ASSESSING THE SCALABILITY OF BIOLOGICALLY-MOTIVATED DEEP LEARNING ALGORITHMS AND ARCHITECTURES

Anonymous authors
Paper under double-blind review

ABSTRACT

The backpropagation of error algorithm (BP) is often said to be impossible to implement in a real brain. The recent success of deep networks in machine learning and AI, however, has inspired a number of proposals for understanding how the brain might learn across multiple layers, and hence how it might implement or approximate BP. As of yet, none of these proposals have been rigorously evaluated on tasks where BP-guided deep learning has proved critical, or in architectures more structured than simple fully-connected networks. Here we present the first results on scaling up a biologically motivated model of deep learning to datasets which need deep networks with appropriate architectures to achieve good performance. We present results on CIFAR-10 and ImageNet. For CIFAR-10 we show that our algorithm, a straightforward, weight-transport-free variant of difference target-propagation (DTP) modified to remove backpropagation from the penultimate layer, is competitive with BP in training deep networks with locally defined receptive fields that have untied weights. For ImageNet we find that both DTP and our algorithm perform significantly worse than BP, opening questions about whether different architectures or algorithms are required to scale these approaches. Our results and implementation details help establish baselines for biologically motivated deep learning schemes going forward.

1 INTRODUCTION

The suitability of the backpropagation of error (BP) algorithm (Rumelhart et al., 1986) for explaining learning in the brain was questioned soon after its popularization (Grossberg, 1987; Crick, 1989). Weaker objections included undesirable characteristics of artificial networks in general, such as their violation of Dale's Law, their lack of cell-type variability, and the need for the gradient signals to be both positive and negative. Much more serious objections were: (1) The need for the feedback connections carrying the gradient to have the same weights as the corresponding feedforward connections and (2) The need for a distinct form of information propagation (error propagation) that does not influence neural activity, and hence does not conform to known biological feedback mechanisms underlying neural communication.

Researchers have long sought biologically plausible and empirically powerful learning algorithms that avoid some of these flaws (Almeida, 1990; Pineda, 1987; 1988; Ackley et al., 1985; O'Reilly, 1996; Xie & Seung, 2003; Hinton & McClelland, 1988; Körding & König, 2001; Guergiuev et al., 2016b; Bengio et al., 2015; Lillicrap et al., 2016). A common theme of some of the most promising approaches – such as Contrastive Hebbian Learning (Movellan, 1991), and Generalized Recirculation (O'Reilly, 1996) – is to use feedback connections to influence *neural activity*, and to use *differences* in feedfoward-driven and feedback-driven activities or products of activities to locally approximate gradients (Ackley et al., 1985; Pineda, 1988; O'Reilly, 1996; Xie & Seung, 2003; Scellier & Bengio, 2017; Whittington & Bogacz, 2017). Since these activity propagation methods don't require explicit propagation of gradients through the network, they go a long way towards answering the second serious objection noted above. However, many of these methods require long "positive" and "negative" settling phases for computing the activities or activity products whose differences provide the learning signal. Proposals for shortening the phases (Hinton, 2007; Bengio et al., 2016) are not entirely satisfactory as they still fundamentally depend on a settling process,

and, in general, any settling process will likely be too slow for a brain that needs to quickly compute hidden activities. Indeed, for the same reason, only a handful of the algorithms that require settling have ever been used on large scale problems in machine learning.

Perhaps the most practical among this family of "activity propagation" algorithms is target propagation (TP) and its variants (LeCun, 1986; 1987; Hinton, 2007; Bengio, 2014; Lee et al., 2015). The intuition for TP is as follows: Suppose you have a feedforward neural network and have the capacity to compute perfect inverses backwards through the network (i.e., given the activities in layer h_{l+1}, we can compute $h_l = f^{-1}(h_{l+1}; \theta_{l+1})$). If we impose an output target (for a given input) on the output layer, then we can propagate activity backwards through the network to infer what the activities *should be* to produce the output target. These backwards propagated activities are denoted the *layer targets*, or simply *targets*. Then, when computing a feedfoward propagation through the network given some input, we can layer-wise compare the feedforward activations to what they should have been (i.e., the *targets*), and use the differences to compute weight changes. TP algorithms do not require settling dynamics, and thus can compute forward passes and updates quickly. As well, for one TP variant (Lee et al., 2015), it has been shown that weight changes that cause future feedforward activity to be nudged towards their targets approximate the weight changes computed by BP.

While TP and its variants are promising as biologically-plausible algorithms, there are some lingering questions about their applicability to the brain. First, the only variant explored empirically – difference target propagation (DTP) – still depends on explicit gradient computation via backpropagation for learning the penultimate layer's outgoing synaptic weights (see Algorithm Box 1 in Lee et al. (2015)). Second, they have not been tested on datasets more difficult than MNIST. And third, they have not been incorporated into architectures more complicated than simple multi-layer perceptrons (MLPs).

In this work we address each of these issues. Our contribution is threefold: (1) We examine the learning and performance of a biologically-motivated algorithm, Difference Target-propagation (DTP), on MNIST, CIFAR, and ImageNet, (2) We develop a variant of DTP called Simplified Difference Target Propagation (SDTP), which eliminates significant lingering biologically implausible features from DTP, and (3) We investigate the role of weight-sharing convolutions, which are key to performance on difficult datasets in artificial neural networks, by testing the effectiveness of locally connected architectures trained with BP, DTP, and SDTP.

Sharing the weights of locally connected units greatly reduces the number of free parameters and this has several very beneficial effects on computer simulations of large neural nets. It improves generalization and it drastically reduces both the amount of memory needed to store the parameters and the amount of communication required between replicas of the same model running on different subsets of the data on different processors. From a biological perspective we are interested in how STDP compares with BP without using weight sharing, so both our BP results and our SDTP results are considerably worse than convolutional neural nets and take far longer to produce.

2 Learning with Backpropagation versus Target Propagation

Consider the case of a feed-forward neural network with L layers $\{h_l\}_{l=1}^{L}$, whose activations h_l are computed by elementwise-applying a non-linear function σ_l to an affine transformation of previous layer activations h_{l-1}:

$$h_l = f(h_{l-1}; \theta_l) = \sigma_l(W_l h_{l-1} + b_l), \quad \theta_l = \{W_l, b_l\}, \tag{1}$$

with input to the network denoted as $h_0 = x$ and the last layer h_L used as output.

For example, in classification problems the output layer h_L parametrizes a predicted distribution over possible labels $p(y|h_L)$, usually using the softmax function. The learning signal is then provided as a loss $\mathcal{L}(h_L)$ incurred by making a prediction for an input x, which in the classification case can be cross-entropy between the ground-truth label distribution $q(y|x)$ and the predicted one:

$$\mathcal{L}(h_L) = -\sum_y q(y|x) \log p(y|h_L).$$

The goal of training is then to adjust the parameters $\Theta = \{\theta_l\}_{l=1}^{L}$ in order to minimize a given loss over the training set of inputs.

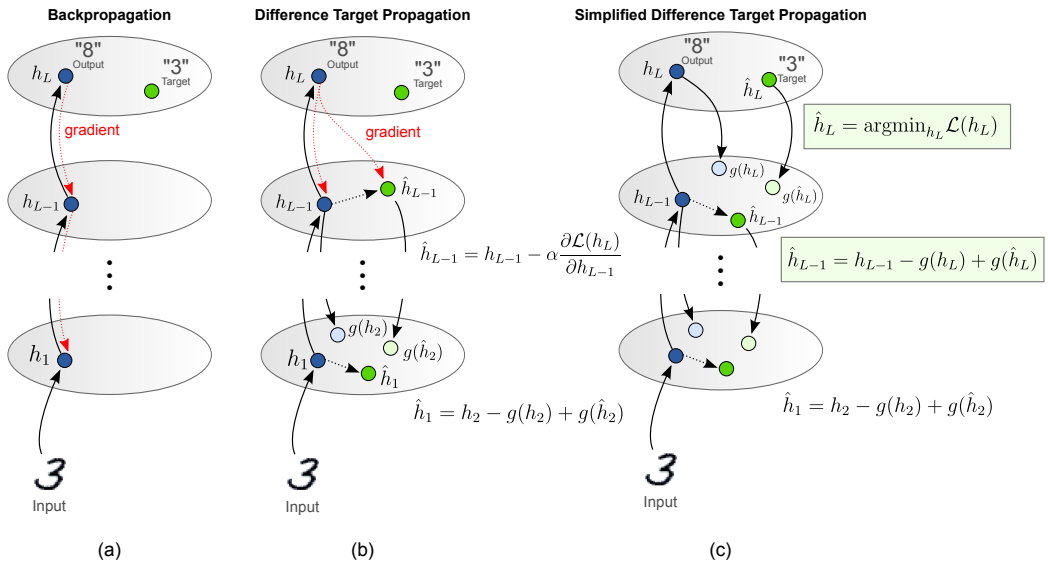


Figure 1: In BP and DTP, the final layer target is used to compute a loss, and the gradients from this loss are shuttled backwards (through all layers, in BP, or just one layer, in DTP) in error propagation steps that do not influence actual neural activity. SDTP never transports gradients using error propagation steps, unlike DTP and BP.

2.1 Backpropagation

Backpropagation (Rumelhart et al., 1986) was popularized as a method for learning in neural networks by computing gradients with respect to layer parameters using the chain rule:

$$\frac{d\mathcal{L}}{dh_l} = \left(\frac{dh_{l+1}}{dh_l}\right)^T \frac{d\mathcal{L}}{dh_{l+1}}, \quad \frac{dh_{l+1}}{dh_l} = W_{l+1}\text{diag}(\sigma'_{l+1}(W_{l+1}h_l + b_{l+1})), \quad \frac{d\mathcal{L}}{d\theta_l} = \left(\frac{dh_l}{d\theta_l}\right)^T \frac{d\mathcal{L}}{dh_l}.$$

Thus, gradients are obtained by first propagating activations forward to the output layer and then recursively applying these equations. These equations imply that gradients are propagated backwards through the network using weights symmetric to their feedforward counterparts. This is biologically problematic because it implies a mode of information propagation (error propagation) that does not influence neural activity, and that depends on an implausible network architecture (symmetric weight connectivity for feedforward and feedback directions, which is called the weight transport problem).

2.1.1 Target propagation and its variants

In target propagation (LeCun, 1986; 1987; Bengio, 2014; Lee et al., 2015) backwards communication induces neural activity, unlike in BP where backwards communication passes on gradients without inducing neural activity. The induced activities are those that layers should strive to match so as to produce the target output. After feedforward propagation given some input, the final output layer h_L is trained directly to minimize the loss $\mathcal{L}$, while all other layers are trained so as to match their associated targets.

In general, good targets are those that minimize the loss computed in the output layer if they were actually realized in feedforward propagation. In networks with invertible layers one could generate such targets by first finding a loss-optimal output activation $\hat{h}_L$ (e.g. the correct label distribution) and then propagating it back using inverse transformations $\hat{h}_l = f^{-1}(\hat{h}_{l+1}; \theta_{l+1})$. Since it is hard to maintain invertibility in a network, approximate inverse transformations (or decoders) can be learned $g(h_{l+1}; \lambda_{l+1}) \approx f^{-1}(h_{l+1}; \theta_{l+1})$. Note that this learning obviates the need for symmetric weight connectivity.

The generic form of target propagation algorithms we consider in this paper can be summarized as a scheduled minimization of two kinds of losses for each layer:

1. *Reconstruction* or *inverse loss* $\mathcal{L}_l^{inv}(\lambda_l) = \|h_{l-1} - g(f(h_{l-1}; \theta_{l-1}); \lambda_l)\|_2^2$ used to train the approximate inverse that is parametrized similarly to the forward computation $g(h_l; \lambda_l) = \sigma_l(V_l h_l + c_l), \lambda_l = \{V_l, c_l\}$, where activations h_{l-1} are assumed to be propagated from the input. One can imagine other learning rules for the inverse, for example, the original DTP algorithm trained inverses on noise-corrupted versions of activations with the purpose of improved generalization. In our implementation we instead used the *denoising criterion* which we find more biologically plausible, see the appendix for details. The loss is applied for every layer except the first, since the first layer does not need to propagate target inverses backwards.
2. *Learning loss* $\mathcal{L}_l(\theta_l) = \|f(h_l; \theta_l) - \hat{h}_{l+1}\|_2^2$ penalizes the layer parameters for producing activations different from their targets. Parameters of the last layer are trained to minimize the task's loss $\mathcal{L}$ directly.

Under this framework both losses are local and involve only single layer's parameters, and implicit dependencies on other layer's parameters are ignored. Variants differ in the way targets $\hat{h}_l$ are computed.

Target propagation "Vanilla" target propagation (TP) computes targets by propagating the higher layers' targets backwards through layer-wise inverses; i.e. $\hat{h}_l = g(\hat{h}_{l+1}; \lambda_{l+1})$. For traditional categorization tasks the same 1-hot vector in the output will always map back to precisely the same hidden unit activities in a given layer. Thus, this kind of naive TP may have difficulties when different instances of the same class have very different appearances since it will be trying to make their representations identical even in the early layers. Also, there are no guarantees about how TP will behave when the inverses are imperfect.

Difference target propagation Difference target propagation updates the output weights and biases using the standard gradient rule, but this is biologically unproblematic because it does not require weight transport (O'Reilly, 1996; Lillicrap et al., 2016). For most other layers in the network, difference target propagation (DTP) (Lee et al., 2015) computes targets as $\hat{h}_l = h_l + g(\hat{h}_{l+1}; \lambda_{l+1}) - g(h_{l+1}; \lambda_{l+1})$. The extra terms provide a stabilizing linear correction for imprecise inverse functions. However, in the original work by Lee et al. (2015) the penultimate layer target, $\hat{h}_{L-1}$, was computed using gradients from the network's loss, rather than by target propagation. That is, $\hat{h}_{L-1} = h_{L-1} - \alpha \frac{\partial \mathcal{L}(h_L)}{\partial h_{L-1}}$, rather than $\hat{h}_{L-1} = h_{L-1} - g(h_L) + g(\hat{h}_L)$. Though not stated explicitly, this approach was presumably taken to insure that the penultimate layer received reasonable and diverse targets despite the low-dimensional 1-hot targets at the output layer. When there are a small number of 1-hot targets (e.g. 10 classes), learning a good inverse mapping from these vectors back to the hidden activity of the penultimate hidden layer (e.g. 1000 units) might be problematic, since the inverse mapping cannot provide information that is both useful and unique to a particular *sample*. Using BP in the penultimate layer sidesteps this concern, but deviates from the intent of using these algorithms to avoid gradient computation and delivery.

Simplified difference target propagation We introduce SDTP as a simple modification to DTP. In SDTP we compute the target for the penultimate layer as $\hat{h}_{L-1} = h_{L-1} - g(h_L) + g(\hat{h}_L)$, where $\hat{h}_L = \text{argmin}_{h_L} \mathcal{L}(h_L)$. This completely removes biologically infeasible gradient communication (and hence weight-transport) from the algorithm. However, it is not clear whether targets for the penultimate layer will be diverse enough (given low entropy classification targets) or precise enough (given the inevitable poor performance of the learned inverse for this layer). This is a non-trivial change that requires empirical validation.

Parallel and alternating training of inverses In the original implementation of DTP[1], the authors trained forward and inverse model parameters by alternating between their optimizations; in practice they trained one loss for one full epoch of the training set before switching to training the other loss. We considered a variant that simply optimizes both losses in parallel, which seems nominally more plausible in the brain since both forward and feedback connections are thought to undergo plasticity

[1] `https://github.com/donghyunlee/dtp/blob/master/conti_dtp.py`

changes simultaneously. Though, it is possible that a kind of alternating learning schedule for forward and backward connections could be tied to wake/sleep cycles.

Noise-preserving versus de-noising autoencoder training In the original DTP algorithm, autoencoder training is done via a noise-preserving loss, which may be a principled choice for the algorithm on a computer (Lee et al., 2015). But in the brain, autoencoder training is de-noising, since uncontrolled noise is necessarily added downstream of a given layer (e.g. by subsequent spiking activity and stochastic vesicle release). Therefore, in our experiments with TP we use de-noising autoencoder training. We also compared noise-preserving and de-noising losses in the context of DTP and SDTP and found that they performed roughly equivalently (see Appendix 4).

Algorithm 1 Simplified Difference Target Propagation

Propagate activity forward:
for $l = 1$ **to** L **do**
  $h_l \leftarrow f_l(h_{l-1}; \theta_l)$
end for
Compute first target: $\hat{h}_L \leftarrow \text{argmin}_{h_L} \mathcal{L}(h_L)$
Compute targets for lower layers:
for $l = L - 1$ **to** 1 **do**
  $\hat{h}_l \leftarrow h_l - g(h_{l+1}; \lambda_{l+1}) + g(\hat{h}_{l+1}; \lambda_{l+1})$
end for
Train inverse function parameters:
for $l = L$ **to** 2 **do**
  Update parameters λ_l using SGD on loss $\mathcal{L}_l^{inv}(\lambda_l)$
  $\mathcal{L}_l^{inv}(\lambda_l) = \|h_{l-1} - g(f(h_{l-1} + \epsilon; \theta_{l-1}); \lambda_l)\|_2^2, \epsilon \sim \mathcal{N}(0, \sigma^2)$
end for
Train feedforward function parameters:
for $l = 1$ **to** L **do**
  Update parameters θ_l using SGD on loss $\mathcal{L}_l(\theta_l)$
  $\mathcal{L}_l(\theta_l) = \|f(h_l; \theta_l) - \hat{h}_{l+1}\|_2^2$ if $l < L$, else $\mathcal{L}_L(\theta_L) = \mathcal{L}$ (task loss)
end for

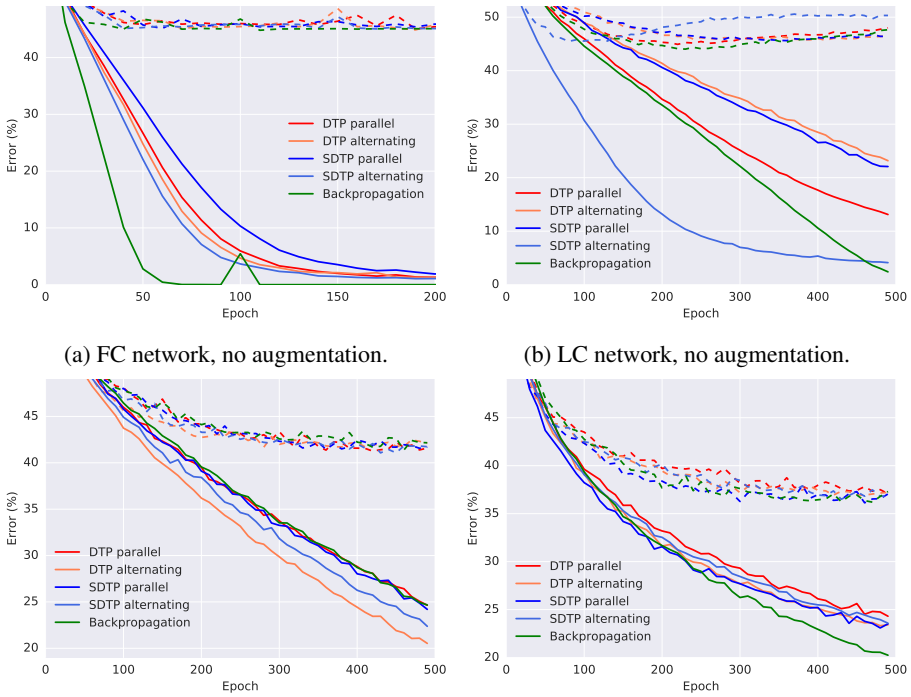
2.2 Biologically-plausible network architectures

Convolution-based architectures are critical for achieving state of the art in image recognition (Krizhevsky et al., 2012). These architectures are biologically implausible, however, because of their extensive weight sharing. To implement convolutions in biology, many neurons would need to share the values of their weights precisely, which is unlikely. In the absence of weight sharing, the "locally connected" receptive field structure of convolutional neural networks is in fact very biologically realistic and may still offer a useful prior. Under this prior, neurons in the brain could sample from small areas of visual space, then pool together to create spatial maps of feature detectors. We assess the the degree to which BP-guided learning is enhanced by convolutions, and not BP *per se*, by evaluating learning methods (including BP) on networks with locally connected layers.

3 Experiments

Since the purpose of our study was not to establish state of the art results, but rather to assess the limitations of biologically-motivated learning methods, we focused on evaluating architectures that were considered reasonable for a particular task or dataset. Thus, we did not perform an exhaustive architecture search beyond adjusting total number of training parameters to prevent overfitting. All experiments share the same straightforward methodology: a hyperparameter search was performed for a fixed architecture, for each learning algorithm. We then selected the best run from each hyperparameter search based on validation set accuracy across 5 consecutive training epochs (i.e. passes over training set) at the end of which we also measured accuracy on the test set.

All locally-connected architectures consist of a stack of locally-connected layers specified as (receptive field size, number of output channels, stride, padding) followed by an output softmax layer. For padding, SAME denotes padding with zeros to ensure unchanged shape of the output with stride $= 1$ and VALID padding denotes no padding. For optimization we use Adam (Kingma & Ba, 2014),



(a) FC network, no augmentation.

(b) LC network, no augmentation.

(c) FC network, augmented data.

(d) LC network, augmented data.

Figure 2: Classification error on CIFAR dataset. Train and test errors are drawn using solid and dashed lines correspondingly.

Table 1: Final errors (%) for different network architectures and training methods on MNIST dataset.

	Fully-connected network		Locally-connected network	
Method	Train	Test	Train	Test
DTP, parallel	0.06	1.77	0.15	1.48
DTP, alternating	0.07	1.85	0.15	1.48
SDTP, parallel	0.06	2.03	0.10	1.49
SDTP, alternating	0.05	1.88	0.05	1.63
Backpropagation	0.00	1.96	0.00	1.46

with different hyper-parameters for forward and inverse models in the case of target propagation. All layers are initialized using the method of Glorot & Bengio (2010). In all networks we used the hyperbolic tangent as a nonlinearity between layers as it was previously found to work better with DTP than ReLUs (Lee et al., 2015).

3.1 MNIST

To compare to previously reported results we began with the MNIST dataset, consisting of 60000 train and 10000 test 28×28 gray-scale images of hand-drawn digits, with 10000 images from the train test reserved for validation.

For the evaluation of fully-connected architectures we chose a network from the original DTP paper (Lee et al., 2015), consisting of 7 hidden layers with 240 units per layer. While 7 hidden layers provide arguably excessive capacity for this task, this setup is well-suited for understanding how suitable the considered methods are for learning in relatively deep networks which are known to be prone to exploding or vanishing learning signals.

Table 2: Final errors (%) for different network architectures and training methods on CIFAR dataset. FC and LC stand for fully-connected and locally-connected networks correspondingly.

	No augmentation				Augmented data			
	FC		LC		FC		LC	
Method	Train	Test	Train	Test	Train	Test	Train	Test
DTP, parallel	1.13	44.97	30.14	44.83	24.73	41.39	25.58	37.62
DTP, alternating	1.14	44.67	29.90	45.99	25.41	42.16	23.49	37.35
SDTP, parallel	1.10	44.66	29.75	45.89	24.70	41.40	23.26	36.71
SDTP, alternating	1.06	44.76	29.56	45.24	26.09	41.72	23.71	36.56
Backpropagation	0.00	45.13	27.63	43.49	25.75	42.13	20.94	36.56

The locally-connected architecture consisted of 4 hidden layers and has the following structure: (3×3, 32, 2, SAME), (3×3, 16, 1, SAME), (3×3, 16, 1, SAME), (3×3, 10, 1, VALID).

Results are reported in table 1 and the learning dynamics is plotted on figure 4. Quite surprisingly, SDTP performed competitively with respect to both DTP and BP, even though it didn't use gradient propagation to assign targets for the penultimate hidden layer. This suggests that, at least for relatively simple learning tasks, the problem of finite number of targets may not be as serious as one might expect.

Locally connected architectures performed well with all variants of target propagation, and about as well as with BP. Still, the resulting test accuracy did not match previous known results obtained with convolutional networks, which can produce less than 1% test error, see, e.g. (LeCun et al., 1998). However, the observed improvement in generalization in our experiments must have been solely caused by locally-connected layers, as none of the fully-connected networks with smaller number of hidden layers (and hence with less excessive numbers of parameters) performed similarly.

We noticed that target propagation showed noisier and slower learning comparing to BP (see Figure 2). Yet, with early stopping and hyper-parameter tuning it performed competitively. One can also see that with a fully-connected architecture BP achieved worse test error selected by our methodology. This is likely explained by the fact that BP overfits to the training set faster (in contrast, none of target propagation variants achieved 0% train error). These same phenomena were also observed in the locally-connected network.

3.2 CIFAR-10

CIFAR-10 is a more challenging dataset introduced by Krizhevsky (2009). It consists of 32×32 RGB images of 10 categories of objects in natural scenes, split into 50000 train and 10000 test images, where we also reserve 10000 train images for validation. In contrast to MNIST, classes in CIFAR-10 do not have a "canonical appearance" such as a "prototypical bird" or "prototypical truck" as opposed to "prototypical 7" or "prototypical 9". This makes them harder to classify with simple template matching, making depth imperative for achieving good performance. To our best knowledge, this is the first empirical study of biologically-motivated learning methods without weight transport on this dataset.

We considered a fully-connected network with 3 hidden layers of 1024 units and a 5-layer network with locally-connected layers having the following structure: (3×3, 32, 2, SAME), (3×3, 32, 2, SAME), (3×3, 16, 1, SAME), (3×3, 16, 2, SAME), (1×1, 10, 1, SAME).

Final results can be found in table 2. One can see that with even on a more complex dataset different TP variants, including the most biologically-feasible SDTP performed similarly to BP. Clearly, the data augmentation employed (random crops and left-right flips) has been necessary for the locally-connected network to demonstrate a significant improvement over the fully-connected network, otherwise LC models begin to overfit (see fig. 2b). At the same time, convolutional analog of the LC network has achieved 31.23% and 34.37% of train and test error correspondingly, without use of data augmentation. This quantitatively demonstrates the need of further advances in biologically-plausible architectures in order to match performance of modern convolutional networks.

3.3 IMAGENET

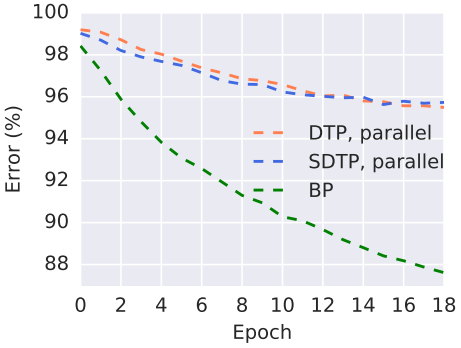


Figure 3: Top-1 test error on ImageNet.

METHOD	TOP-1 TEST ERROR
DTP, PARALLEL	95.49
SDTP, PARALLEL	95.63
BACKPROPAGATION	87.62

Table 3: Top-1 test error on ImageNet after 18 epochs.

Finally, we assessed performance of the methods on the ImageNet dataset (Russakovsky et al., 2015), a large-scale benchmark that has propelled recent progress in deep learning. Again, to the best of our knowledge, this is the first empirical study of biologically-motivated methods and architectures conducted on a dataset of such scale and difficulty. ImageNet consists of 1271167 training examples from which 10000 were reserved for validation and 50000 for testing. It has 1000 object classes appearing in a variety of natural scenes and captured in high-resolution images (resized to 224×224).

The locally-connected architecture we considered for this experiment was inspired by the ImageNet architecture used in (Springenberg et al., 2014). Unfortunately, the naive replacement of convolutional layers with locally-connected layers would result into a computationally-prohibitive architecture, so we decreased number of output channels in the layers and also removed layers with 1×1 filters. We also slightly decreased filters in the first layer, from 11×11 to 9×9. The resulting network had the following architecture: (9×9, 48, 4, SAME), pooling, (5×5, 64, 1, SAME), pooling, (3×3, 96, 1, SAME), pooling, (3×3, 128, 1, SAME), spatial 6×6 average. Here every pooling layer is an average pooling with 3×3 receptive field. See the appendix for details of implementing locally-connected networks.

To further reduce the amount of required computation, we included only parallel variants of DTP and SDTP in the evaluation, as these methods are more representative of the biological constraints, and are more straightforward to implement given the size of this dataset's epochs. Models were trained for 5 days, resulting in 18 passes over training set.

The final results can be found in table 3. Unlike on MNIST and CIFAR, on ImageNet all variants performed quite poorly. Additionally, it is on this dataset where we first observed a striking difference between BP and the TP variants. A number of factors could contribute to this result. One factor may be that deeper networks might require more careful hyperparameter tuning when using TP; for example, different learning rates or amount of noise injected for each layer. A second factor may be the difficulty with learning in the output layer, where a 1000-dimensional vector is predicted from just a 128-dimensional output from the final spatial average layer. Moreover, the inverse computation involves non-compressing learning, which has not been well studied in the context of TP. Unfortunately, preserving the original 1920 channels in the layer certainly presents a computational challenge. Addressing both of these factors could help improve performance, so it would be untimely to conclude on any principal inefficiencies of TP. Therefore, we leave the challenge of matching performance of BP on ImageNet to the future work.

4 DISCUSSION

Historically, there has been significant disagreement about whether BP can tell us anything interesting about learning in the brain (Crick, 1989; Grossberg, 1987). Indeed, from the mid 1990s to 2010, work on applying BP to the brain all but disappeared. Recent progress in machine learning has prompted a revival of this debate; where other approaches have failed, deep networks trained via BP have been key to achieving impressive performance on difficult datasets such as ImageNet. It is once again natural to wonder whether some approximation of BP might underlie learning in the brain.

However, none of the algorithms proposed as approximations of BP have been tested on the datasets that were instrumental in convincing the machine learning and neuroscience communities to revisit these questions.

Here we introduced a straightforward variant of difference target-propogation that completely removed gradient propagation and weight transport and tested it on the challenging task of classifying CIFAR and ImageNet images. We also investigated and reported results on the use of local connectivity. We demonstrated that networks trained with SDTP without any weight sharing (i.e. weight transport in the backward pass or weight tying in convolutions) are generally able to compete with those trained with BP on difficult tasks such as CIFAR. However, BP significantly outperforms both DTP and SDTP on ImageNet, and more work is required to understand why this issue arises at scale.

We note that although activity-propagation-based algorithms go a long way towards biological plausibility, there are still many biological constraints that we did not address here. For example, we've set aside the question of spiking neurons entirely to focus on asking whether variants of TP can scale up to solve difficult problems *at all*. The question of spiking networks is an important one (Samadi et al., 2017; Guergiuev et al., 2016a), but it is nevertheless possible to gain algorithmic insight to the brain without tackling all of the elements of biological complexity simultaneously. Similarly, we also ignore Dale's law in all of our experiments (Parisien et al., 2008). In general, we've aimed at the simplest models that allow us to address questions around (1) *weight sharing*, and (2) *the form and function of feedback communication*.

Algorithms that contend for a role in helping us understand learning in cortex should be able to perform well on difficult domains without relying on weight transport or tying. Thus, our results offer a new benchmark for future work looking to evaluate the effectiveness of potential biologically plausible algorithms in more powerful architectures and on more difficult datasets.

REFERENCES

David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines. *Cognitive science*, 9(1):147–169, 1985.

Luis B Almeida. A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In *Artificial neural networks*, pp. 102–111. IEEE Press, 1990.

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

Yoshua Bengio, Dong-Hyun Lee, Jorg Bornschein, Thomas Mesnard, and Zhouhan Lin. Towards biologically plausible deep learning. *arXiv preprint arXiv:1502.04156*, 2015.

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

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

Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249–256, 2010.

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

Jordan Guergiuev, Timothy P Lillicrap, and Blake A Richards. Deep learning with segregated dendrites. *arXiv preprint arXiv:1610.00161*, 2016a.

Jordan Guergiuev, Timothy P Lillicrap, and Blake A Richards. Towards deep learning with segregated dendrites. *arXiv preprint arXiv:1610.00161*, 2016b.

G.E. Hinton. How to do backpropagation in a brain. *NIPS 2007 Deep Learning Workshop*, 2007.

Geoffrey E Hinton and James L McClelland. Learning representations by recirculation. In *Neural information processing systems*, pp. 358–366. New York: American Institute of Physics, 1988.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Konrad P Körding and Peter König. Supervised and unsupervised learning with two sites of synaptic integration. *Journal of computational neuroscience*, 11(3):207–215, 2001.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Yann LeCun. Learning process in an asymmetric threshold network. In *Disordered systems and biological organization*, pp. 233–240. Springer, 1986.

Yann LeCun. *Modèles connexionnistes de lapprentissage*. PhD thesis, PhD thesis, These de Doctorat, Université Paris 6, 1987.

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

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

Timothy P Lillicrap, Daniel Cownden, Douglas B Tweed, and Colin J Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 7, 2016.

Javier R Movellan. Contrastive hebbian learning in the continuous hopfield model. In *Connectionist models: Proceedings of the 1990 summer school*, pp. 10–17, 1991.

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

Christopher Parisien, Charles H Anderson, and Chris Eliasmith. Solving the problem of negative synaptic weights in cortical models. *Neural computation*, 20(6):1473–1494, 2008.

Fernando J Pineda. Generalization of back-propagation to recurrent neural networks. *Physical review letters*, 59(19):2229, 1987.

Fernando J Pineda. Dynamics and architecture for neural computation. *Journal of Complexity*, 4(3): 216–245, 1988.

DE Rumelhart, GE Hinton, and RJ Williams. Learning representations by back-propagation errors. *Nature*, 323:533–536, 1986.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115 (3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Arash Samadi, Timothy P Lillicrap, and Douglas B Tweed. Deep learning with dynamic spiking neurons and fixed feedback weights. *Neural computation*, 2017.

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

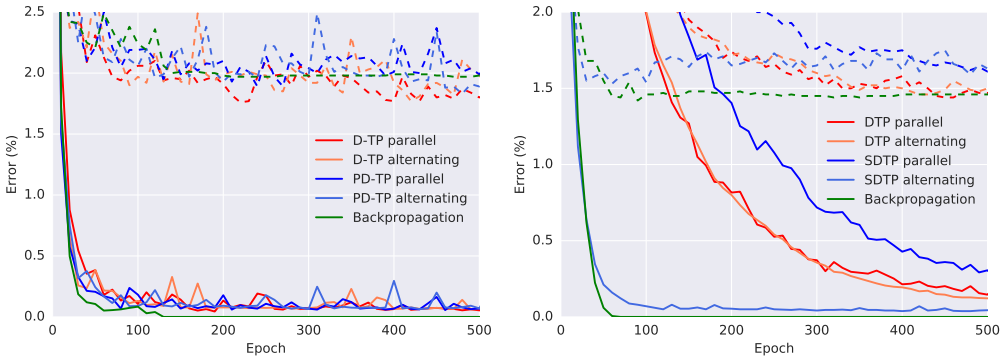
Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

James CR Whittington and Rafal Bogacz. An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural computation*, 2017.

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

APPENDIX

EXTRA FIGURES



(a) Fully-connected network.

(b) Locally-connected network.

Figure 4: Classification error on MNIST dataset. Train and test errors are drawn using solid and dashed lines correspondingly.

IMPLEMENTATION DETAILS FOR LOCALLY-CONNECTED ARCHITECTURES

Although locally-connected layers can be seen as a simple generalization of convolution layers, their implementation is not entirely straightforward. First, a locally-connected layer has many more trainable parameters than a convolutional layer with an equivalent specification (i.e. receptive field size, stride and number of output channels). This means that a simple replacement of every convolutional layer with a locally-connected layer can be computationally prohibitive for larger networks. Thus, one has to decrease the number of parameters in some way to run experiments using a reasonable amount of memory and compute. In our experiments we opted to decrease the number of output channels in each layer by a given factor. Obviously, this can have a negative effect on the resulting performance and more work needs to be done to scale locally-connected architectures.

Inverse operations When training locally-connected layers with target propagation, one also needs to implement the inverse computation in order to train the feedback weights. As in fully-connected layers, the forward computation implemented by both locally-connected and convolutional layers can be seen as a linear transformation $y = Wx + b$, where the matrix W has a special, sparse structure (i.e., has a block of non-zero elements, and zero-elements elsewhere), and the dimensionality of y is not more than x.

The inverse operation requires computation of the form $x = Vy + c$, where matrix V has a similar sparse structure as W^T. However, given this sparsity of V, computing the inverse of y using V would be highly inefficient (Dumoulin & Visin, 2016). We instead use an implementation trick often used in deconvolutional architectures. First, we define a forward computation $z = Ax$, where z and A are dummy activities and weights. We then define a transpose matrix as the *gradient* of this feedforward operation:

$$V = A^T = \left(\frac{dz}{dx}\right)^T,$$

and thus

$$x = Vy + c = \left(\frac{dz}{dx}\right)^T y + c.$$

The gradient $\frac{dz}{dx}$ (and its multiplication with y) can be very quickly computed by the means of automatic differentiation in many popular deep learning frameworks. Note that this is strictly an implementation detail and does not introduce any additional use of gradients or weight sharing in learning.

Table 4: Hyperparameter search space used for the experiments

HYPERPARAMETER	SEARCH DOMAIN
Learning rate of model Adam optimizer	$[10^{-5}; 3 \times 10^{-4}]$
β_1 parameter of model Adam optimizer	Fixed to 0.9
β_2 parameter of model Adam optimizer	$\{0.99, 0.999\}$
ϵ parameter of model Adam optimizer	$\{10^{-4}, 10^{-6}, 10^{-8}\}$
Learning rate of inverse Adam optimizer	$[10^{-5}; 3 \times 10^{-4}]$
β_1 parameter of inverse Adam optimizer	Fixed to 0.9
β_2 parameter of inverse Adam optimizer	$\{0.99, 0.999\}$
ϵ parameter of inverse Adam optimizer	$\{10^{-4}, 10^{-6}, 10^{-8}\}$
Learning rate α used to compute targets for h_{L-1} in DTP	$[0.01; 0.2]$
Gaussian noise magnitude σ used to train inverses	$[0.01; 0.3]$

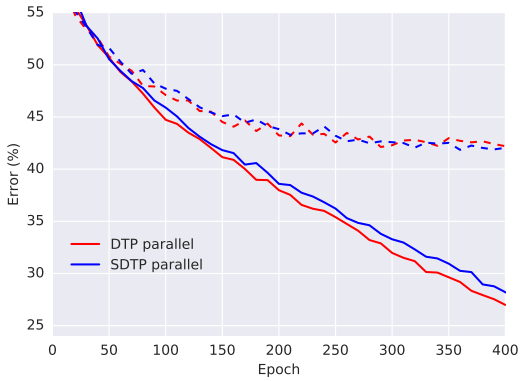


Figure 5: Train and test errors (%) for a FC network trained with noise-preserving inverse loss on augmented CIFAR dataset.

DETAILS OF HYPERPARAMETER OPTIMIZATION

For DTP and SDTP we optimized over parameters of the model and inverse Adam optimizers, learning rate α used to compute targets for h_{L-1} in DTP, and the Gaussian noise magnitude σ used to train inverses. For backprop we optimized only the model Adam optimizer parameters. For all experiments the best hyperparameters were found by random searches over 60 random configurations drawn from the relevant ranges specified in table 4.

DENOISING VS NOISE-PRESERVING TRAINING OF INVERSES

As we mention in section 2.1.1, in our implementation of TP algorithms we use denoising training of model inverses which we find more biologically motivated than noise-preserving training used by Lee et al. (2015). In particular, because downstream activity will always have noise applied to it (e.g., given that downstream neurons spike stochastically), one is always fundamentally in the denoising case in the brain.

We did not observe a significant empirical difference between these two methods in practice for either DTP and SDTP. Figure 5 shows the learning dynamics for parallel versions of DTP and SDTP with noise-preserving inverse losses to compare with figure 2c with denoising inverse loss. One can see that the considered methods converge to roughly same train and test errors with similar speed.