Training Neural Networks with Local Error Signals

Arild Nøkland [*1] **Lars H. Eidnes** [*2]

Abstract

Supervised training of neural networks for classification is typically performed with a global loss function. The loss function provides a gradient for the output layer, and this gradient is back-propagated to hidden layers to dictate an update direction for the weights. An alternative approach is to train the network with layer-wise loss functions. In this paper we demonstrate, for the first time, that layer-wise training can approach the state-of-the-art on a variety of image datasets. We use single-layer sub-networks and two different supervised loss functions to generate local error signals for the hidden layers, and we show that the combination of these losses help with optimization in the context of local learning. Using local errors could be a step towards more biologically plausible deep learning because the global error does not have to be transported back to hidden layers. A completely backprop free variant outperforms previously reported results among methods aiming for higher biological plausibility. Code is available.[1]

1. Introduction

Neural networks for classification are typically trained with a global cross-entropy loss, with the prediction error being back-propagated layer-by-layer from the output layer to hidden layers (D. E. Rumelhart, 1986). The hidden layer weights cannot be updated before the forward and backward pass has completed. This backward locking prevents parallelization of the weight updates. It also prevents reuse of the memory used to store hidden layer activations. Several methods have been proposed to avoid the backward locking and memory reuse problems (Jaderberg et al., 2017; Gomez et al., 2017).

Back-propagation of global errors is not biologically plausible for a number of reasons (Bengio et al., 2015). Several more realistic alternatives have been proposed (Bengio, 2014; Lee et al., 2015b; Lillicrap et al., 2014; Nøkland, 2016; Scellier & Bengio, 2017). These methods do not seem to scale to larger and more complicated problems like CIFAR10 and ImageNet (Bartunov et al., 2018).

In this paper, we demonstrate that the backward-locking problem can be avoided by layer-wise training of the hidden layers with locally generated errors. The local loss functions do not depend on a globally generated error, the gradient is not backpropagated to previous layers, and the hidden layer weights can be updated during the forward pass. At the inference stage, the network behaves as a standard network trained with global back-prop. When the weights for a hidden layer have been updated, the gradient and the activations do not have to be kept in memory any more. This alleviates the memory requirements when training deep networks. Although we train all layers at the same time, locally generated errors also enables greedily training layers one-at-a-time, which can reduce the memory footprint even more, and also reduce the training time.

Using local errors could be a step towards more biologically plausible deep learning because the global error does not have to be propagated back to hidden layers. The global target can be projected back to hidden layers instead.

Despite the promise that local loss functions can make training faster, more memory efficient, more parallel and more biologically plausible, layer-wise supervised training has been poorly explored in the literature.

2. Related work

2.1. Local Loss Functions

Local loss functions have been used to pre-train hidden layers independently of the global loss, and this has in certain cases been shown to improve the performance after fine-tuning using global back-propagation (Hinton et al., 2006; Bengio et al., 2007; Salakhutdinov & Hinton, 2009; Erhan et al., 2010; Vincent et al., 2008; Paine et al., 2014; Dong

[*]Equal contribution [1]Kongsberg Seatex, Trondheim, Norway [2]Trondheim, Norway. Correspondence to: Arild Nøkland <arild.nokland@gmail.com>, Lars H. Eidnes <larseidnes@gmail.com>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

[1]The code for the experiments is available at https://github.com/anokland/local-loss

et al., 2018). Local loss functions have been used as an auxillary objective to improve performance (Lee et al., 2015a; Zhang et al., 2016; Szegedy et al., 2015; Wang et al., 2015; Weston et al., 2012). Using supervised layer-wise loss functions, without fine-tuning, has also been explored previously (Mostafa et al., 2017; Malach & Shalev-Shwartz, 2018; Marquez et al., 2018). The best reported result on CIFAR-10 is 7.2% using local classifiers and ensembling, approaching the results of global backprop (Belilovsky et al., 2018). Our contribution is to show that local classifiers combined with a local similarity matching loss can match global backprop in terms of test error.

Training hidden layers with synthetic gradients is another way to avoid the backward locking problem (Jaderberg et al., 2017). This method uses local loss functions to train subnetworks to approximate the true gradient. The synthetic gradient modules are trained with an L2 loss to predict the true gradient from the layer above. The input to the module is the hidden layer activation and in some cases, the target vector. The method relates to layer-wise supervised training because the target information is used to train hidden layers. The method differs from our approach because we don't try to approximate a back-propagated gradient, instead we utilize the target vector to create an error signal independently of the layers above.

2.2. Similarity Measures in Neuroscience

Similarity measures have been used in the neuroscience field to characterize neural activity patterns. Representational similarity analysis (RSA) measures similarity of representations under different experimental conditions (Kriegeskorte et al., 2008a). By comparing the activity associated with each pair of experimental conditions one can obtain a representational dissimilarity matrix (RDM), much like the similarity matrix we use in this work. For instance, RSA performed on recordings from the inferior temporal (IT) cortex in monkeys show that neural responses to images are clustered according to object categories (Kiani et al., 2007). The category clusters surprisingly match between humans and monkeys when exposed to the same real-world object images (Kriegeskorte et al., 2008b).

2.3. Similarity Measures in Machine Learning

The similarity matching loss function in this paper can be related to previous work in unsupervised clustering and feature learning. Suppose we have n datapoints as the columns of matrix $H = (\mathbf{h}_1, \ldots, \mathbf{h}_n)$. A decomposition of this matrix can be expressed as follows:

$$\min_{C,G} \|H - CG\|_F^2 \tag{1}$$

If we enforce orthogonality on C, requiring $C^TC = I$, the minimization in (1) implements the subspace version of principal component analysis (PCA). If an L1 penalty is placed on G, the minimization performs sparse coding. Under the constraint that the n columns of G are one-hot cluster selectors, solving this minimization finds a k-means clustering of the data, where the k columns of C are the cluster centroids. Under the constraint that $H, C, G \geq 0$, the minimization finds a non-negative matrix factorization (NMF) of the data. Each of these methods have been used for unsupervised feature learning in computer vision (Coates & Ng, 2011; Raina et al., 2007; Lee & Seung, 1999).

Given some self-similarity measure $S(.)$, consider the objective:

$$\min_{G} \|S(H) - S(G)\|_F^2 \tag{2}$$

If $S(.)$ measures the euclidian distance between data points, this minimization implements multidimensional scaling (MDS). If we define $S(X) = X^TX$ and constrain G to be ≥ 0, the minimization in (2) finds what is called the symmetric NMF (Kuang et al., 2012). If instead of non-negativity we enforce orthogonality, requiring $GG^T = I$, the minimization in (2) implements (Ding et al., 2005; Kuang et al., 2012) the family of methods called spectral clustering (Ng et al., 2002). By choosing different similarity measures $S(.)$ in the first term of (2), this minimization can perform different graph clustering objectives, namely ratio association, kernel clustering and normalized cuts (Kuang et al., 2012).

The above are *unsupervised* clustering and feature learning methods. For our purposes in this paper, we use what can be seen as a *supervised* clustering loss, where two data points belong to the same cluster if they have the same label. Given a label matrix $Y = (\mathbf{y_1}, \ldots, \mathbf{y_n})$, whose columns are the one-hot encoded labels of the data, we minimize:

$$\min_{\theta} \|S(NeuralNet(H;\theta)) - S(Y)\|_F^2 \tag{3}$$

Here the matrix Y is fixed, and instead the parameters θ of $NeuralNet$ are adjusted to minimize the loss. A straightforward interpretation of this loss is that it encourages the neural network to learn representations of the data such that distinct classes have distinct representations.

This supervised clustering loss is related to methods like linear disciminative analysis (LDA) (Fisher, 1936), and neighbourhood component analysis (NCA) (Goldberger et al., 2005) because they both utilize label information to perform clustering.

3. Method

We use standard convolutional and fully connected network architectures, but instead of globally back-propagating errors, each weight layer is trained by a local learning signal, that is not back-propagated down the network. The learning signal is provided by two separate single-layer sub-networks,

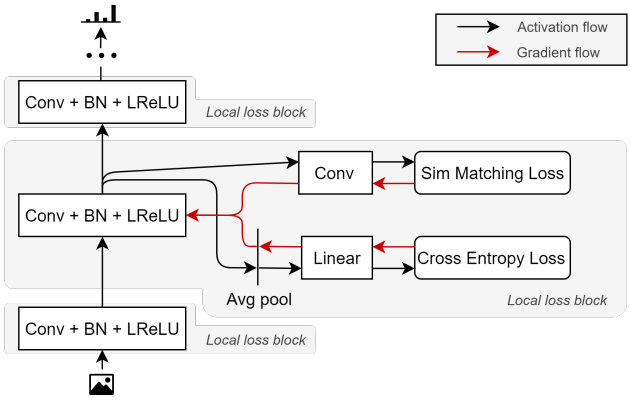

Figure 1. With the local loss functions, each weight layer is trained by two single-layer sub-networks, each with their own distinct loss function.

each with their own distinct loss function. One sub-network is trained with a standard cross-entropy loss, and the other with a similarity matching loss (see Figure 1).

3.1. Similarity Matching Loss

The similarity matching loss measures the L2 distance between matrices, where the elements contain the pair-wise similarities between examples in a mini-batch. We denote this loss as L_{sim} or **sim** loss. Given a mini-batch of hidden layer activations $H = (\mathbf{h}_1, \ldots, \mathbf{h}_n)$, and a one-hot encoded label matrix $Y = (\mathbf{y}_1, \ldots, \mathbf{y}_n)$, we have:

$$L_{sim} = \|S(NeuralNet(H)) - S(Y)\|_F^2 \tag{4}$$

When H is the output of a linear layer, $NeuralNet(.)$ is a linear layer. When H is the output of a convolutional layer, $NeuralNet(.)$ is a convolutional layer with kernel size 3x3, stride 1 and padding 1, followed by a standard deviation operation over each feature map. This operation will reduce the dimension of the output to 2. $S(X)$ is the adjusted cosine similarity matrix, or correlation matrix, of a mini-batch X. $S(X)$ contains elements s_{ij} where

$$s_{ij} = s_{ji} = \frac{\widetilde{\mathbf{x}}_i^T \widetilde{\mathbf{x}}_j}{\|\widetilde{\mathbf{x}}_i\|_2 \|\widetilde{\mathbf{x}}_j\|_2} \tag{5}$$

Subscripts i and j denote indices in the mini-batch. $\widetilde{\mathbf{x}}_\mathbf{i}$ denotes the mean-centered vector $\mathbf{x}_\mathbf{i}$.

3.2. Prediction Loss

The prediction loss measures the cross-entropy between a prediction from a local classifier and the target. We denote this loss as L_{pred} or **pred** loss.

The **pred** loss for a matrix H of hidden layer-activations:

$$L_{pred} = CrossEntropy(Y, W^T H) \tag{6}$$

where W is a weight matrix with width equal to the number of classes and height equal to hidden dimension, and Y is matrix of one-hot encoded targets. If H is the output of a convolutional layer, average-pooling is performed first to reduce the size, then the feature maps are flattened.

3.3. Backprop Free Version

A more biologically plausible version of the similarity matching loss is to replace $NeuralNet(.)$ in (4) with a standard deviation operation over each feature map, and apply the similarity matching objective directly on these features. Then no back-propagation is required to calculate the gradient for the hidden layer. To eliminate the requirement for the global target to be available at each hidden layer, the one-hot encoded target vector is replaced with a random transformation of the same target vector. We denote this loss as $L_{sim-bio}$ or **sim-bpf** loss.

The L_{pred} loss can be made more biologically plausible by using feedback alignment (Lillicrap et al., 2014) to transport the prediction error back to the hidden layer. To eliminate the requirement for the global target to be available at each hidden layer, the classifier is trained to predict a binarized random transformation of the target vector using a binary cross-entropy loss. We denote this loss as $L_{pred-bpf}$ or **pred-bpf** loss.

The experiment section includes one experiment on CIFAR-10 using these two versions, and their combination.

3.4. Combined Loss

We denote the weighted combination of the above loss functions as $L_{predsim}$ or simply as **predsim** loss.

$$L_{predsim} = (1 - \beta)L_{pred} + \beta L_{sim} \tag{7}$$

And equally for the more biologically plausible loss functions. We denote this loss as $L_{predsim-bpf}$ or simply as **predsim-bpf** loss.

$$L_{predsim-bpf} = (1 - \beta)L_{pred-bpf} + \beta L_{sim-bpf} \tag{8}$$

4. Experiments

We performed experiments on MNIST, Fashion-MNIST, Kuzushiji-MNIST, CIFAR-10, CIFAR-100, STL-10 and SVHN to evaluate the performance of the training method. We used fully-connected architetures and two VGG-like architectures that we found to perform well (Simonyan & Zisserman, 2014). For each model we compared the performance when trained with the **glob** loss (i.e. standard global back-prop), the **pred** loss, the **sim** loss and the **predsim** loss. We also trained the best performing model with cutout regularization (Devries & Taylor, 2017), keeping all

hyper-parameters except the dropout rate identical, to see if it could improve the result further.

The architectures and hyper-parameters were chosen to give good performance for the **predsim** loss. The hyper-parameters were kept identical for all loss variations for a given dataset and architecture combination. Experiments were kept as simple as possible, and only dropout rate, learning rate, length of training, hidden layer dimension and average-pooling kernel size (used in the **pred** loss) varied across experiments.

We used two different simple VGG-like convolutional networks. Both consist of 2x2 max-pooling layers, 3x3 convolutional layers with stride 1 and padding 1, and fully connected layers. The first architecture is denoted VGG8B. The layers are **conv128-conv256-pool-conv256-conv512-pool-conv512-pool-conv512-pool-fc1024-fc**. The dimension of the output layer depends on the number of classes.

The second network is deeper and is denoted VGG11B. The layers are **conv128-conv128-conv128-conv256-pool-conv256-conv512-pool-conv512-conv512-pool-conv512-pool-fc1024-fc**. The dimension of the output layer depends on the number of classes.

The experiments were executed using the PyTorch framework. For local loss functions, the computational graph was detached after each hidden layer to prevent backward gradient flow from the output loss. The output layer was trained normally with a cross-entropy loss function.

A batch size of 128 was used in all experiments. ADAM was used for optimization (Kingma & Ba, 2014). The weighting factor β was manually tuned and set to 0.99 for all experiments with the **predsim** loss.

For networks trained with global or **pred** loss we used the ReLU non-linearity. For networks trained with **sim** or **predsim** loss we used leaky-ReLU with a negative slope of 0.01 (Maas et al., 2013) because it delivered more stable training. Before each non-linearity we applied batch normalization (Ioffe & Szegedy, 2015). After each non-linearity we applied dropout (Srivastava et al., 2014), with equal dropout rate for all layers.

The training time was 100 epochs for MNIST and Kuzushiji-MNIST, 200 epochs for Fashion-MNIST, and SVHN, and 400 epochs for the other datasets. The learning rate was multiplied by a factor of 0.25 at 50%, 75% and 89% and 94% of the total training time. Because of the high number of experiments, we performed only single-time runs. We report the test error for the last training epoch.

In some of the experiments, the number of convolutional filters were multiplied by a factor of 2 or 3. This is denoted in the tables as (2x) or (3x) trailing the network name.

Despite the large number pf parameters, we were able to train the networks on a single GPU.

4.1. MNIST

MNIST consists of hand-written digits and is the most commonly used dataset within the deep learning community. The dataset is trivial to learn and good performance here does not say much about the performance on harder tasks. We have included these experiments here for completeness.

The initial learning rate was 5e-4. The average-pooling kernel size for the **pred** loss was chosen so that the input dimension to the local classifier was 1024. The dropout rate was 0.1 for MLP and 0.2 for VGG8B. For the cutout experiment, the cutout hole size was 14.

We used 2 pixel jittering for data augmentation as done in the CapsNet paper (Sabour et al., 2017). We provide the CapsNet result here as a baseline for convolutional networks, even though better results have been achieved with ensembling and extensive data augmentation (Wan et al., 2013). We provide the performance of the Ladder network as baseline for fully-connected networks (Rasmus et al., 2015).

It is clear that jittering is helping substantially for all fully-connected networks, but the best result is with the **predsim** loss. VGG8B with **predsim** loss and cutout is on par with CapsNet.

Table 1. MNIST with 2 pixel jittering. Test error in percent.

			Local loss functions		
Model	#par	glob	pred	sim	predsim
3x1024 MLP	2.9M	0.75	0.68	0.80	**0.62**
VGG8B	7.3M	**0.26**	0.40	0.65	0.31
VGG8B+CO	7.3M	-	-	-	0.26
Ladder	-	0.57	-	-	-
CapsNet	8.2M	0.25	-	-	-

4.2. Fashion-MNIST

Fashion-MNIST is a rather new dataset with different classes of clothing and is a drop-in replacement for MNIST (Xiao et al., 2017). It is harder, but has the same size, input dimension and number of classes as MNIST.

The initial learning rate was 5e-4 for MLP and VGG8B, and 3e-4 for VGG8B(2x). The average-pooling kernel size for the **pred** loss was chosen so that the input dimension to the local classifier was 1024 for VGG8B and 2048 for VGG8B(2x). The dropout rate was 0.025 for MLP, 0.1 for VGG8B and 0.2 for VGG8b(2x). For the cutout experiment, the cutout hole size was 14.

As a baseline we show the test error for a WideResNet-28-10 (Zagoruyko & Komodakis, 2016) with and without random erasing data augmentation (Zhong et al., 2017).[2] This is to our knowledge the best published results on this dataset. Note that the baseline network has about 5 times more parameters than VGG8B. To make the comparison more fair, we also trained a version of VGG8B where the number of convolutional filters were doubled. This version performs better than the baseline, even though the number of parameters was smaller.

Table 2. Fashion-MNIST with 2 pixel jittering and horizontal flipping. Test error in percent.

Model	#par	glob	Local loss functions pred	sim	predsim
3x1024 MLP	2.9M	**8.37**	8.60	9.70	8.54
VGG8B	7.3M	**4.53**	5.66	5.12	4.65
VGG8B(2x)	28M	4.55	5.11	4.92	**4.33**
8B(2x)+CO	28M	-	-	-	4.14
WRN	37M	4.63	-	-	-
WRN+RE	37M	4.16	-	-	-

4.3. Kuzushiji-MNIST

Kuzushiji-MNIST is another drop-in replacement for MNIST containing hand-drawn japanese characters (Clanuwat et al., 2018).

The initial learning rate was 5e-4. The average-pooling kernel size for the **pred** loss was chosen so that the input dimension to the local classifier was 1024. The dropout rate was 0.2 for MLP and 0.3 for VGG8B. For the cutout experiment, the dropout rate was 0.15 and the cutout hole size was 14.

As a baseline we have included the first published results on this dataset, a PreActResNet-18 (He et al., 2016) with and without manifold mixup regularization (Clanuwat et al., 2018). VGG8B with **predsim** loss and cutout achieved a test error that surpasses the baseline, even though the number of parameters was smaller.

4.4. CIFAR-10

CIFAR-10 consist of 50000 training images of dimension 32x32 pixels (Krizhevsky, 2009). The dataset has 10 classes.

The initial learning rate was 5e-4 for MLP, VGG8B and VGG11B, and 3e-4 for VGG11B(2x) and VGG11B(3x).

[2]Due to an issue in the early version of the dataset, the baseline numbers reported in (Zhong et al., 2017) are too low, see https://github.com/zhunzhong07/Random-Erasing. We rerun the released code with the current dataset version and report the test error from the last epoch.

Table 3. Kuzushiji-MNIST with no data augmentation. Test error in percent.

Model	#par	glob	Local loss functions pred	sim	predsim
3x1024 MLP	2.9M	**5.99**	7.26	9.80	7.33
VGG8B	7.3M	1.53	2.22	2.19	**1.36**
VGG8B+CO	7.3M	-	-	-	0.99
PARN	11M	2.18	-	-	-
PARN+MM	11M	1.17	-	-	-

The average-pooling kernel size for the **pred** loss was chosen so that the input dimension to the local classifier was 2048 for VGG8B and VGG11B, and 4096 for VGG11(2x) and VGG11B(3x). The dropout rate was 0.1 for MLP, 0.2 for VGG8B and VGG11B, 0.25 for VGG11B(2x) and 0.3 for VGG11B(3x). For the cutout experiment, the cutout hole size was 16.

As a baseline we have included test error for WideResNet-40-10 with and without cutout (Devries & Taylor, 2017). Note that better results have been reported using regularization and data augmentation techniques (Verma et al., 2018; Cubuk et al., 2018). The **predsim** loss worked better than the other loss functions for the tested architectures. By multiplying the number of convolutional filters by 3, the VGG11B model trained with **predsim** loss approaches the test error of WideResNet.

Table 4. CIFAR10 with standard data augmentation. Test error in percent.

Model	#par	glob	Local loss functions pred	sim	predsim
3x3000 MLP	27M	33.6	32.3	33.5	**30.9**
VGG8B	8.9M	5.99	8.40	7.16	**5.58**
VGG11B	12M	5.56	8.39	6.70	**5.30**
VGG11B(2x)	42M	4.91	7.30	6.66	**4.42**
VGG11B(3x)	91M	5.02	7.37	9.34[3]	**3.97**
11B(3x)+CO	91M	-	-	-	3.60
WRN	56M	3.87	-	-	-
WRN+CO	56M	3.08	-	-	-

We also tested the backprop free training methods on this dataset, with a random target projection of size 128. The weighting factor β was set to 0.01 for the **predsim-bpf** loss.

The initial learning rate was 5e-4 for VGG8B, and 3e-4 for VGG8B(2x). The average-pooling kernel size for the **pred** and **sim** loss was chosen so that the flattened output dimension was 4096. The dropout rate was 0.05 for VGG8B

[3]The test error was 5.60% in epoch 399.

and 0.1 for VGG8B(2x).

The best published results on this task with no backpropagation is to our knowledge 16.9% using dense feedback alignment (Moskovitz et al., 2018). The second best result is 18.0% using K-means and SVM (Coates & Ng, 2011). If we consider sign-concordant feedback as backprop free training, the best result is 12.6% (Moskovitz et al., 2018). Our result with the **predsim-bpf** loss surpasses these results by a large margin.

Table 5. CIFAR10 with standard data augmentation. No backpropagation. Test error in percent.

Model	#par	pred-bpf	sim-bpf	predsim-bpf
VGG8B	8.9M	9.80	13.39	**9.02**
VGG8B(2x)	31M	-	-	7.80

4.5. CIFAR-100

CIFAR-100 consist of 50000 training images of dimension 32x32 pixels (Krizhevsky, 2009). The dataset has 100 classes.

The initial learning rate was 5e-4 for MLP, VGG8B and VGG11B, and 3e-4 for VGG11B(2x) and VGG11B(3x). The average-pooling kernel size for the **pred** loss was chosen so that the input dimension to the local classifier was 4096. The dropout rate was 0.025 for MLP, 0.05 for VGG8B and VGG11B, 0.1 for VGG11B(2x) and 0.15 for VGG11B(3x). We were not able to improve the test error with cutout on this dataset, so this result is excluded from the table.

Because of the high number of classes, we limited the number of classes in each mini-batch to 20 until first drop in learning rate. This was to make the target similarity matrix $S(.)$ less sparse, and we found that this improved the final result.

As a baseline we have included test error for WideResNet-40-10 with and without cutout (Devries & Taylor, 2017). Note that better results have been reported using regularization and data augmentation techniques (Yamada et al., 2018; Cubuk et al., 2018). The **predsim** loss worked better than the other loss functions for the tested architectures. By multiplying the number of convolutional filters by 3, the VGG11B model trained with **predsim** loss approaches the test error of WideResNet.

4.6. SVHN

SVHN is a dataset with house number images of dimension 32x32 pixels (Netzer et al., 2011). The training set has 73257 images and the extra training set has 531131 images. We used both training sets in our experiments. No data augmentation was used.

Table 6. CIFAR100 with standard data augmentation. Test error in percent.

Model	#par	glob	Local loss functions pred	sim	predsim
3x3000 MLP	27M	62.6	58.9	62.5	**56.9**
VGG8B	9.0M	26.2	29.3	32.6	**24.1**
VGG11B	12M	25.2	29.6	30.8	**24.1**
VGG11B(2x)	42M	23.4	26.9	28.0	**21.2**
VGG11B(3x)	91M	23.7	25.9	28.0	**20.1**
WRN	56M	18.8	-	-	-
WRN+CO	56M	18.4	-	-	-

The initial learning rate was 3e-4. The average-pooling kernel size for the **pred** loss was chosen so that the input dimension to the local classifier was 2048. The dropout rate was 0.3. For the cutout experiment, the dropout rate was 0.15 and the cutout hole size was 16.

As a baseline, we show the test error for WideResNet-16-8 with and wihout cutout (Devries & Taylor, 2017). Note that better results are reported with extensive data augmentation (Cubuk et al., 2018). The **predsim** loss clearly worked better than the other loss functions for the tested architecture, but lags behind the test error for WideResNet.

Table 7. SVHN with extra training data, but no data augmentation. Test error in percent.

Model	#par	glob	Local loss functions pred	sim	predsim
VGG8B	8.9M	2.29	2.12	1.89	**1.74**
VGG8B+CO	8.9M	-	-	-	1.65
WRN	11M	1.60	-	-	-
WRN+CO	11M	1.30	-	-	-

4.7. STL-10

STL-10 is a dataset of images belonging to 10 classes (Coates et al., 2011). The image dimension is 96x96 pixels. The training dataset consists of 5000 labeled images and a lot of unlabeled images. We used only labeled images, and we did not use the prescribed testing protocol, we just trained one model on all training examples. No data augmentation was used, making this a difficult task because of the small amount of training data.

The network architecture here was identical to the earlier description, except that the first convolutional layer was replaced with a 7x7 kernel layer with stride 2. This was to reduce the feature map size early in the network.

The initial learning rate was 5e-4. The average-pooling kernel size for the **pred** loss was chosen so that the input dimension to the local classifier was 2048. The dropout rate was 0.1. For the cutout experiment, the cutout hole size was 48.

As a baseline we show the results for WideResNet-16-8 with and without cutout regularization (Devries & Taylor, 2017). The authors use the same training and testing protocol as used here. Our result with **predsim** loss is better than the baseline.

Table 8. STL-10 with no data augmentation. Test error in percent.

			Local loss functions		
Model	#par	glob	pred	sim	predsim
VGG8B	12M	33.08	26.83	23.15	**20.51**
VGG8B+CO	12M	-	-	-	19.25
WRN	11M	23.48	-	-	-
WRN+CO	11M	20.77	-	-	-

5. Discussion

5.1. Results with Local Loss Functions

Our first observation is that the local prediction loss (**pred**) achieved test errors close to those of global back-propagation. This is in line with previous work (Mostafa et al., 2017; Belilovsky et al., 2018), but still interesting because hidden layers are decoupled from the layers above during training.

A surprising observation is that the local similarity matching loss (**sim**) is able to provide a remarkable good training signal for hidden layers. In some cases, the test error is lower than for a back-propagated global cross-entropy loss. The loss encourages examples from distinct classes to have distinct representations, measured by the cosine similarity. This can be seen as a kind of supervised clustering. This objective is sufficient to create a hidden representation that is suitable for classification, independently of the layers above.

The results indicate that training with a local cross-entropy (**pred**) or similarity matching (**sim**) loss alone does not match a global loss in terms of test error. However, if both loss functions are combined (**predsim**), the results improve significantly. The performance of the training method varies across architectures and datasets. Overall, we observe no loss in accuracy when comparing to global back-propagation in VGG-like architectures. This conclusion is based on the assumption that the global loss results for VGG architectures are representative. If we compare them with results for residual-free architectures reported in the literature and in open-source implementations, they seem to be equally good or better.

For supervised layer-wise training on CIFAR-10, we improve the state-of-the-art from 7.2% (Belilovsky et al., 2018) to 3.6% test error. Layer-wise training of VGG-like models is competitive with residual architectures trained with global back-propagation on several datasets. For STL-10 with no data augmentation, our result is the best reported.

We use dropout (Srivastava et al., 2014), and batch-normalization (Ioffe & Szegedy, 2015), for regularization. Without these, the results are much worse. We have demonstrated that more advanced regularization methods like cutout, (Devries & Taylor, 2017), can improve the results further.

We found that VGG-like architectures work best with the proposed training method. For residual architectures like ResNet and WideResNet, we got better results when the residual connections were removed. We also tried to replace max-pooling layers with 2-strided convolutional layers, but this did not work equally well.

Avoiding a global loss function has several benefits for practical neural network training. The backward-locking problem is no longer a problem, and weights can be updated during the forward pass. This alleviates the memory requirements since activations do not have to be kept in memory for the backward pass. We trained all layers simultaneously, but using local loss functions can also enable greedy training of hidden layers one-by-one. It also allows for model and data parallelism, where different parts of the model can be trained on different GPU's, with each GPU processing different batches.

5.2. Decoupling Optimization from Generalization

We have compared test errors of various losses on a wide range of datasets. These results do not on their own help us disentangle effects on optimization from effects on generalization. Looking at the training error can shed light on this. In general, full backprop achieved a faster drop in training error, and as low or lower final training error, compared to the local losses in our experiments. This is not too surprising, considering it has access to the true gradient of the global loss at each layer.

On STL-10, the predsim loss achieved the best reported test error without using unlabeled data. This dataset is characterized by relatively large images (96x96), and few training examples (5000). Large models are prone to overfit on this data. Both backprop and each of the local losses were able to reach a training error $< 0.2\%$. At the same time, each of the local losses found solutions with lower test error than that of backprop. This immediately suggests that local learning may provide an inductive bias towards solutions that generalize well.

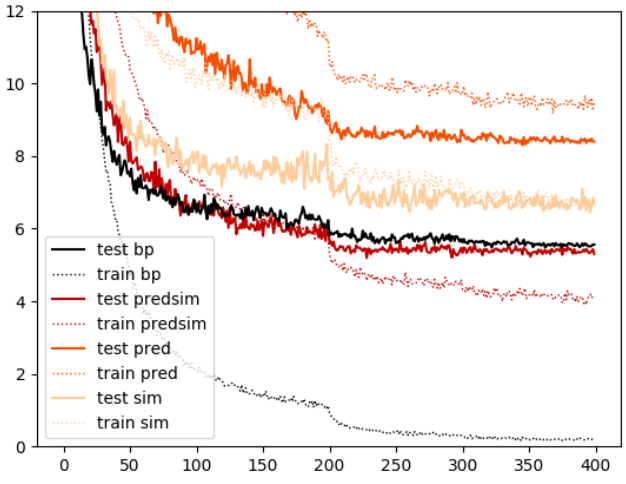

Figure 2. Training and test classification errors on CIFAR10 with full backprop and different local loss functions, with a VGG11B(1x) network.

Observing that the combined local loss generally achieves lower test error than each loss on their own, one could argue that the **sim** loss may be adding a regularizing effect. Looking at the training curves (Figure 2) shows that this is *not* the full story. Here, the **sim** loss achieved a significantly lower training error than **pred**, and their combination in **predsim** achieved lower training error than either method on its own. In every experiment where the local losses could be compared, it held true that the **predsim** loss achieved a lower training error than either local loss on its own. This shows that both losses help *optimization* in the context of local learning in deep networks.

To summarize, our evidence points to full backprop generally achieving a faster drop in training error, and lower final training error, but that local learning in many experiments appeares to add an inductive bias that reduces overfitting. In the context of local learning, the **sim** and **pred** losses both help with optimization in a complementary way.

We have shown that the **predsim** loss can achieve strong test error results on many datasets. If our results come from better generalization, one could suspect local learning to perform worse on large datasets like ImageNet, where models are less prone to overfit. Due to time and compute constraints, we do not have ImageNet results in this paper. However, recent results from (Belilovsky et al., 2018) established that local layer-wise training can scale to ImageNet, by achieving a surprisingly low top-5 single-crop test error of 10.2% on ImageNet with a loss similar to what we call **pred** loss. This leaves open the possibility that adding a **sim** loss could improve this result further.

5.3. Biological Plausibility

We have proposed a combination of local loss functions that offers an alternative to end-to-end training with global back-propagation. Neither of the two losses provide backprop free training. However, the error does not have to be transported back through the whole network, a single step of back-propagation is sufficient.

If we remove the back-propagation requirement and add direct projections from the global target to hidden layers (**predsim-bpf**), the performance deteriorated, but 7.8% error on CIFAR-10 is still the best reported result for backprop free methods. This method is biologically plausible in many ways, but some issues still exist. We use unrealistic weight-sharing in convolutional layers, and we allow the weights to switch sign. We also use batch-normalization, which doesn't have a biologically realistic counterpart. The method is an offline algorithm since we use mini-batch training. In addition we have ignored that real neurons communicate with spikes.

Local loss functions could be a step towards a solution to the credit assignment problem (Bengio et al., 2015). Using local classifiers to generate training signals has been investigated previously. Our contribution is to show that such classifiers can be trained with feedback alignment, and that the target can be replaced with a random projection of the target. We also show increased accuracy when the prediction loss is combined with a similarity matching loss.

With the backprop free **pred-bpf** loss, the weight transport problem, (Grossberg, 1987), is avoided because feedback alignment does not require symmetric weights. An online version of feedback alignment learning can be implemented in a biologically plausible way using multi-compartment neurons (Guerguiev et al., 2017), and in cortical microcircuits for continuous learning without separate forward and backward passes (Sacramento et al., 2018). An online version of the **sim-bpf** loss can be implemented using local Hebbian and anti-Hebbian learning rules, at least for unsupervised learning (Pehlevan & Chklovskii, 2014; Pehlevan et al., 2018; Giovannucci et al., 2018).

Biologically plausible algorithms for error-driven learning have so far focused on how to transport the error back to hidden layers (Lillicrap et al., 2014; Nøkland, 2016), or how to transport a target back to hidden layers (Xie & Seung, 2003; Bengio, 2014; Lee et al., 2015b; Scellier & Bengio, 2017; Whittington & Bogacz, 2017). In the context of experiments performed in this paper, global error transportation is not a requirement for error-driven learning. Neither is it a requirement to propagate the global target backwards through the network. The hidden layers can be trained independently of the layers above, without loss in accuracy.

References

Bartunov, S., Santoro, A., Richards, B. A., Hinton, G. E., and Lillicrap, T. P. Assessing the scalability of biologically-motivated deep learning algorithms and architectures. *CoRR*, abs/1807.04587, 2018. URL http://dblp.uni-trier.de/db/journals/corr/corr1807.html#abs-1807-04587.

Belilovsky, E., Eickenberg, M., and Oyallon, E. Greedy layerwise learning can scale to imagenet. *CoRR*, abs/1812.11446, 2018. URL http://arxiv.org/abs/1812.11446.

Bengio, Y. How auto-encoders could provide credit assignment in deep networks via target propagation. *CoRR*, abs/1407.7906, 2014. URL http://arxiv.org/abs/1407.7906.

Bengio, Y., Lamblin, P., Popovici, P., and Larochelle, H. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems*, volume 19. MIT Press, Cambridge, MA., 2007.

Bengio, Y., Lee, D., Bornschein, J., Mesnard, T., and Lin, Z. Towards biologically plausible deep learning. *CoRR*, abs/1502.04156, 2015.

Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Lamb, A., Yamamoto, K., and Ha, D. Deep learning for classical japanese literature. *CoRR*, abs/1812.01718, 2018. URL http://arxiv.org/abs/1812.01718.

Coates, A. and Ng, A. Y. Selecting receptive fields in deep networks. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 24*, pp. 2528–2536. Curran Associates, Inc., 2011. URL http://papers.nips.cc/paper/4293-selecting-receptive-fields-in-deep-networks.pdf.

Coates, A., Lee, H., and Ng, A. Y. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, pp. 215–223, 2011. URL http://www.jmlr.org/proceedings/papers/v15/coates11a/coates11a.pdf.

Cubuk, E. D., Zoph, B., Mané, D., Vasudevan, V., and Le, Q. V. Autoaugment: Learning augmentation policies from data. *CoRR*, abs/1805.09501, 2018. URL http://arxiv.org/abs/1805.09501.

D. E. Rumelhart, G. E. Hinton, R. J. W. Learning internal representations by error propagation. *Nature*, 323:533–536, 1986.

Devries, T. and Taylor, G. W. Improved regularization of convolutional neural networks with cutout. *CoRR*, abs/1708.04552, 2017. URL http://arxiv.org/abs/1708.04552.

Ding, C., He, X., and Simon, H. D. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proceedings of the 2005 SIAM International Conference on Data Mining*, pp. 606–610. SIAM, 2005.

Dong, L., Gan, Y., Mao, X., Yang, Y., and Shen, C. Learning deep representations using convolutional auto-encoders with symmetric skip connections. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2018, Calgary, AB, Canada, April 15-20, 2018*, pp. 3006–3010, 2018. doi: 10.1109/ICASSP.2018.8462085. URL https://doi.org/10.1109/ICASSP.2018.8462085.

Erhan, D., Bengio, Y., Courville, A. C., Manzagol, P., Vincent, P., and Bengio, S. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11:625–660, 2010. URL https://dl.acm.org/citation.cfm?id=1756025.

Fisher, R. A. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(7):179–188, 1936.

Giovannucci, A., Minden, V., Pehlevan, C., and Chklovskii, D. B. Efficient principal subspace projection of streaming data through fast similarity matching. *CoRR*, abs/1808.02083, 2018. URL http://arxiv.org/abs/1808.02083.

Goldberger, J., Hinton, G. E., Roweis, S. T., and Ruslan R, S. Neighbourhood components analysis. In Saul, L. K., Weiss, Y., and Bottou, L. (eds.), *Advances in Neural Information Processing Systems 17*, pp. 513–520. MIT Press, 2005. URL http://papers.nips.cc/paper/2566-neighbourhood-components-analysis.pdf.

Gomez, A. N., Ren, M., Urtasun, R., and Grosse, R. B. The reversible residual network: Backpropagation without storing activations. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 2211–2221, 2017.

Grossberg, S. Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science*, 11(1):23–63, 1987. doi: 10.1111/j.1551-6708.1987.tb00862.x. URL https://doi.org/10.1111/j.1551-6708.1987.tb00862.x.

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. URL https://doi.org/10.7554/eLife.22901.

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, pp. 630–645, 2016. doi: 10.1007/978-3-319-46493-0_38. URL https://doi.org/10.1007/978-3-319-46493-0_38.

Hinton, G. E., Osindero, S., and Teh, Y. W. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006. doi: 10.1162/neco.2006.18.7.1527. URL http://dx.doi.org/10.1162/neco.2006.18.7.1527.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015*, volume 37 of *JMLR Proceedings*, pp. 448–456. JMLR.org, 2015. URL http://jmlr.org/proceedings/papers/v37/ioffe15.html.

Jaderberg, M., Czarnecki, W. M., Osindero, S., Vinyals, O., Graves, A., Silver, D., and Kavukcuoglu, K. Decoupled neural interfaces using synthetic gradients. In Precup, D. and Teh, Y. W. (eds.), *ICML*, volume 70 of *JMLR Workshop and Conference Proceedings*, pp. 1627–1635. JMLR.org, 2017. URL http://dblp.uni-trier.de/db/conf/icml/icml2017.html#JaderbergCOVGSK17.

Kiani, R., Esteky, H., Mirpour, K., and Tanaka, K. Object category structure in response patterns of neuronal population in monkey inferior temporal cortex. *J. Neurophysiol*, pp. 4296–4309, 2007.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980.

Kriegeskorte, N., Mur, M., and Bandettini, P. Representational similarity analysis - connecting the branches of systems neuroscience. *Frontiers in systems neuroscience*, 2, 2008a. ISSN 1662-5137. doi: 10.3389/neuro.06.004.2008. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2605405.

Kriegeskorte, N., Mur, M., Ruff, D. A., Kiani, R., Bodurka, J., Esteky, H., Tanaka, K., and Bandettini, P. A. Matching categorical object representations in inferior temporal cortex of man and monkey. *Neuron*, 60(6):1126 – 1141, 2008b. ISSN 0896-6273. doi: https://doi.org/10.1016/j.neuron.2008.10.043. URL http://www.sciencedirect.com/science/article/pii/S0896627308009434.

Krizhevsky, A. Learning multiple layers of features from tiny images. *Technical report, University of Toronto*, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Kuang, D., Ding, C., and Park, H. Symmetric nonnegative matrix factorization for graph clustering. In *Proceedings of the 2012 SIAM international conference on data mining*, pp. 106–117. SIAM, 2012.

Lee, C.-Y., Xie, S., Gallagher, P. W., Zhang, Z., and Tu, Z. Deeply-supervised nets. In Lebanon, G. and Vishwanathan, S. V. N. (eds.), *AISTATS*, volume 38 of *JMLR Proceedings*. JMLR.org, 2015a. URL http://dblp.uni-trier.de/db/conf/aistats/aistats2015.html#LeeXGZT15.

Lee, D., Zhang, S., Fischer, A., and Bengio, Y. Difference target propagation. In *ECML/PKDD (1)*, Machine Learning and Knowledge Discovery in Databases, pp. 498–515. Springer International Publishing, 2015b.

Lee, D. D. and Seung, H. S. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788, 1999.

Lillicrap, T. P., Cownden, D., Tweed, D. B., and Akerman, C. J. Random feedback weights support learning in deep neural networks. *CoRR*, abs/1411.0247, 2014. URL http://arxiv.org/abs/1411.0247.

Maas, A. L., Hannun, A. Y., and Ng, A. Y. Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

Malach, E. and Shalev-Shwartz, S. A provably correct algorithm for deep learning that actually works. *CoRR*, abs/1803.09522, 2018. URL http://arxiv.org/abs/1803.09522.

Marquez, E. S., Hare, J. S., and Niranjan, M. Deep cascade learning. *IEEE Trans. Neural Netw. Learning Syst.*, 29(11):5475–5485, 2018. doi: 10.1109/TNNLS.2018.2805098. URL https://doi.org/10.1109/TNNLS.2018.2805098.

Moskovitz, T. H., Litwin-Kumar, A., and Abbott, L. Feedback alignment in deep convolutional networks. *CoRR*, 12 2018.

Mostafa, H., Ramesh, V., and Cauwenberghs, G. Deep supervised learning using local errors. *CoRR*, abs/1711.06756, 2017. URL http://dblp.uni-trier.de/db/journals/corr/corr1711.html#abs-1711-06756.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. 2011.

Ng, A. Y., Jordan, M. I., and Weiss, Y. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pp. 849–856, 2002.

Nøkland, A. Direct feedback alignment provides learning in deep neural networks. In Lee, D. D., Sugiyama, M., von Luxburg, U., Guyon, I., and Garnett, R. (eds.), *NIPS*, pp. 1037–1045, 2016. URL http://dblp.uni-trier.de/db/conf/nips/nips2016.html#Nokland16.

Paine, T. L., Khorrami, P., Han, W., and Huang, T. S. An analysis of unsupervised pre-training in light of recent advances. *CoRR*, abs/1412.6597, 2014. URL http://arxiv.org/abs/1412.6597.

Pehlevan, C. and Chklovskii, D. B. A hebbian/anti-hebbian network derived from online non-negative matrix factorization can cluster and discover sparse features. In *Signals, Systems and Computers, 2014 48th Asilomar Conference on*, pp. 769–775. IEEE, 2014.

Pehlevan, C., Sengupta, A. M., and Chklovskii, D. B. Why do similarity matching objectives lead to hebbian/anti-hebbian networks? *Neural Computation*, 30(1), 2018. URL http://dblp.uni-trier.de/db/journals/neco/neco30.html#PehlevanSC18.

Raina, R., Battle, A., Lee, H., Packer, B., and Ng, A. Y. Self-taught learning: Transfer learning from unlabeled data. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pp. 759–766, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-793-3. doi: 10.1145/1273496.1273592. URL http://doi.acm.org/10.1145/1273496.1273592.

Rasmus, A., Berglund, M., Honkala, M., Valpola, H., and Raiko, T. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 3546–3554, 2015. URL http://papers.nips.cc/paper/5947-semi-supervised-learning-with-ladder-networks.

Sabour, S., Frosst, N., and Hinton, G. E. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 3859–3869, 2017. URL http://papers.nips.cc/paper/6975-dynamic-routing-between-capsules.

Sacramento, J., Costa, R. P., Bengio, Y., and Senn, W. Dendritic cortical microcircuits approximate the backpropagation algorithm. *CoRR*, abs/1810.11393, 2018. URL http://dblp.uni-trier.de/db/journals/corr/corr1810.html#abs-1810-11393.

Salakhutdinov, R. and Hinton, G. E. Deep boltzmann machines. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, AISTATS 2009*, volume 5 of *JMLR Proceedings*, pp. 448–455. JMLR.org, 2009. URL http://www.jmlr.org/proceedings/papers/v5/salakhutdinov09a.html.

Scellier, B. and Bengio, Y. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Front. Comput. Neurosci.*, 2017, 2017. doi: 10.3389/fncom.2017.00024. URL https://doi.org/10.3389/fncom.2017.00024.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL http://arxiv.org/abs/1409.1556.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014. URL http://dl.acm.org/citation.cfm?id=2670313.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *CVPR*, pp. 1–9. IEEE Computer Society, 2015. ISBN 978-1-4673-6964-0. URL http://dblp.uni-trier.de/db/conf/cvpr/cvpr2015.html#SzegedyLJSRAEVR15.

Verma, V., Lamb, A., Beckham, C., Najafi, A., Courville, A., Mitliagkis, I., and Bengio, Y. Manifold mixup: Encouraging meaningful on-manifold interpolation as a regularizer. *arXiv preprint arXiv:1806.05236v3*, 2018.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P. Extracting and composing robust features with denoising autoencoders. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, pp. 1096–1103, 2008. doi: 10.1145/1390156.1390294. URL http://doi.acm.org/10.1145/1390156.1390294.

Wan, L., Zeiler, M. D., Zhang, S., LeCun, Y., and Fergus, R. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Proceedings*, pp. 1058–1066. JMLR.org, 2013. URL http://jmlr.org/proceedings/papers/v28/wan13.html.

Wang, L., Lee, C., Tu, Z., and Lazebnik, S. Training deeper convolutional networks with deep supervision. *CoRR*, abs/1505.02496, 2015. URL http://arxiv.org/abs/1505.02496.

Weston, J., Ratle, F., Mobahi, H., and Collobert, R. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade - Second Edition*, pp. 639–655. 2012. doi: 10.1007/978-3-642-35289-8_34. URL https://doi.org/10.1007/978-3-642-35289-8_34.

Whittington, J. C. R. 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. doi: 10.1162/NECO_a_00949. URL https://doi.org/10.1162/NECO_a_00949.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017. URL https://github.com/zalandoresearch/fashion-mnist.

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

Yamada, Y., Iwamura, M., and Kise, K. Shakedrop regularization. *CoRR*, abs/1802.02375, 2018. URL http://arxiv.org/abs/1802.02375.

Zagoruyko, S. and Komodakis, N. Wide residual networks. In *Proceedings of the British Machine Vision Conference 2016, BMVC*, 2016. URL http://www.bmva.org/bmvc/2016/papers/paper087/index.html.

Zhang, Y., Lee, K., and Lee, H. Augmenting supervised neural networks with unsupervised objectives for large-scale image classification. In Balcan, M.-F. and Weinberger, K. Q. (eds.), *ICML*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 612–621. JMLR.org, 2016. URL http://dblp.uni-trier.de/db/conf/icml/icml2016.html#ZhangLL16.

Zhong, Z., Zheng, L., Kang, G., Li, S., and Yang, Y. Random erasing data augmentation. *CoRR*, abs/1708.04896, 2017. URL http://arxiv.org/abs/1708.04896.