1038/s41583-020-0277-3

Loshchilov, I., and Hutter, F. (2016). Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*.

Martin, E., Ernoult, M., Laydevant, J., Li, S., Querlioz, D., Petrisor, T., and Grollier, J. (2020). Eqspike: spike-driven equilibrium propagation for neuromorphic implementations. *arXiv preprint arXiv:2010.07859*.

O'Connor, P., Gavves, E., and Welling, M. (2018). "Initialized equilibrium propagation for backprop-free training" in *International Conference on Learning Representations 2019*.

O'Connor, P., Gavves, E., and Welling, M. (2019). "Training a spiking neural network with equilibrium propagation," in *The 22nd International Conference on Artificial Intelligence and Statistics* (Montreal, QC), 1516–1523.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., et al. (2017). "Automatic differentiation in pytorch," in *NeurIPS 2017 Workshop Autodiff Decision Program*.

Payeur, A., Guerguiev, J., Zenke, F., Richards, B., and Naud, R. (2020). Burst-dependent synaptic plasticity can coordinate learning in hierarchical circuits. *bioRxiv [Preprint]*. doi: 10.1101/2020.03.30.015511

Pineda, F. J. (1987). Generalization of back-propagation to recurrent neural networks. *Phys. Rev. Lett*. 59, 2229–2232. doi: 10.1103/PhysRevLett.59.2229

Richards, B. A., Lillicrap, T. P., Beaudoin, P., Bengio, Y., Bogacz, R., Christensen, A., et al. (2019). A deep learning framework for neuroscience. *Nat. Neurosci*. 22, 1761–1770. doi: 10.1038/s41593-019-0520-2

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2008). The graph neural network model. *IEEE Trans. Neural Netw*. 20, 61–80. doi: 10.1109/TNN.2008.2005605

Scellier, B., and Bengio, Y. (2017). Equilibrium propagation: bridging the gap between energy-based models and backpropagation. *Front. Comput. Neurosci*. 11:24. doi: 10.3389/fncom.2017.00024

Scellier, B., and Bengio, Y. (2019). Equivalence of equilibrium propagation and recurrent backpropagation. *Neural Comput*. 31, 312–329. doi: 10.1162/neco_a_01160

Scellier, B., Goyal, A., Binas, J., Mesnard, T., and Bengio, Y. (2018). Generalization of equilibrium propagation to vector field dynamics. *arXiv preprint arXiv:1808.04873*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res*. 15, 1929–1958.

Xiao, W., Chen, H., Liao, Q., and Poggio, T. (2018). Biologically-plausible learning algorithms can scale to large datasets. *arXiv preprint arXiv:1811.03567*.

Zoppo, G., Marrone, F., and Corinto, F. (2020). Equilibrium propagation for memristor-based recurrent neural networks. *Front. Neurosci.* 14:240. doi: 10.3389/fnins.2020.00240

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.