Dual Propagation: Accelerating Contrastive Hebbian Learning with Dyadic Neurons

Rasmus Høier [1] **D. Staudt** [1] **Christopher Zach** [1]

Abstract

Activity difference based learning algorithms—such as contrastive Hebbian learning and equilibrium propagation—have been proposed as biologically plausible alternatives to error back-propagation. However, on traditional digital chips these algorithms suffer from having to solve a costly inference problem twice, making these approaches more than two orders of magnitude slower than back-propagation. In the analog realm equilibrium propagation may be promising for fast and energy efficient learning, but states still need to be inferred and stored twice. Inspired by lifted neural networks and compartmental neuron models we propose a simple energy based compartmental neuron model, termed dual propagation, in which each neuron is a dyad with two intrinsic states. At inference time these intrinsic states encode the error/activity duality through their difference and their mean respectively. The advantage of this method is that only a single inference phase is needed and that inference can be solved in layerwise closed-form. Experimentally we show on common computer vision datasets, including Imagenet32x32, that dual propagation performs equivalently to back-propagation both in terms of accuracy and runtime.

1. Introduction

In spite of the massive success of the error back-propagation (BP) method for training deep neural networks, there are several theoretical and practical reasons to consider alternatives. One theoretical reason is the question of biological plausibility, which was already raised in (Crick, 1989) soon after the publication of the influential back-propagation paper (Rumelhart et al., 1986). A significant practical challenge is the energy consumption of using back-propagation

[1]Chalmers University of Technology, Sweden. Correspondence to: Rasmus Høier <hier@chalmers.se>.

In *ICML Workshop on Localized Learning (LLW)*, Honolulu, Hawaii, USA. 2023. Copyright 2023 by the author(s).

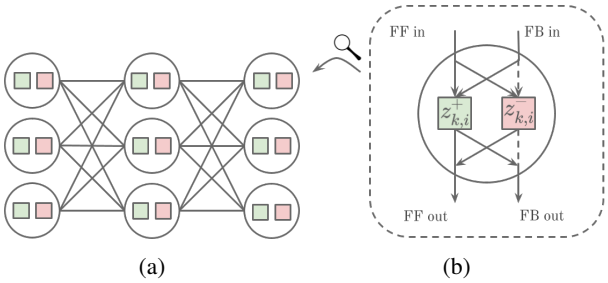


Figure 1: (a) A fully connected network with compartmental neurons. (b) Close-up of neuron i in layer k. Solid arrows indicate identity connections and dashed arrows indicate sign inversion. The neuron possesses positively and negatively nudged states $z^+_{k,i}$ and $z^-_{k,i}$ and propagates two types of signals. The mean $\frac{1}{2}(z^+_{k,i} + z^-_{k,i})$ is propagated upstream to layer $k+1$ while the difference $\frac{1}{2}(z^+_{k,i} - z^-_{k,i})$ is propagated downstream to layer $k-1$.

to train deep neural networks (Strubell et al., 2019), which can be largely attributed to floating point processing but also to the required fine-grained synchronization of the individual steps—therefore preventing the utilization of e.g. energy efficient analog circuits (Yi et al., 2022). Consequently, taking inspiration from biological motivated learning algorithms may contribute to making deep neural networks substantially more energy efficient. Research along these lines has converged on the idea of encoding weight updates in terms of neuronal activity differences, Neural Gradient Representation by Activity Differences (NGRAD) (Lillicrap et al., 2020). NGRAD methods differ in terms of what is meant by activity difference. Some approaches use the difference in neuron activity at different times, while others compute the difference between different sub-states within individual neurons.

Contrastive Hebbian learning (Movellan, 1991; Xie & Seung, 2003) and its modern incarnation (Scellier & Bengio, 2017) fall into the NGRAD category of methods and replace the fine-grained synchronization in BP with globally synchronizing two inference phases. This allows continuous and asynchronous transmission of neural activity signals e.g. in (partially) analog computing devices (e.g. (O'Connor

Table 1: Comparison of selected biologically motivated algorithms and back-propagation.

Global synchron.	Yes			No		
Method	BP	CHL	EP	DCMC	LPOM	DP
States	Activity/error	Clamped/free	Nudged/Free	Activity/error	Clamped/Free	+/- nudged
Inference	Closed-form	Iterative	Iterative	Iterative	Iterative	Layerwise closed-form
Steps	2	(10–100)	~ 300	~ 1000	(10–100)	≥ 2

et al., 2019; Zoppo et al., 2020; Kendall et al., 2020)). Lifted neural networks (such as (Carreira-Perpinan & Wang, 2014; Zhang & Brand, 2017; Li et al., 2020)) only require a single, largely asynchronously operating phase, but lack the simplicity in computing the neural activations that is found in back-propagation or in Hopfield nets. This might be the factor preventing lifted networks—to our knowledge—being implemented in neuromorphic or analog devices.

In this paper we propose a single phase contrastive Hebbian learning-type algorithm, which we refer to as dual propagation (DP). Neural units in our framework maintain internally two states (a "dyad") and are therefore an instance of a compartment based neuron model. The resulting weight update rules are fully local, and the inference rules have layerwise closed-form solutions. Experimentally we explore the effects of different neuron state update schemes on MNIST, including one which updates layers of neurons in a random sequence as well as a resource efficient version which can be efficiently implemented on top of existing auto-differentiation frameworks. Furthermore, the resource efficient version is benchmarked on CIFAR10, CIFAR100 and Imagenet32x32, where it performs equivalently to back-propagation both in terms of accuracy and computational runtime.

Table 1 lists a selection of NGRAD algorithms as well as back-propagation grouped into two groups based on whether they require global synchronization. The number of steps denotes how many state updates per neural unit are required. This information was not available for CHL and LPOM, hence we provide estimates based on our experience. For EP and DCMC the number of steps was based on numbers reported in (Laborieux et al., 2021; Laborieux & Zenke, 2022) and (Sacramento et al., 2018) respectively. These numbers are dependent on network architecture, so they should be considered indicative only.

2. Related Work

Contrastive Hebbian learning and equilibrium propagation Contrastive Hebbian learning was originally proposed for Hopfield networks with continuous units (Movellan, 1991), but has since been applied to layered networks as well (O'Reilly, 1996; Xie & Seung, 2003). In this framework inference consists in minimizing a Lyapunov function (or network potential) with respect to activations via suitable iterations. During training this procedure is carried out twice (once with and once without clamping the output units to the target label), yielding so-called clamped and free steady states. Learning amounts to minimizing the difference between the Lyapunov function for these two states. The necessary global synchronization to initiate the two phases is somewhat problematic for a biologically motivated algorithm, as biological neural networks operate in a streaming context

More recently, equilibrium propagation (Scellier & Bengio, 2017; Scellier et al., 2018) has been proposed as an improvement on CHL. In this framework output neurons are not clamped but instead "nudged" (or soft clamped via a target loss) towards the target labels. Like CHL, equilibrium propagation is computationally costly when implemented on conventional hardware. However, this is not an issue when using equilibrium propagation for training analog neural networks (Kendall et al., 2020; Scellier, 2021).

Lifted networks In the lifted neural network framework, the loss function is augmented by terms which penalize neurons not conforming to a chosen activation function (Askari et al., 2018). Learning proceeds by first minimizing this augmented objective with respect to neuron activations and subsequently with respect to the weights. Different variations of this idea have been explored in (Carreira-Perpinan & Wang, 2014; Zhang & Brand, 2017; Whittington & Bogacz, 2017; Gu et al., 2020; Høier & Zach, 2020; Li et al., 2019; 2020; Zach & Estellers, 2019; Song et al., 2020). Lifted neural network and two-phase CHL are actually two sides of the same coin, as lifted networks implicitly integrate one of the two phases (Zach, 2021).

Compartmental neuron models The segregated dendrite model (Guerguiev et al., 2017) takes inspiration from pyramidal neurons, formulating inference and learning rules in terms of simplified dynamics between apical and basal dendritic compartments and the soma. However, similarly to CHL this model requires global synchronization of two distinct phases in order to compute errors as the temporal difference in apical dendritic activity. An alternative to computing errors in terms of temporal activity differences is to encode errors as the activity difference between distinct neuronal states. This is the approach taken by dendritic

cortical microcircuits (DCMC) (Sacramento et al., 2018). Hence the network does not require a global synchronization between distinct phases. However, this comes at the cost of requiring auxiliary neurons (interneurons). Note that both of these particular methods aim for a higher degree of biological plausibility, by modelling spiking neurons, which also makes them comparatively computationally costly.

Asynchronous DNN training A number of methods aim to improve the parallelism in training DNNs by decoupling the layer dependency. Approaches proposed in the literature include auxililary coordindates ((Carreira-Perpinan & Wang, 2014), an early instance of a lifted network), decoupled training using ADMM (Taylor et al., 2016) and block-coordinate method (Zhang & Brand, 2017), and synthetic gradients (Jaderberg et al., 2017). Usually, these methods are not inspired by biological plausibility, but aim at better utilization of highly paralleized computing resources.

Weight transport While biological neural networks use distinct unidirectional pathways to transport signals to and from neurons artificial, neural networks trained with BP employs weight symmetry (activity is transported by W and errors by $W^\top$). Feedback alignment (FA) (Lillicrap et al., 2014; 2016) and direct feedback alignment (DFA) (Nøkland, 2016)) are variations of back-propagation that aim to remove this symmetry constraint[1]. In (D)FA errors are transported backwards using a distinct set of static randomly initialized weights. For fully connected architectures the learnable forwards weights can be observed to partially align to the static backwards weights, providing useful learning signals. However, deep convolutional networks trained with (D)FA fail to learn efficiently (Bartunov et al., 2018; Moskovitz et al., 2018). Refenetti et al (Refinetti et al., 2021) explains this in terms of the difficulty of having the highly sparse Toeplitz representation of a convolutional layers align to random feedback weights. Difference target propagation (Lee et al., 2015) also aim to address the weight transport problem, but take a different approach. In this framework each layer is modelled as an autoencoder, and distinct feed-back weights are trained to propagate useful targets to hidden layers by approximately inverting the feed-forward mapping. DTP has been explored in a variety flavors (Lee et al., 2015; Bartunov et al., 2018; Meulemans et al., 2020; Ernoult et al., 2022), mainly differing in how feed-back weights are learned.

Although weight transport is not the focus of this paper, we explore a classic approach for training network with asymmetric feedforward and feedback weights. Kolen-Pollack learning (Kolen & Pollack, 1994) of feedback weights amounts to applying the same weight updates to both sets of weights, combined with weight decay regularization. Given enough updates the weights will then converge to the same values. This approach has previously been used to facilitate learning in networks with distinct feedforward and feedback weights (Akrout et al., 2019; Laborieux et al., 2021). Weight mirrors (Akrout et al., 2019) and stochastic approximation to estimate the feedback mapping (Ernoult et al., 2022) are recent contributions to address the weight transport problem. We utilize the Kolen-Pollack scheme in some experiments to asses the general compatibility of our dual propagation method with enhanced biological plausibility.

[1]Occasionally these algorithms are referred to as random back-propagation (Baldi et al., 2018)

Notations For a differentiable and strictly convex function G we denote the regular Bregman divergence by $D_G(z\|y) = G(z) - G(y) - (z-y)^\top \nabla G(y)$ and its reparametrized version by $\bar{D}_G(z\|x) := G(z) - z^\top x + G^*(x)$. $\bar{D}_G(z\|x)$ is non-negative due to the Fenchel-Young inequality, and $\bar{D}_G(z\|x) = 0$ iff $z = \nabla G^*(x) = f(x)$ for a suitable mapping $f = \nabla G^* = (\nabla G)^{-1}$. We therefore have $D_G(z\|g(x)) = \bar{D}_G(z\|x)$. Constraints $x \in C$ are written as $\imath_C(x)$ in their functional form.

3. Contrastive Hebbian Learning with Dyadic Neurons

In this section we present a framework inspired by contrastive Hebbian learning, that is based on positively and negatively nudged internal states maintained for every neuron. Consequently, the proposed framework shares a number of high-level similarities with other lifted neural network approaches, but retains closed-form inference steps to determine the neural activations. Further, under suitable smoothness assumption on the activation non-linearities, the gold-standard BP gradient is approximated to second order.

3.1. The Contrastive Objective

Our model is based on two sets of states for neural activity denoted by z^+ and z^-. Since we focus on layered feedforward DNNs, $z_k^\pm$ denotes the respective activations in layer k, where layer 0 is the input and layer L is the output layer. For a target loss ℓ the main objective in our model driving the update of the network parameters is given by

$$\begin{aligned}
\mathcal{L}_\alpha(\theta) &:= \min_{z^+} \max_{z^-} \alpha\ell(z_L^+) + \bar{\alpha}\ell(z_L^-) \\
&+ \sum_{k=1}^{L} \tfrac{1}{\beta_k}\big(\bar{D}_{G_k}(z_k^+\|W_{k-1}\bar{z}_{k-1}) - \bar{D}_{G_k}(z_k^-\|W_{k-1}\bar{z}_{k-1})\big) \\
&= \min_{z^+} \max_{z^-} \alpha\ell(z_L^+) + \bar{\alpha}\ell(z_L^-) \\
&+ \sum_{k=1}^{L} \tfrac{1}{\beta_k}\big(G_k(z_k^+) - G_k(z_k^-) + (z_k^- - z_k^+)^\top W_{k-1}\bar{z}_{k-1}\big)\,,
\end{aligned} \tag{1}$$

where $\alpha \in [0,1]$, $\bar{\alpha} = 1-\alpha$, $\bar{z}_k := \alpha z_k^+ + \bar{\alpha} z_k^-$. G_k are strictly convex and differentiable functions determining the activation non-linearity at layer k. There are implicit constraints fixing $z_0^+ = z_0^- = x$ to the network's input x, and the target loss ℓ carries the information on the desired prediction (e.g. target label). For notational brevity we omit an explicit indication of bias vectors.

The general motivation for this cost function is that the states z^+ are "nudged" towards reducing the target loss ℓ, and z^- are negatively nudged states increasing the target loss. Both states are regularized via the Bregman divergences in $\mathcal{L}_\alpha$. An important property of (1) is that the problematic term $G_k^*(W_{k-1}z_{k-1})$ in $\bar{D}_{G_k}(z_k, W_{k-1}z_{k-1}$ cancels—at the expense of a min-max inner problem structure. After the activations $z^\pm$ are determined in the inference phase, the network weights $\theta = (W_0, \dots, W_{L-1})$ can be adjusted (see Section 3.2). The objective in (1) is stated for a single training sample (x, ℓ) but is straightforwardly extended over an entire training set.

The choice of G_k determines the induced network non-linearity, e.g. $G_k = \|\cdot\|^2/2$ yields linear units, $G_k = \|\cdot\|^2/2 + \imath_{\geq 0}(\cdot)$ introduces ReLU activation mappings (e.g. (Zhang & Brand, 2017)), and G_k being the negated Shannon entropy adds a soft-max layer (e.g. (Zach, 2021)).

It turns out that only three choices for α are particularly meaningful. Before focusing on the main case $\alpha = 1/2$ in the remainder of this work, we briefly discuss the choices of $\alpha = 1$ and $\alpha = 0$.

The case $\alpha = 1$: In this case it is possible to maximize w.r.t. z^- in closed form, and the objective is given by

$$\mathcal{L}_1(\theta) = \min_{z^+} \ell(z_L^+) + \sum_{k=1}^{L} \tfrac{1}{\beta_k} \bar{D}_{G_k}(z_k^+ \| W_{k-1} z_{k-1}^+), \quad (2)$$

which can be identified essentially as the LPOM objective proposed in (Li et al., 2019), which in the limit $\beta_k \to 0^+$ yields a back-propagation variant based on appropriate directional derivatives (Zach, 2021). The optimal state z^+ minimizes the target loss, but is regularized to stay close to the forward pass prediction $f_k(W_{k-1}z_{k-1}^+)$. One advantage of $\mathcal{L}_1$ over similar lifted network formulations such as MAC (Carreira-Perpinan & Wang, 2014) is, that the inner minimization problem w.r.t. z^+ in $\mathcal{L}_1$ is at least layer-wise convex, and inferring the network activations z_k^+ can be conducted by approximately solving

$$\begin{aligned} \min_{z_k^+} \beta_k \left(G_k(z_k^+) - (z_k^+)^\top W_{k-1} z_{k-1}^+ \right) \\ + \beta_{k+1} \left(G_{k+1}^*(W_k z_k^+) - (z_{k+1}^+)^\top W_k z_k^+ \right). \end{aligned} \quad (3)$$

In general, determining z_k^+ (with all other states fixed) requires an iterative algorithm and cannot be obtained in closed-form.

The case $\alpha = 0$: Similarly, by first using the min-max lemma and by minimizing w.r.t. z^+ in closed form, the resulting objective is an upper bound of

$$\mathcal{L}_0(\theta) \geq \max_{z^-} \ell(z_L^-) - \sum_{k=1}^{L} \tfrac{1}{\beta_k} \bar{D}_{G_k}(z_k^- \| W_{k-1} z_{k-1}^-). \quad (4)$$

The r.h.s. can be interpreted as an "anti-LPOM" objective, and the solution for z^- maximizes ℓ, but is regularized by the second part in the loss. Assuming $\min g(u) = 0$ and $v := \arg\min_u g(u)$, the general relation

$$\min_u f(u) + g(u) \leq f(v) \leq \max_u f(u) - g(u) \quad (5)$$

implies that $\mathcal{L}_0(\theta) \geq \mathcal{L}_1(\theta)$. Loosely speaking, $\mathcal{L}_0$ uses the opposite directional derivatives (in the direction of the increasing loss in contrast to $\mathcal{L}_1$) when $\beta_k \to 0^+$. $\mathcal{L}_0$ shares the block-concave structure of the inner tasks with layer-wise convexity of $\mathcal{L}_1$, but also its difficulty of inference similar to (3). $\mathcal{L}_0$ also puts an upper limit on the choice of β_L in order to prevent an unbounded maximization instance w.r.t. z_L^-.

The general case $\alpha \in (0,1)$ For any choice $\alpha \in (0,1)$ optimization over one set of unknowns (i.e. maximizing out z^- or minimizing out z^+) is not easily possible in $\mathcal{L}_\alpha$. Setting $\alpha \in (0,1)$ is nevertheless formally appealing, as layer-wise inference (Section 3.2) is very efficient and can be conducted in closed-form.

Unlike contrastive Hebbian learning frameworks adapted for layered networks (such as (Xie & Seung, 2003; Zach & Estellers, 2019)), the objective $\mathcal{L}_\alpha$ in (1) does not need layer-wise discounting by using an increasing sequence for β_k satisfying $\beta_k \ll \beta_{k+1}$. Generally, the most important parameter is β_L determining the amount of "nudging" (or soft-clamping) introduced by the target loss ℓ, and setting $\beta_k = \beta_L$ for all $k = 1, \dots, L-1$ is sufficient in practice.

3.2. Inference Rules and Weight Updates

In this section we discuss the inference method to determine the solutions z^+ and z^- of the inner optimization tasks in (1). The method is formally based on block-coordinate descent (BCD) by solving for z_k^+ and z_k^- for a specific layer while keeping all other states $z_{\setminus k}^\pm$ fixed. We emphasize that these steps are inspired by BCD, but due to the min-max structure convergence results for BCD are not readily applicable, and therefore the proposed inference method requires a different analysis (see Section 3.3).

Minimization and maximization over z_k^+ and z_k^-, respectively, in (1) can be carried out simultaneously for an entire

layer, yielding the closed form inference rules

$$\begin{aligned} z_k^+ &\leftarrow f_k\left(W_{k-1}\bar{z}_{k-1} + \tfrac{\alpha\beta_k}{\beta_{k+1}} W_k^\top (z_{k+1}^+ - z_{k+1}^-)\right) \\ z_k^- &\leftarrow f_k\left(W_{k-1}\bar{z}_{k-1} - \tfrac{\bar{\alpha}\beta_k}{\beta_{k+1}} W_k^\top (z_{k+1}^+ - z_{k+1}^-)\right) \end{aligned} \tag{6}$$

for $k = 1, \dots, L-1$. Recall that $z_0^+ = z_0^- = x$ for the network input x, and that $\bar{z}_k := \alpha z_k^+ + \bar{\alpha} z_k^-$ is the possibly weighted average. The assignment of $z_L^\pm$ depends on the target loss ℓ. If the output layer is linear and $\ell(z_L) = \|z_L - y\|^2/2$ is the least-squares loss, then we obtain

$$\begin{aligned} z_L^+ &\leftarrow \tfrac{1}{1+\alpha\beta_L}\left(W_{L-1}\bar{z}_{L-1} + \alpha\beta_L y\right) \\ z_L^- &\leftarrow \tfrac{1}{1-\bar{\alpha}\beta_L}\left(W_{L-1}\bar{z}_{L-1} - \bar{\alpha}\beta_L y\right). \end{aligned} \tag{7}$$

Note that solving for z_L^- is an unbounded convex maximization problem if $\bar{\alpha}\beta_L \geq 1$, and therefore we need to choose $\beta_L < 1/\bar{\alpha} = 1/(1-\alpha)$. If the target loss is linear (or as occurring more commonly, a differentiable target loss is linearized), i.e. $\ell(z) = g^\top z$, then the update for a linear output layer reduces to

$$z_L^+ \leftarrow W_{L-1}\bar{z}_{L-1} - \alpha\beta_L g \quad z_L^- \leftarrow W_{L-1}\bar{z}_{L-1} + \bar{\alpha}\beta_L g. \tag{8}$$

It is worth noting that in the absence of upstream activity (or a constant target loss) the update in (6) basically reduces to a standard feed-forward pass, with $z_k^+ = z_k^-$.

Once the states $z^\pm$ are inferred (or sufficiently close to their optimal solution), the contribution of a sample (x, ℓ) to the overall gradient w.r.t. the trainable parameters $\theta = (W_0, \dots, W_{L-1})$ is given by

$$\tfrac{\partial}{\partial W_{k-1}}\mathcal{L}_\alpha = \tfrac{1}{\beta_k}\left(z_k^- - z_k^+\right)\bar{z}_{k-1}^\top. \tag{9}$$

This is an instance of contrastive Hebbian learning using only information that is local to the artificial synapse.

3.3. Analysis

In our analysis we consider only the case $\alpha = 1/2$, i.e. $\mathcal{L}_{1/2}$, since $\mathcal{L}_1$ is discussed in the literature (Li et al., 2019; 2020; Zach, 2021) (and $\mathcal{L}_0$ is quite related as discussed above), and the validity of the closed-form updates in (6) hinges on the choice $\alpha = 1/2$ as described below in the convergence analysis.

Equivalence to back-propagation in the limit In this paragraph we demonstrate that the first factor in (9), $(z_k^- - z_k^+)/\beta_k$, converges to

$$\tfrac{d}{dz_k}\ell(z_L) \qquad \text{s.t. } z_k = f_k(W_{k-1}z_{k-1}) \tag{10}$$

i.e.

$$\begin{aligned} \tfrac{d}{dz_L}\ell(z_L) &= \ell'(z_L) \\ \tfrac{d}{dz_k}\ell(z_L) &= W_k^\top f_{k+1}'(W_k z_k)\, \tfrac{d}{dz_{k+1}}\ell(z_L) \end{aligned} \tag{11}$$

for $\beta_k \to 0^+$. Consequently, the learning rule in (9) approaches the back-propagation method in the limit.

We use $a_k := W_{k-1}\bar{z}_{k-1}$ and $\Delta_k := \frac{1}{2\beta_k}(z_k^- - z_k^+)$ in the following. Since we are interested in the limit case when $\beta_k \to 0^+$, it is sufficient to employ a linearized target loss, $\ell(z_L) = g^\top z_L$. Using (8) we deduce that $(z_L^- - z_L^+)/\beta_L = g$ and the claim is therefore true for $d\ell(z_L)/dz_L$. For $k < L$ we expand

$$\begin{aligned} &\lim_{\beta_k \to 0^+} \tfrac{z_k^- - z_k^+}{\beta_k} \\ &= \lim_{\beta_k \to 0^+} \tfrac{f(a_k + \beta_k W_k^\top \Delta_{k+1}) - f(a_k - \beta_k W_k^\top \Delta_{k+1})}{\beta_k} \\ &= 2W_k^\top f_{k+1}'(a_k)\Delta_k = W_k^\top f_{k+1}'(a_k)\, \tfrac{z_{k+1}^- - z_{k+1}^+}{\beta_{k+1}}, \end{aligned} \tag{12}$$

but the last factor equals $d\ell(z_L)/dz_{k+1}$ by our induction hypothesis. Hence, in the weak feedback setting (where $\beta_k \approx 0$ for all $k \in \{1, \dots, L\}$), dual propagation approximates back-propagation. This property is not surprising and is also shared with many (contrastive) Hebbian learning frameworks.

This part of the analysis is valid for $\alpha \in [0, 1]$ in general, as deviating from $\alpha = 1/2$ only introduces an asymmetry in the finite differences estimate for the derivative. The induction to show this equivalence requires $z_k^\pm$ to be fixed points of the updates in (6) and (8) to be applicable. Consequently, we focus our attention on the question whether the state updates reach such fixed point at all in the remainder of this section.

Convergence analysis: linear networks We assume $G_k(z_k) = \|z_k\|^2/2$ and a linearized loss, $\ell(z_L) = g^\top z_L$. For brevity we assume all β_k are equal to a common $\beta > 0$. In this setting the updates for z_k^+ and z_k^- reduce to

$$\begin{aligned} z_L^\pm &\leftarrow W_{L-1}\bar{z}_{L-1} \mp \beta g \\ z_k^\pm &\leftarrow W_{k-1}\bar{z}_{k-1} \pm \tfrac{1}{2}W_k^\top (z_{k+1}^+ - z_{k+1}^-) \end{aligned} \tag{13}$$

We reparametrize the updates in terms of

$$\bar{z}_k = \tfrac{1}{2}(z_k^+ + z_k^-) \qquad \delta_k = \tfrac{1}{2}(z_k^+ - z_k^-), \tag{14}$$

i.e., $z_k^+ = \bar{z}_k + \delta_k$ and $z_k^- = \bar{z}_k - \delta_k$. After rearranging terms, the update steps above translate to

$$\bar{z}_k \leftarrow W_{k-1}\bar{z}_{k-1} \quad \delta_L \leftarrow -\beta g \quad \delta_k \leftarrow W_k^\top \delta_{k+1} \tag{15}$$

in terms of the average state $\bar{z}_k$ and error signal δ_k. These steps can be identified as steps in the forward and backward pass of back-propagation in a linear network. A difference to regular back-propagation is that the layers are not traversed in a predefined order, but potentially in an arbitrary sequence. The relevant observation is that $\bar{z}_k$ has the true value (and remains at that state) if the sequence of layer update contains

the ordered sequence $[1:k]$ as subsequence. Analogously, δ_k is assigned (and fixed) to the correct error signal if $[L:k]$ is a subsequence of the traversed layers. Thus, the states $\bar{z}$ and δ (and consequently z^+ and z^-) have their correct values once the sequence of visited states contains one entire forward and backward pass as subsequences. Hence, the condition on the traversal sequence is that $(1,\dots,L,\dots,1)$ appears eventually as a subsequence. This is actually a necessary condition for all types of supervised learning methods for DNNs: the input needs to reach the loss, and the loss needs to be distributed through the entire network.

Convergence analysis: nonlinear networks We use similar ideas as above to analyze the proposed dual propagation method for nonlinear networks. The activations resulting from the pure forward pass are denoted as z^*, i.e. $z_k^* = f_k(W_{k-1}z_{k-1}^*)$. As before, we assume $\beta_k = \beta$ for all k as above for notational brevity and assume a linearized target loss $\ell(z_L) = g^\top z_L$. We use two back-propagated error signals δ^+ and δ^- corresponding to forward and backward finite differences, respectively. The underlying recursion for $\delta_k^\pm$ is $\delta_L^\pm \leftarrow -\beta g$ and

$$\begin{aligned}\delta_k^+ &\leftarrow f_k\left(W_{k-1}z_{k-1}^* + W_k^\top \delta_{k+1}^+\right) - z_k^* \\ \delta_k^- &\leftarrow z_k^* - f_k\left(W_{k-1}z_{k-1}^* - W_k^\top \delta_{k+1}^+\right)\end{aligned} \tag{16}$$

When $\beta \to 0^+$ and differentiable f_k this approaches

$$\tfrac{1}{\beta}\delta_k^\pm \to f_k'(W_{k-1}z_{k-1}^*)W_k^\top \delta_{k+1} \tag{17}$$

(recall that $\delta_{k+1}^\pm$ scales with β via $\delta_L^\pm = -\beta g$). If f_k is not differentiable at its argument, we obtain directional derivatives instead (if they exist). As argued above, any sequence of updates $z_k^* \leftarrow f_k(W_{k-1}z_{k-1}^*)$ and $\delta_k^\pm$ in (16) eventually yields the correct forward activations and error signals (i.e. the finite-difference approximation of the derivative) under mild conditions. After introducing $z_k^\pm = z_k^* \pm \delta_k^\pm$ we deduce the respective updates

$$\begin{aligned}z_k^+ &= z_k^* + \delta_k^+ \leftarrow f(W_{k-1}z_{k-1}^* + W_k^\top \delta_{k+1}^+) \\ z_k^- &= z_k^* - \delta_k^- \leftarrow f(W_{k-1}z_{k-1}^* - W_k^\top \delta_{k+1}^-),\end{aligned} \tag{18}$$

which are almost the updates given in (6). In general we need to maintain 3 quantities (e.g. z_k^+, z_k^- and z_k^*, unless $\delta_k^+ = \delta_k^-$). By observing that

$$\bar{z}_k = \tfrac{1}{2}(z_k^+ + z_k^-) = z_k^* + \tfrac{1}{2}(\delta_k^+ - \delta_k^-) \tag{19}$$

we conclude that $\bar{z}_k$ is a good approximation of z_k^* as long as $\delta_k^+ \approx \delta_k^-$, which is satisfied if the activation function f_k is at least locally approximately linear. Consequently the algorithm replaces z_{k-1}^* with $\bar{z}_{k-1}$ in (18) to maintain only two sets of neural activations (and to be in line with other CHL-based methods), thereby yielding the updates in (6). As shown in Section B, this line of reasoning is applicable only when $\alpha = 1/2$. In toy experiments, choosing α significantly different from $1/2$ leads to at least inferior results (when all β_k are small) or even to divergent behavior of $z^\pm$ when using (6) for, e.g., with the choice of $\beta_k = 1/2$ all k. The case $\alpha = 1/2$ works equally well for a wide range of $\beta_k < 1/(1-\alpha) = 2$.

It is interesting to note that learning the network parameters and the initial motivations for the state updates in (6) are based on (1), but the state updates in (18) (and their approximation in (6)) are best understood as proceeding towards the global optimum of the following objective,

$$\begin{aligned}U(z^*, \delta^\pm) &:= \beta g^\top(\delta_L^+ + \delta_L^-) + \tfrac{1}{2}\|\delta_L^+\|^2 + \tfrac{1}{2}\|\delta_L^-\|^2 \\ &+ \sum \left\|\delta_k^+ - f_k\left(a_k^* + W_k^\top \delta_{k+1}^+\right)\right\|^2 \\ &+ \sum \left\|\delta_k^- - f_k\left(a_k^* - W_k^\top \delta_{k+1}^+\right)\right\|^2 + \sum D_{\bar{G}_k}(z_k^* \| a_k^*)\end{aligned} \tag{20}$$

(where $a_k^* := W_{k-1}z_{k-1}^*$ is the pre-synaptic activation). The last two lines are zero for the solution of the forward and backward pass, and $\delta_L^\pm = -\beta g$ minimizes the remaining first line. The updates are not necessarily decreasing (20) in each step, in particular they cannot be derived as a block-coordinate method to minimize (20).

3.4. Biological plausibility

One may conceptualize our framework in different ways. Mathematically it is natural to frame the learning objective in terms of the positively and negatively nudged states z^+ and z^- as they are elements in the same space, and the optimization problem is easy to (approximately) solve. However, in a biological circuit one might also conceive of the difference (δ_k) and mean ($\bar{z}_k$) of the nudged states as the actual compartments of the neuron.

This is superficially similar to the approach taken by (Guerguiev et al., 2017), in which feedback signals are integrated in the apical dendrite and feedforward signals are integrated in the basal dendrite. However, in our case the error $\delta_k = \frac{1}{2}(z_k^+ - z_k^-)$ also depends on the feed-forward input. Inspired by (Sacramento et al., 2018) this dependence on both feed-forward and feed-back signals (or a relaxed version of it), may be achieved using auxiliary neurons.

In practice our method (as well as all the other methods listed in Table 1) are far from modelling the complexity of biological neurons. Neuroscience research suggests that dendrites are multi-compartmental structures, which in terms of computational capabilities are closer to small networks of artificial neurons than to individual artificial neurons. E.g. a single dendrite can solve the XOR problem (Chavlis & Poirazi, 2021; Beniaguev et al., 2021), whereas a dot product based artificial neuron (i.e. McCulloch-Pitts neuron) can not. As such the proposed dyadic framework is

still a greatly simplified model, and should be seen as an example of a minimal circuitry required to perform effective credit-assignment while obeying certain known biological constraints, such as using only local information, being capable of operating asynchronously and not requiring computing the derivatives of the nonlinearities.

4. Implementation

We conducted all experiments by jointly optimizing $\mathcal{L}_{1/2}$ in (1) with respect to weights and activations. In order to train a network in a supervised setting using (6) and (9), information about the input data as well as about the prediction error needs to be propagated to all layers. In a network with L layers (counting hidden layers and output layer) this would require $L-1$ updates of all neurons. However, since (6) has closed form solution, the most economical approach is to simply update neurons sequentially: layer $1 \to 2 \to \ldots \to L \to L-1 \to \ldots \to 1$. If the network is initialized with zero activity, then the first $L-1$ updates reduce to a standard feed-forward pass. This is shown on the example of a single batch in Algorithm 1 (black and blue lines). The algorithm can be efficiently implemented on top of an existing autodiff framework using custom derivative rules. A less cost-efficient approach (on traditional computing hardware) is to randomly select layers to update for some T_{max} number of iterations. This is done in a very similar way, as shown in Algorithm 1 by the red lines (replacing the blue ones).

Algorithm 1 Dual propagation

$\mathbf{z_0^+},\ \mathbf{z_0^-} \leftarrow \mathbf{x},\ \mathbf{x}$
$\mathbf{z_k^+},\ \mathbf{z_k^-} \leftarrow \mathbf{0},\ \mathbf{0} \qquad \forall\, 0 < k \leq L$
for $k \in [1, ..., L-1]$ **do** // Regular
  $z_k^+, z_k^- \leftarrow f_k(z_{k-1}^+), f_k(z_{k-1}^-)$
end
for $T \in [1, ..., T_{max}]$ **do** // Random
  $k \leftarrow$ sample inter from [1,..., L]
  update z_k^+, z_k^- using (6) / (8) / (7)
end
for $k \in [L, ..., 1]$ **do**
  update z_k^+, z_k^- using (6) / (8) / (7) // Regular
  $W_{k-1} \leftarrow W_{k-1} - \frac{\eta}{2\beta_k}(z_k^+ - z_k^-)(z_{k-1}^+ + z_{k-1}^-)^\top$
end for

Our networks use the ReLU non-linearity, hence $G_k(z_k) = \|z_k\|^2/2 + \imath_{\geq 0}(z_k)$ for $k = 1, \ldots, L-1$, and the output layer is linear ($G_L(z_L) = \|z_L\|^2/2$). Note that G_k is not actually used in (6) or (9). It is only necessary to compute G_k if one wants to monitor $\mathcal{L}_\alpha$ during training.

4.1. Target Loss Functions

In a small MLP based a sensitivity analysis of the impact of the choice of β_L (Section 5.1), we employed a linearized MSE function by using (8). As mentioned in Section 3.2, this allowed us to try out larger values of β_L. However, in experiments where repeated state updates are made, one needs to make a choice of what linearization point to use at subsequent iterations. For this reason we simply employed the regular MSE loss in the remaining MLP experiments of Section 5.1 and used the update rule (7).

For the experiments on deep convolutional neural networks in Section 5.2 the mean square error loss was insufficient for achieving good performance, so a linearized version of softmax cross-entropy (the cross-entropy loss of the softmax of the output neurons) was employed.

4.2. Max-Pooling Layers

As in back-propagation, units participating in a max-pooling operation are suppressed unless they are the local maximum. The suppressed units do not receive feedback from upstream layers and do not propagate their activity forwards. For the efficient version of dual propagation (blue in Algorithm 1) this is achieved by using the standard autodiff rules for max-pooling layers. This is similar to the approach taken by (Laborieux et al., 2021).

5. Experiments

We evaluate dual propagation on MNIST, CIFAR10, CIFAR100, and Imagenet32x32. MNIST is used to analyse variations of the algorithm, the other datasets for comparison with back-propagation. Through this we show that DP performs nearly identical to BP in both runtime and accuracy. Our code is available on github[2].

5.1. MLP Trained on MNIST

Before applying dual propagation on more challenging tasks and datasets, we explored the impact of variations of the algorithm on the MNIST digit classification dataset, using a ReLU MLP with architecture $784-1000-1000-1000-1000-10$.

Apart from DP and R-DP presented in Algorithm 1, we also explore three other variations. Lazy dual propagation (L-DP) differs from DP only in that hidden and output units are not reset to zero activity before processing a new batch of data. This means that feature vectors from previous data points are allowed to provide potentially disruptive feedback. In multi-step dual propagation (MS-DP) we perform five inference passes up and down on the same batch with a

[2] https://github.com/Rasmuskh/Dual-Propagation

weight update after each full pass. This is qualitatively similar to the repeated weight updates employed in (Ernoult et al., 2020; Salvatori et al., 2022). Parallel dual propagation (P-DP) updates all neurons in parallel. This method requires $2L - 1$ updates for an informative signal to reach all layers. The output neurons only receive feedback from the loss function during the last L updates, since the loss signal isn't meaningful until the input signal has had a chance to reach the output layer.

As mentioned, it is advantageous to choose $\beta_k = \beta_{k+1}$ for $1 \leq k \leq L$, but that still leaves us with the freedom to choose β_L. The error signal arriving at layer $L - 1$ is inversely proportional to β_L, making it necessary to divide the learning rate by β_L.

The impact of different choices of the hyper-parameter β_L is explored in an initial sensitivity experiment. As shown in Table 2, $\beta_L = 1$ was found to perform the best and was consequently used in subsequent experiments.

Table 2: Performance impact of different choices of β_L on MNIST test accuracy.

β_L	0.01	0.1	1.0	10	100
Test acc. (%)	98.04	98.34	98.38	94.95	85.42

Table 3: MNIST test accuracies obtained with an MLP with 4 hidden layers (of 1000 units each).

Method	BP	DP	MS-DP	L-DP	P-DP	R-DP-100
Test acc (%)	98.45 ±0.04	98.43 ±0.03	98.40 ±0.02	98.42 ±0.07	98.47 ±0.04	98.48 ±0.11

The MLP was trained on the MNIST dataset using the ADAM optimizer (Kingma & Ba, 2014). 10% of the training data was reserved for validation, and performance on the validation data was used to select which checkpoint of the model to evaluate on the test dataset. The resulting test accuracy, summarized in Table 3, illustrate that the variations of DP (and BP) essentially perform equivalently. However, for R-DP it was found to be essential that a sufficient number of neuron updates are made. Fig. 2 illustrates this. With 60 updates learning fails, with 80 updates the loss converges noisily, and with 100 updates it converges smoothly.

Fig. 3 shows the average angle between the layerwise gradients computed by DP and the corresponding analytical BP gradient for one of the random seeds. Throughout the 100 epoch run, the average angle remains below $11.5°$ (corresponding to a minimum cosine similarity of 0.98).

5.2. Deep CNN Experiments

We benchmarked dual propagation against back-propagation on the CIFAR10 and CIFAR100 datasets (Krizhevsky et al., 2009). The algorithms were used to train a VGG16 model (Simonyan & Zisserman, 2014). A variant model with distinct forward and backward weights was also trained with dual propagation, using the Kolen-Pollack algorithm to learn the feedback weights (Kolen & Pollack, 1994). The models were trained using standard data augmentation techniques (random-crops and horizontal flips) and the training data of 50 000 images was split into 45 000 for training and 5 000 for validation. For each run, a snapshot of the model was selected based on validation accuracy and later evaluated on test data. The average test accuracy across five random seeds are listed in Table 4. Interestingly the KP-DP approach performs slightly better on CIFAR100, suggesting that the initially unaligned feedback weights might have a regularizing effect. Training accuracy and softmax cross-entropy loss across epochs are plotted in Fig. 4. On an NVIDIA A100 GPU both BP and DP had runtimes of ∼3.5 seconds per epoch and ∼4.5 seconds per epoch for CIFAR10 and CIFAR100 respectively (a smaller batchsize was used for CIFAR100). KP-DP had an additional overhead of about 1 second per epoch, presumably due having to update an additional set of weights.

The VGG16 model was also trained on Imagenet32x32, yielding essentially equivalent performance for DP and BP. Standard data augmentation techniques (random-crops and horizontal flips) was also employed in this experiment. As Imagenet32x32 does not have a public test dataset we used the validation data as test data and reserved 5% of the training data for validation. Training time per epoch was ∼61 seconds on an NVIDIA A100 gpu for both methods. The average test accuracies accross five random seeds are listed in Table 4. Kolen-Pollack learning did not work for this dataset.

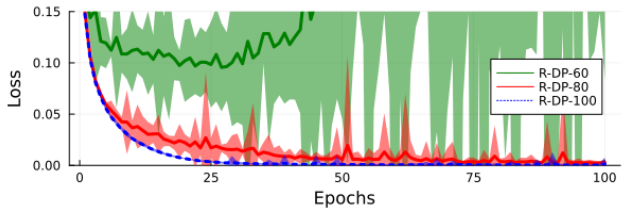


Figure 2: MNIST training loss for dual propagation for different numbers of randomly ordered layer updates. Shaded regions indicate ±3 standard deviations.

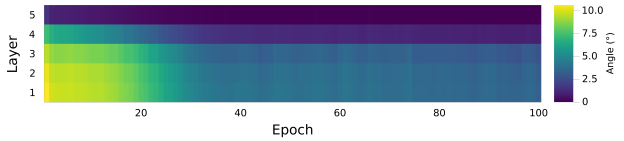


Figure 3: Layerwise angle between gradients computed using dual propagation and back-propagation in a five layer MLP.

Table 4: CIFAR10, CIFAR100 and ImageNet32x32 test accuracy, obtained with dual propagation (DP) Kolen-Polack dual propagation (KP-DP) and back-propagation (BP) using a VGG16 architecture. (*) The to our knowledge best published results for equilibrium propagation (EP) (Laborieux & Zenke, 2022) and difference target propagation (DTP) (Ernoult et al., 2022) are listed in the rightmost columns for reference (note that these are based on different network architectures).

Method		BP	DP	KP-DP	EP*	DTP*
CIFAR10	Top-1	92.26 ± 0.23	92.30 ± 0.11	91.84 ± 0.11	88.6 ± 0.2	89.38 ± 0.20
CIFAR100	Top-1	69.63 ± 0.24	69.57 ± 0.51	70.40 ± 0.25	61.6 ± 0.1	–
	Top-5	88.13 ± 0.22	88.36 ± 0.13	88.57 ± 0.15	86.0 ± 0.1	–
ImageNet 32x32	Top-1	41.28 ± 0.19	41.48 ± 0.19	–	36.5 ± 0.3	36.81
	Top-5	64.89 ± 0.11	64.90 ± 0.13	–	60.8 ± 0.4	60.54

We observe that the asymmetric network became successively more sensitive to the choice of hyper-parameters as the number of classes increased (from CIFAR10 to CIFAR100 to Imagenet32x32).

For reference we also report the, to our knowledge, best results for NGRAD algorithms in the rightmost columns of Table 4, namely the results for (Holomorphic) Equilibrium propagation ((Laborieux & Zenke, 2022)) and a recent variant of difference target propagation (Ernoult et al., 2022). We emphasize that both EP and DTP are capable of approximating back-propagation, provided a sufficient amount of computational resources and time is dedicated. For EP this means that sufficiently many inference iterations must be run and for DTP it means that sufficiently many iterations of feedback weight learning must be run. Thus, the performance difference between these algorithms and DP, as shown in Table 4, mainly reflects their high computational costs, which makes hyperparameter search challenging and, crucially, makes training very deep networks computationally infeasible. Consequently both the results for EP (Laborieux & Zenke, 2022) and DTP (Ernoult et al., 2022) were obtained using small VGG-like networks (5-7 layers).

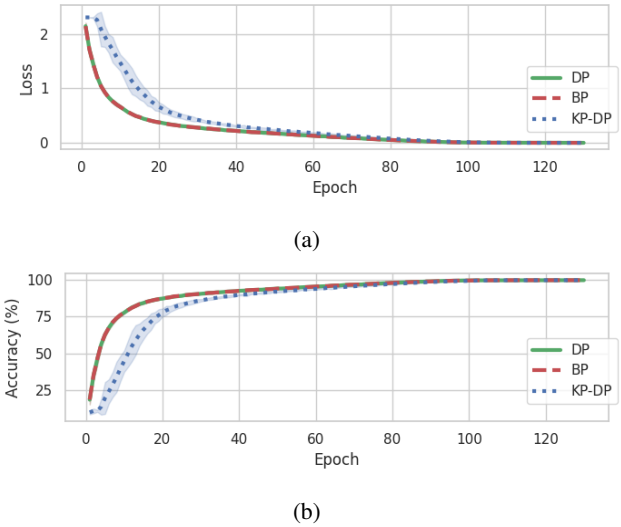


Figure 4: Training metrics (accuracy and softmax cross-entropy loss), for VGG16 networks trained on CIFAR10 with back-propagation (BP), dual propagation (DP) and a variant of dual propagation with distinct feedback weights trained with Kolen-Pollack algorithm KP-DP. Shaded regions indicate ± 3 standard deviations.

6. Conclusion

Variations of contrastive Hebbian learning are gaining traction as they are possibly highly suitable for DNN training on energy-efficient analog computing devices. However, their high computational demand when implemented on digital hardware is clearly a disadvantage when exploring these algorithms. Our proposed algorithm, dual propagation, differs from traditional contrastive Hebbian learning algorithms in that the errors are computed across different compartments of individual neurons rather than across different temporal states of the same neuron. The resulting formulation allows for closed-form neuron update rules, which makes dual propagation competitive with back-propagation, both in terms of accuracy and runtime.

An important question for future work is whether the proposed dual propagation method is easily implementable on analog computing hardware. In the digital realm, dual propagation may prove highly valuable for training quantized neural networks by a suitable choice of β_k, hence steering the finite difference approximation of the possibly non-smooth activation function.

Acknowledgements This work was supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, and by the Chalmers AI Research Centre (CHAIR). The experiments were enabled by the supercomputing resource Berzelius provided by National Supercomputer Centre at Linköping University and the Knut and Alice Wallenberg foundation.

References

Akrout, M., Wilson, C., Humphreys, P., Lillicrap, T., and Tweed, D. B. Deep learning without weight transport. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Askari, A., Negiar, G., Sambharya, R., and Ghaoui, L. E. Lifted neural networks. *arXiv preprint arXiv:1805.01532*, 2018.

Baldi, P., Sadowski, P., and Lu, Z. Learning in the machine: Random backpropagation and the deep learning channel. *Artificial intelligence*, 260:1–35, 2018.

Bartunov, S., Santoro, A., Richards, B., Marris, L., Hinton, G. E., and Lillicrap, T. Assessing the scalability of biologically-motivated deep learning algorithms and architectures. *Advances in neural information processing systems*, 31, 2018.

Beniaguev, D., Segev, I., and London, M. Single cortical neurons as deep artificial neural networks. *Neuron*, 109 (17):2727–2739, 2021.

Carreira-Perpinan, M. and Wang, W. Distributed optimization of deeply nested systems. In *Artificial Intelligence and Statistics*, pp. 10–19, 2014.

Chavlis, S. and Poirazi, P. Drawing inspiration from biological dendrites to empower artificial neural networks. *Current opinion in neurobiology*, 70:1–10, 2021.

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

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

Ernoult, M. M., Normandin, F., Moudgil, A., Spinney, S., Belilovsky, E., Rish, I., Richards, B., and Bengio, Y. Towards scaling difference target propagation by learning backprop targets. In *International Conference on Machine Learning*, pp. 5968–5987. PMLR, 2022.

Gu, F., Askari, A., and El Ghaoui, L. Fenchel lifted networks: A lagrange relaxation of neural network training. In *International Conference on Artificial Intelligence and Statistics*, pp. 3362–3371. PMLR, 2020.

Guerguiev, J., Lillicrap, T. P., and Richards, B. A. Towards deep learning with segregated dendrites. *eLife*, 6:e22901, dec 2017. ISSN 2050-084X. doi: 10.7554/eLife.22901.

Høier, R. K. and Zach, C. Lifted regression/reconstruction networks. *arXiv preprint arXiv:2005.03452*, 2020.

Jaderberg, M., Czarnecki, W. M., Osindero, S., Vinyals, O., Graves, A., Silver, D., and Kavukcuoglu, K. Decoupled neural interfaces using synthetic gradients. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1627–1635. JMLR. org, 2017.

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

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kolen, J. and Pollack, J. Backpropagation without weight transport. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, volume 3, pp. 1375–1380 vol.3, 1994. doi: 10.1109/ICNN.1994.374486.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Laborieux, A. and Zenke, F. Holomorphic equilibrium propagation computes exact gradients through finite size oscillations. *arXiv preprint arXiv:2209.00530*, 2022.

Laborieux, A., Ernoult, M., Scellier, B., Bengio, Y., Grollier, J., and Querlioz, D. Scaling equilibrium propagation to deep convnets by drastically reducing its gradient estimator bias. *Frontiers in neuroscience*, 15:129, 2021.

Lee, D.-H., Zhang, S., Fischer, A., and Bengio, Y. Difference target propagation. In *Joint european conference on machine learning and knowledge discovery in databases*, pp. 498–515. Springer, 2015.

Li, J., Fang, C., and Lin, Z. Lifted proximal operator machines. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4181–4188, 2019.

Li, J., Xiao, M., Fang, C., Dai, Y., Xu, C., and Lin, Z. Training neural networks by lifted proximal operator machines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

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

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

Lillicrap, T. P., Santoro, A., Marris, L., Akerman, C. J., and Hinton, G. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346, 2020.

Meulemans, A., Carzaniga, F., Suykens, J., Sacramento, J., and Grewe, B. F. A theoretical framework for target propagation. *Advances in Neural Information Processing Systems*, 33:20024–20036, 2020.

Moskovitz, T. H., Litwin-Kumar, A., and Abbott, L. Feedback alignment in deep convolutional networks. *arXiv preprint arXiv:1812.06488*, 2018.

Movellan, J. R. Contrastive hebbian learning in the continuous hopfield model. In *Connectionist models*, pp. 10–17. Elsevier, 1991.

Nøkland, A. Direct feedback alignment provides learning in deep neural networks. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

O'Reilly, R. C. Biologically plausible error-driven learning using local activation differences: The generalized recirculation algorithm. *Neural computation*, 8(5):895–938, 1996.

O'Connor, P., Gavves, E., and Welling, M. Training a spiking neural network with equilibrium propagation. In *The 22nd international conference on artificial intelligence and statistics*, pp. 1516–1523. PMLR, 2019.

Refinetti, M., D'Ascoli, S., Ohana, R., and Goldt, S. Align, then memorise: the dynamics of learning with feedback alignment. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8925–8935. PMLR, 18–24 Jul 2021.

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

Sacramento, J., Ponte Costa, R., Bengio, Y., and Senn, W. Dendritic cortical microcircuits approximate the backpropagation algorithm. *Advances in neural information processing systems*, 31, 2018.

Salvatori, T., Song, Y., Millidge, B., Xu, Z., Sha, L., Emde, C., Bogacz, R., and Lukasiewicz, T. Incremental predictive coding: A parallel and fully automatic learning algorithm. *arXiv preprint arXiv:2212.00720*, 2022.

Scellier, B. A deep learning theory for neural networks grounded in physics. *arXiv preprint arXiv:2103.09985*, 2021.

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

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

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Song, Y., Lukasiewicz, T., Xu, Z., and Bogacz, R. Can the brain do backpropagation?—exact implementation of backpropagation in predictive coding networks. *NeuRIPS Proceedings 2020*, 33(2020), 2020.

Strubell, E., Ganesh, A., and McCallum, A. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*, 2019.

Taylor, G., Burmeister, R., Xu, Z., Singh, B., Patel, A., and Goldstein, T. Training neural networks without gradients: A scalable admm approach. In *International Conference on Machine Learning*, pp. 2722–2731, 2016.

Whittington, J. C. and Bogacz, R. An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural computation*, 29(5):1229–1262, 2017.

Xie, X. and Seung, H. S. Equivalence of backpropagation and contrastive hebbian learning in a layered network. *Neural computation*, 15(2):441–454, 2003.

Yi, S.-i., Kendall, J. D., Williams, R. S., and Kumar, S. Activity-difference training of deep neural networks using memristor crossbars. *Nature Electronics*, pp. 1–7, 2022.

Zach, C. Bilevel programs meet deep learning: A unifying view on inference learning methods. *arXiv preprint arXiv:2105.07231*, 2021.

Zach, C. and Estellers, V. Contrastive learning for lifted networks. In Sidorov, K. and Hicks, Y. (eds.), *Proceedings of the British Machine Vision Conference (BMVC)*, pp. 163.1–163.12. BMVA Press, 9 2019. doi: 10.5244/C.33.163.

Zhang, Z. and Brand, M. Convergent block coordinate descent for training tikhonov regularized deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 1721–1730, 2017.

Zoppo, G., Marrone, F., and Corinto, F. Equilibrium propagation for memristor-based recurrent neural networks. *Frontiers in neuroscience*, 14:240, 2020.

A. Deriving the update relations (6)

Solving for z_k^+: The terms in $\mathcal{L}_\alpha$ (1) dependent on z_k^+ are given by

$$\begin{aligned}
V_k^+(z_k^+) &= \beta_k^{-1}\left(\alpha G_k(z_k^+) - \alpha(z_k^+)^\top W_{k-1}z_{k-1}^+\right) + \bar{\alpha}G_k(z_k^+) - \bar{\alpha}(z_k^+)^\top W_{k-1}z_{k-1}^-\big) \\
&\quad + \beta_{k+1}^{-1}\alpha\left(G_{k+1}(z_{k+1}^+) - (z_{k+1}^+)^\top W_k z_k^+ - G_{k+1}(z_{k+1}^-) + (z_{k+1}^-)^\top W_k z_k^+\right) \\
&\doteq \beta_k^{-1}\left(G_k(z_k^+) - \alpha(z_k^+)^\top W_{k-1}z_{k-1}^+\right) - \bar{\alpha}(z_k^+)^\top W_{k-1}z_{k-1}^-\big) + \beta_{k+1}^{-1}\alpha\left(-(z_{k+1}^+)^\top W_k z_k^+ + (z_{k+1}^-)^\top W_k z_k^+\right) \\
&\propto G_k(z_k^+) - \alpha(z_k^+)^\top W_{k-1}z_{k-1}^+\big) - \bar{\alpha}(z_k^+)^\top W_{k-1}z_{k-1}^- + \tfrac{\alpha\beta_k}{\beta_{k+1}}\left(-(z_{k+1}^+)^\top W_k z_k^+ + (z_{k+1}^-)^\top W_k z_k^+\right) \\
&= G_k(z_k^+) - (z_k^+)^\top\left(\alpha W_{k-1}z_{k-1}^+ + \bar{\alpha}W_{k-1}z_{k-1}^- + \tfrac{\alpha\beta_k}{\beta_{k+1}}W_k^\top(z_{k+1}^+ - z_{k+1}^-)\right).
\end{aligned} \tag{21}$$

Hence, z_k^+ is given by

$$z_k^+ \leftarrow f_k\left(\alpha W_{k-1}z_{k-1}^+ + \bar{\alpha}W_{k-1}z_{k-1}^- + \tfrac{\alpha\beta_k}{\beta_{k+1}}W_k^\top(z_{k+1}^+ - z_{k+1}^-)\right). \tag{22}$$

Solving for z_k^-: Analogously, the terms in $\mathcal{L}_\alpha$ dependent on z_k^- are given by

$$\begin{aligned}
V_k^-(z_k^-) &\propto -G_k(z_k^-) + \alpha(z_k^-)^\top W_{k-1}z_{k-1}^+ + \bar{\alpha}(z_k^-)^\top W_{k-1}z_{k-1}^- \\
&\quad + \tfrac{\bar{\alpha}\beta_k}{\beta_{k+1}}\left(G_{k+1}(z_{k+1}^+) - (z_{k+1}^+)^\top W_k z_k^- - G_{k+1}(z_{k+1}^-) + (z_{k+1}^-)^\top W_k z_k^-\right) \\
&\doteq -G_k(z_k^-) + \alpha(z_k^-)^\top W_{k-1}z_{k-1}^+ + \bar{\alpha}(z_k^-)^\top W_{k-1}z_{k-1}^- + \tfrac{\bar{\alpha}\beta_k}{\beta_{k+1}}\left(-(z_{k+1}^+)^\top W_k z_k^- + (z_{k+1}^-)^\top W_k z_k^-\right) \\
&= -G_k(z_k^-) + (z_k^-)^\top\left(\alpha W_{k-1}z_{k-1}^+ + \bar{\alpha}W_{k-1}z_{k-1}^- + \tfrac{\bar{\alpha}\beta_k}{\beta_{k+1}}W_k^\top(z_{k+1}^- - z_{k+1}^+)\right),
\end{aligned} \tag{23}$$

which implies

$$z_k^- \leftarrow f_k\left(\alpha W_{k-1}z_{k-1}^+ + \bar{\alpha}W_{k-1}z_{k-1}^- + \tfrac{\bar{\alpha}\beta_k}{\beta_{k+1}}W_k^\top(z_{k+1}^- - z_{k+1}^+)\right). \tag{24}$$

B. Propagation of asymmetric finite differences

We consider a more general version of (16): $\delta_L^\pm$ are given by $\delta_L^+ \leftarrow -\alpha\beta g$ and $\delta_L^- \leftarrow \bar{\alpha}\beta g$ (assuming a linearized target loss ℓ with gradient g at the current linearization point), and the two feedback signals are propagated through the network via

$$\delta_k^+ \leftarrow f_k\left(W_{k-1}z_{k-1}^* + \alpha W_k^\top\delta_{k+1}^+\right) - z_k^* \qquad\qquad \delta_k^- \leftarrow z_k^* - f_k\left(W_{k-1}z_{k-1}^* - \bar{\alpha}W_k^\top\delta_{k+1}^-\right), \tag{25}$$

where $\alpha \in [0,1]$ and $\bar{\alpha} := 1-\alpha$. Our aim to is reparametrize $\delta_k^\pm$ using $z_k^\pm$ such that

$$z_k^+ - z_k^- = \delta_k^+ + \delta_k^- = f_k\left(W_{k-1}z_{k-1}^* + \alpha W_k^\top\delta_{k+1}^+\right) - f_k\left(W_{k-1}z_{k-1}^* - \bar{\alpha}W_k^\top\delta_{k+1}^-\right) \tag{26}$$

and $z_k^* = \bar{z}_k = \alpha z_k^+ + \bar{\alpha}z_k^-$. After introducing $\delta_k := \delta_k^+ + \delta_k^-$, combining these constraints yields

$$\alpha z_k^+ = \alpha z_k^- + \alpha\delta_k \qquad\qquad \alpha z_k^+ = z_k^* - \bar{\alpha}z_k^-, \tag{27}$$

which implies

$$\alpha z_k^- + \alpha\delta_k - z_k^* + \bar{\alpha}z_k^- = 0 \iff z_k^- = z_k^* - \alpha\delta_k \tag{28}$$

and analogously $z_k^+ = z_k^* + \bar{\alpha}\delta_k$. Observe that there is an asymmetry in the role of α in the forward and backward (adjoint) process, e.g. choosing $\alpha = 0$ yields $(z_k^+, z_k^-) = (z_k^* - \delta_k, z_k^*)$ and $(\delta_k^+, \delta_k^-) = (0, \delta_k)$. We have agreement of this model with the updates in (6) and (8) only when $\alpha = 1/2$. Consequently, the reasoning presented in Section 3.3 applies only for the choice of $\alpha = 1/2$.

C. Hyper-parameter settings

The hyper-parameters used in the experiments of Section 5.1 are listed in Table 5. All experiments used these hyper-parameters except for MS-DP which used a learning rate of 6e-6 (to account for the higher number of weight updates per minibatch) and the experiments listed in Table 2, which only trained for 50 epochs. η is the learning rate and b_1, b_2 and ϵ are parameters for the ADAM optimizer. The hyper-parameters used in the experiments of Section 5.2 are listed in Table 6. For these experiments a linear learning rate warmup schedule was employed followed by cosine decay.

Table 5: Hyper-parameters used in the MLP experiments.

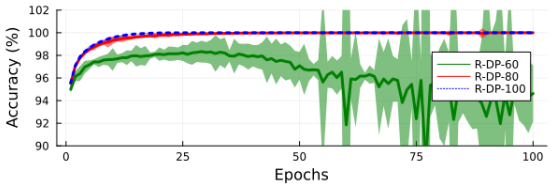
Dataset	Optimizer	η	b_1	b_2	ϵ	Epochs	batchsize
MNIST	ADAM	3e-5	0.9	0.999	$1e-8$	100	100

Table 6: Hyper-parameters used in VGG16 experiments.

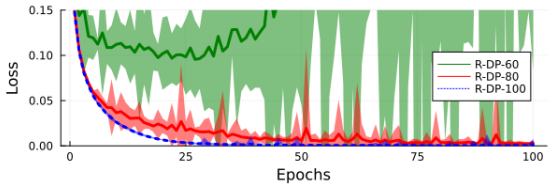
Dataset	Model	Momentum	η_{start}	η_{peak}	η_{end}	Warmup epochs	Epochs	Weight decay	batchsize
CIFAR10	BP	0.9	0.005	0.025	0	10	130	5e-4	100
	DP	0.9	0.005	0.025	0	10	130	5e-4	100
	KP-DP	0.9	0.0001	0.025	0	15	130	5e-4	100
CIFAR100	BP	0.9	0.005	0.015	0	10	200	5e-4	50
	DP	0.9	0.005	0.015	0	10	200	5e-4	50
	KP-DP	0.9	0.0001	0.015	0	30	200	5e-4	50
Imagenet32x32	BP	0.9	0.005	0.015	0	10	200	5e-4	250
	DP	0.9	0.005	0.015	0	10	200	5e-4	250

D. Plots of training metrics

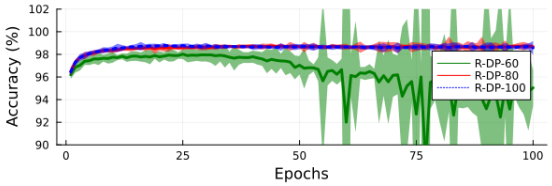
We illustrate the training evolution of DP and BP on the various datasets in Figs. 5, 6 and 7, respectively.



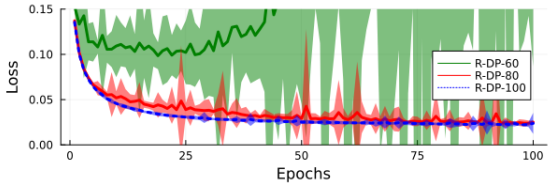
(a) MNIST: Training accuracy.



(b) MNIST: Training loss.



(c) MNIST: Validation accuracy.



(d) MNIST: Validation loss.

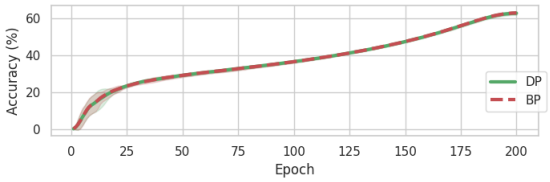
Figure 5: Training metrics (accuracy and MSE), for five layer MLP trained with R-DP using different numbers of random updates. Shaded regions indicate ± 3 standard deviations.

E. Network architecture

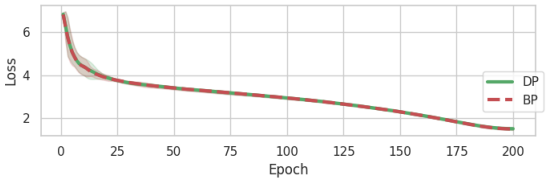
To allow for full reproducibility, Table 7 shows the architecture of the VGG16 version without batchnorm used in Sec 5.2.

Table 7: The architecture of the VGG16 network used in CIFAR10, CIFAR100 and Imagenet32x32 experiments. All convolutional layers have 3×3 kernels with stride 1, all pooling layers are of size 2×2 with stride 2. I.e., convolutional layers preserve image size, pooling layers half it. The z^+/z^- dyad is implemented in the ReLU layers. For the Kolen-Pollack variation, all Convolution and Fully Connected layers are replaced with asymmetric variants.

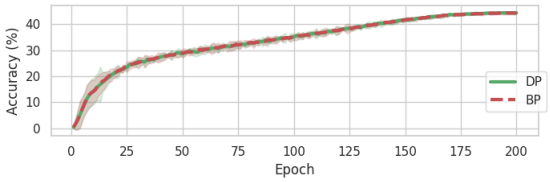
	Type	Kernel/Stride	Channels or Size
1	Convolutional	3×3 / 1	64
2	ReLU	–	–
3	Convolutional	3×3 / 1	64
4	ReLU	–	–
5	Max Pool	2×2 / 2	
6	Convolutional	3×3 / 1	128
7	ReLU	–	–
8	Convolutional	3×3 / 1	128
9	ReLU	–	–
10	Max Pool	2×2 / 2	
11	Convolutional	3×3 / 1	256
12	ReLU	–	–
13	Convolutional	3×3 / 1	256
14	ReLU	–	–
15	Convolutional	3×3 / 1	256
16	ReLU	–	–
17	Max Pool	2×2 / 2	–
18	Convolutional	3×3 / 1	512
19	ReLU	–	–
20	Convolutional	3×3 / 1	512
21	ReLU	–	–
22	Convolutional	3×3 / 1	512
23	ReLU	–	–
24	Max Pool	2×2 / 2	–
25	Convolutional	3×3 / 1	512
26	ReLU	–	–
27	Convolutional	3×3 / 1	512
28	ReLU	–	–
29	Convolutional	3×3 / 1	512
30	ReLU	–	–
31	Max Pool	2×2 / 2	–
32	Flatten	–	–
33	Fully Connected	–	4096
34	ReLU	–	–
35	Fully Connected	–	4096
36	ReLU	–	–
37	Fully Connected	–	#Classes



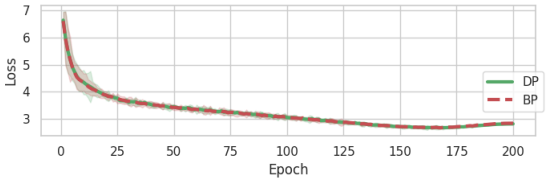
(a) Imagenet32x32: Training accuracy.



(b) Imagenet32x32: Training loss.

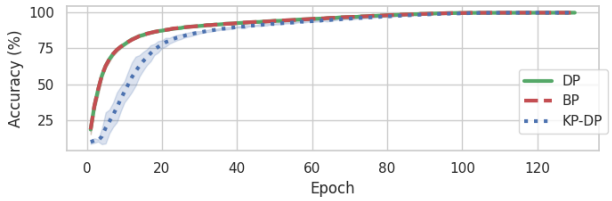


(c) Imagenet32x32: Validation accuracy.

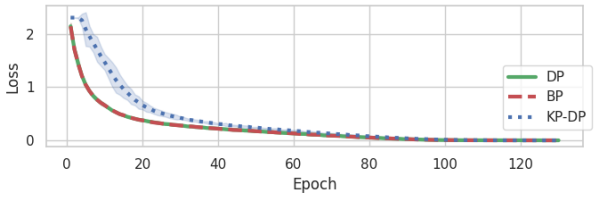


(d) Imagenet32x32: Validation loss.

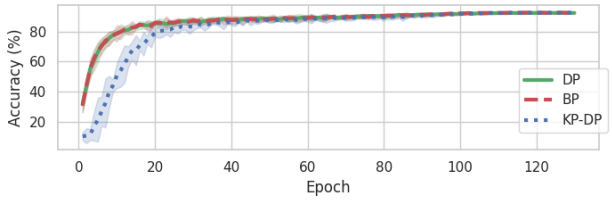
Figure 6: Training metrics (accuracy and softmax cross-entropy loss), for VGG16 networks trained with back-propagation (BP) and dual propagation (DP). Shaded regions indicate ± 3 standard deviations.



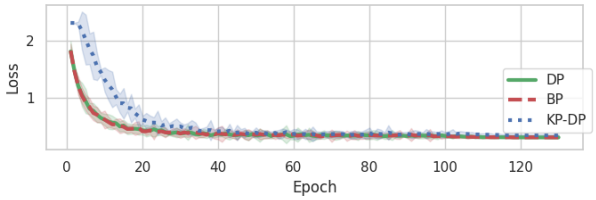
(a) CIFAR10: Training accuracy.



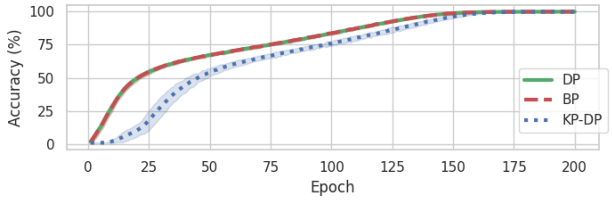
(b) CIFAR10: Training loss.



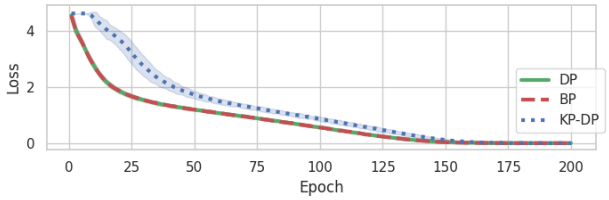
(c) CIFAR10: Validation accuracy.



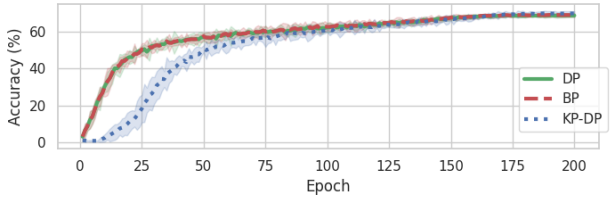
(d) CIFAR10: Validation loss.



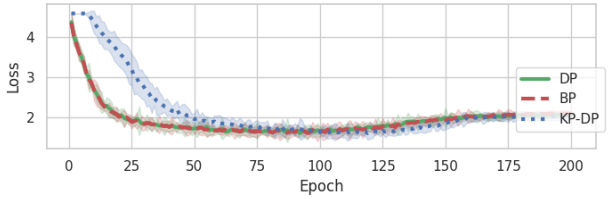
(e) CIFAR100: Training accuracy.



(f) CIFAR100: Training loss.



(g) CIFAR100: Validation accuracy.



(h) CIFAR100: Validation loss.

Figure 7: Training metrics (accuracy and softmax cross-entropy loss), for VGG16 networks trained with back-propagation (BP), dual propagation (DP) and a variant of dual propagation with distinct feedback weights trained with Kolen-Pollack algorithm KP-DP. Shaded regions indicate ± 3 standard deviations.