REDUCING OVERFITTING IN DEEP NETWORKS BY DECORRELATING REPRESENTATIONS

Michael Cogswell
Virginia Tech
Blacksburg, VA
cogswell@vt.edu

Faruk Ahmed
Université de Montréal
Montréal, Quebec, Canada
faruk.ahmed@umontreal.ca

Ross Girshick
Facebook AI Research (FAIR)
Seattle, WA
rbg@fb.com

Larry Zitnick
Microsoft Research
Seattle, WA
larryz@microsoft.com

Dhruv Batra
Virginia Tech
Blacksburg, VA
dbatra@vt.edu

ABSTRACT

One major challenge in training Deep Neural Networks is preventing overfitting. Many techniques such as data augmentation and novel regularizers such as Dropout have been proposed to prevent overfitting without requiring a massive amount of training data. In this work, we propose a new regularizer called DeCov which leads to significantly reduced overfitting (as indicated by the difference between train and val performance), and better generalization. Our regularizer encourages diverse or non-redundant representations in Deep Neural Networks by minimizing the cross-covariance of hidden activations. This simple intuition has been explored in a number of past works but surprisingly has never been applied as a regularizer in supervised learning. Experiments across a range of datasets and network architectures show that this loss always reduces overfitting while almost always maintaining or increasing generalization performance and often improving performance over Dropout.

1 INTRODUCTION

Deep Neural Networks (DNNs) have recently achieved remarkable success on a wide range of tasks – *e.g.*, image classification on ImageNet (Krizhevsky et al., 2012), scene recognition on MIT Places (Zhou et al., 2014), image captioning with MS COCO (Lin et al., 2014b; Vinyals et al., 2015; Chen & Zitnick, 2015), and visual question answering (Antol et al., 2015). One significant reason for improvement of these methods over their predecessors has to do with scale. Faster computers coupled with optimization improvements such Batch Normalization, Adaptive SGD, and ReLus let us quickly train wider and deep networks. Access to large annotated datasets and regularizers such as Dropout has provided significant reduction in the amount of overfitting in these large networks, thus enabling the performance we see today.

In this paper, we focus on the problem of overfitting, which is observed when a high capacity model (such as a DNN) performs very well on training data but poorly on held out data. Even when trained on large annotated datasets (such as ImageNet (Deng et al., 2009) or Places (Zhou et al., 2014), containing millions of labelled images), deep networks are susceptible to overfitting. This problem is further exacerbated when moving to new domains and tasks – since DNNs tend not to generalize with a few examples, each new task tends to require curating and annotating a new large dataset. While there has been some success with transfer learning (Girshick et al., 2014; Donahue et al., 2014; Yosinski et al., 2014), networks still overfit.

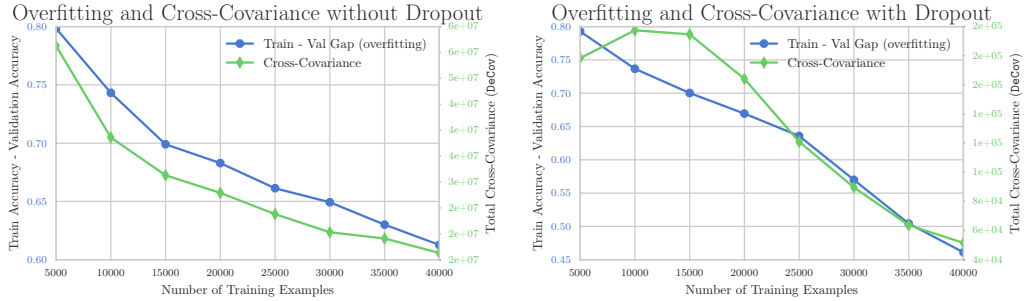
A promising alternative to creating even larger datasets is to apply different forms of regularization to the network while training to avoid overfitting. These methods include regularizing the norm of the weights (Tikhonov, 1943), Lasso (Tibshirani, 1996), Dropout (Srivastava et al., 2014), DropConnect (Wan et al., 2013), Maxout (Goodfellow et al., 2013), etc.

One particular regularizer of interest to DNNs is Dropout (Srivastava et al., 2014), which attempts to prevent co-adaptation of neuron activations. Co-adaptation occurs when two or more hidden units rely on one another to perform some function which helps fit training data, thus becoming highly

correlated. Co-adaptation is reduced by Dropout using an approximate model averaging technique that sets a randomly selected set of activations to zero at training time. Srivastava et al. (2014) show that this has a regularizing effect, leading to increased generalization and sparser, less correlated features. Notice that this is without *explicitly* encouraging decorrelation in hidden activations.

To further investigate the relationship between hidden activation correlations and overfitting, we show in Fig. 1 two quantities from a CNN trained for image classification on CIFAR100 (Krizhevsky & Hinton, 2009) – (1) the amount of overfitting in the model (as measured by the gap between train and val accuracy), and (2) the amount of correlation in hidden activations (as measured by the Frobenius norm of the sample cross-covariance matrix computed from vectors of hidden activations; details in Section 2). Both these quantities of interest are reported as a function of amount of training data (x-axis) and with/without Dropout (left/right subplot). As expected, both increased training data and Dropout have a regularizing effect and lead to reduced overfitting.

The figure also shows an interesting novel trend – as the amount of overfitting reduces, so does the degree of correlation in hidden activations. In essence, overfitting and co-adaptation seem to be correlated. The open question of course is – is the relationship causal?



(a) Without Dropout (b) With Dropout

Figure 1: Two principal ways to prevent overfitting in deep models are to train with more data (x axis) and to train with Dropout (right plot). As expected, both of these decrease validation error (left axis), but they also happen to decrease hidden activation cross-covariance (right axis). We investigate whether explicitly minimizing cross-covariance can lead to reduced overfitting.

This leads to the principal questions of this paper – Is it possible to bias networks towards decorrelated representations by directly reducing correlation between hidden units? And do such decorrelated representations generalize better?

Overview and Contributions. The goal of this paper is to learn DNNs with decorrelated activations and study the effect of this decorrelation on their generalization performance. Towards this end, we propose a fairly natural loss called `DeCov`, which explicitly encourages decorrelation between the activations in a deep neural network. This loss requires no additional supervision, so it can be added to any existing network.

In addition to the link discussed above, our motivation also comes from the classical literature on bagging and ensemble averaging (Hansen & Salamon, 1990; Perrone & Cooper, 1993; Breiman, 1996), which suggests that decorrelated ensembles perform better than correlated ones.

Our experiments encompass a range of datasets (MNIST (LeCun et al., 1995), CIFAR10/100 (Krizhevsky & Hinton, 2009), ImageNet (Deng et al., 2009)), and different kinds of network architectures (Caffe implementations of LeNet (LeCun et al., 1995), AlexNet (Krizhevsky et al., 2012), and Network in Network (Lin et al., 2014a)). All cases suggest that `DeCov` acts as a novel and useful regularizer.

2 Approach: DeCov Loss

To express our notion of redundant or co-adapated features, we impose a loss on the activations of a chosen hidden layer. In a manner similar to Dropout, our proposed Decov loss may be applied to a single layer or multiple layers in a network. For simplicity, let us focus on a single layer. Let $\mathbf{h}^n \in \mathbb{R}^d$ denote the activations at the chosen hidden layer, where $n \in \{1, \ldots, N\}$ indexes one example from a batch of size N. The covariances between all pairs of activations i and j form a matrix C:

$$C_{i,j} = \frac{1}{N} \sum_n (h_i^n - \mu_i)(h_j^n - \mu_j) \tag{1}$$

where $\mu_i = \frac{1}{N} \sum_n h_i^n$ is the sample mean of activation i over the batch.

We want to minimize covariance between different features, which corresponds to penalizing the norm of C. However, the diagonal of C contains the variance of each hidden activation and we have no reason to require the dynamic range of activations to be small, so we subtract this term from the matrix norm to get our final DeCov loss

$$\mathcal{L}_{\texttt{DeCov}} = \frac{1}{2}\left(\|C\|_F^2 - \|diag(C)\|_2^2\right) \tag{2}$$

where $\|\cdot\|_F$ is the frobenius norm, and the $diag(\cdot)$ operator extracts the main diagonal of a matrix into a vector. In our experiments, subtracting the diagonal made little difference for small networks, but led to increased stability for larger networks.

Perhaps the best quality of this loss is that it requires no supervision, so it can be added to any set of activations. In a manner similar to Dropout, our experiments typically apply Decov loss to fully connected layers towards the deep end of a network (*e.g.*, fc6 and fc7 for AlexNet). However, note that Decov affects *all parameters* up to the layer where it is applied (and not just the parameters in the specific layer).

At first glance, one seeming peculiarity about this loss is that its global minimum can be found by setting all weights for $\mathbf{h}$ to 0. This is similar to an L_2 regularizer in that both encourage weights to tend toward 0, but one important difference between these two regularizers is that $\mathcal{L}_{\texttt{DeCov}}$ depends on input data and is not a function purely of a weight vector like one might find in a classical regularizer such as L_2 or L_1.

To understand this further, consider the gradient of the loss with respect to a particular activation a for a particular example m

$$\frac{\partial \mathcal{L}_{\texttt{DeCov}}}{\partial h_a^m} = \frac{1}{N}\sum_{j\neq a}\left[\frac{1}{N}\sum_n (h_a^n - \mu_a)(h_j^n - \mu_j)\right](h_j^m - \mu_j). \tag{3}$$

Let us denote the rightmost term in this expression by $I(j,m) = (h_j^m - \mu_j)$.

This term is large (in absolute value) when feature j is discriminative for example m w.r.t. the mean of the batch. If j were not discriminative for m then h_j^m would be close to μ_j. Hence, we can consider I as an "importance" term, corresponding to a notion of how significant feature j is for example m.

Also notice that the term on the left in the gradient expression is simply the covariance between feature a and feature j. Thus, the gradient can be re-written as

$$\frac{\partial \mathcal{L}_{\texttt{DeCov}}}{\partial h_a^m} = \frac{1}{N}\sum_{j\neq a} C_{a,j}\cdot I(j,m). \tag{4}$$

Interpretation. Intuitively, the covariance term can be thought of as measuring (linear) redundancy: features a and j are redundant if they vary together. Thus, the DeCov loss tries to prevent features from being redundant, but redundancy is weighted by importance (I). Specifically, a feature j contributes towards a large gradient of feature a on example m if j is important for m *and* correlated with a. This means important features correlated with a (*e.g.*, j) contribute to a large gradient of a, suppressing the activation h_a^m. A feature which fires only in specialized situations (*e.g.*, a cat's ear) will likely be nearly identical or noisy for most other examples (*e.g.*, non-cats) and will not contribute towards gradients of other specialized features.

3 Related Work

Redundancy Based Representations. The idea of using low redundancy to learn representations has been around for decades. In an early attempt to model human perception, Barlow (1961) lists 3 possible learning principles, the 3rd being the notion that representations should not be redundant.

Later work continued to investigate this intuition in the context of unsupervised feature learning. Three objectives emerged, each of which formalize the notion differently. (1) An information theoretic view is expressed by Linsker (1988). The main idea is to maximize information gained by predicting the next representation/layer between input and output. (2) The closest objective to ours is cross-correlation (not cross-covariance), which appears in (Bengio & Bergstra, 2009) and complements a temporal coherence objective. It also appears in (Pearlmutter & Hinton, 1986) where it complements an objective which encourages units to capture higher order input statistics. (3) Finally, redundancy minimization is realized through predictability minimization in (Schmidhuber, 1992) for the purpose of learning factorial codes (representations whose units are independent). This objective says that one unit should not be predictable given *all* of the others in its layer as input.

All of these works focus on unsupervised feature learning and do not experiment with supervised models. Furthermore, these early pioneering works were limited by data and evaluated small networks without many of the modern design choices and features (e.g. ReLus, Dropout, SGD instead of Hebb's update rule, batch-normalization, *etc*.). We propose redundancy minimization for a new purpose (regularization), evaluate it using modern techniques such as end-to-end learning using SGD with respect to a supervised objective, and do this in the context of harder challenges presented by modern datasets. To the best of our knowledge, such a setting has not been considered before.

Correlation/Covariance Losses in Other Settings. Other works have used similar penalties, but in different settings and to different effects. Deep Canonical Correlation Analysis (Deep CCA) (Andrew et al., 2013) and Correlational Neural Networks (CorrNets) (Chandar et al., 2015) apply a similar loss which *maximizes* correlation, unlike our *minimization* of cross-covariance. Both methods are used to learn better features in the presence of multiple views or modalities. They embed inputs to a common space and maximize correlation between aligned pairs.

Another idea similar to ours is that of Cheung et al. (2014), which aims to discover and disentangle hidden factors. The goal is to separate supervised factors of variation (*e.g*., class of MNIST digits) from unsupervised factors of variation (*e.g*., handwriting style). In order to achieve this goal, they impose a covariance (not correlation) loss between (1) the softmax outputs of a neural network trained to recognize digits and (2) a hidden representation which is used in conjunction with (1) to reconstruct the input (via an auto-encoder).

These two works suggest that correlation losses significantly impact learned representations in the context of modern networks. One key difference between these two approaches and ours is that while their formulations decorrelate (Cheung et al., 2014) and disregard (Andrew et al., 2013; Chandar et al., 2015) parts of *different* representations, our approach tries to decorrelate parts of the *same* representation. Moreover, the ultimate goals are different. Unlike these approaches, our goal is simply to improve supervised classification performance by reducing overfitting, and not to reconstruct the original data.

Dropout and Batch Normalization. Two recent approaches to regularization in deep neural networks are Dropout (Srivastava et al., 2014) and to some extent Batch Normalization (Ioffe & Szegedy, 2015). Dropout aligns with our intuition and goals more closely as it aims to improve classification performance by reducing co-adaptation of activations. On the other hand, Batch Normalization focuses on faster optimization by reducing *internal co-variate shift*, which is the constant variation of a layer's input as it learns. Some Batch Normalization results indicate it could act as a regularizer, but this has not been exhaustively verified yet. Our approach is similar to Batch Normalization due to its use of mini-batch statistics.

4 Experiments

We begin with a synthetic dual "modality" experiment, which serves as a testbed for measuring improvement due to decorrelation. Next, we use an autoencoder (as in Srivastava et al. (2014)) to contrast `DeCov` and Dropout. Finally, we use a variety of experiments to report Image Classification performance on CIFAR10/100 and ImageNet, noticing significant improvement in *all cases*. Note that we set the Dropout rate to 0.5 as suggested by Srivastava et al. (2014).

4.1 Dual modality experiments with MNIST: Predicting Side-by-Side Digits

We propose a synthetic dual "modality" task on MNIST – simultaneously predict the class labels for two digits placed adjacent in an image. We created a dataset where each example consists of two MNIST digit images horizontally concatenated and separated by 16 black pixels (to prevent interference between feature maps in the first layers). Fig. 2 shows a few examples.

The important detail of this experiment is the particular bias we inject into the distribution of left and right digits. Let

$$P(l) = 0.1 \;\text{ and }\; P(r|l) = \begin{cases} 0 & \text{if } l \in \{0,\ldots,4\} \text{ and } r \in \{0,\ldots,4\} \\ 0.2 & \text{if } l \in \{0,\ldots,4\} \text{ and } r \in \{5,\ldots,9\} \\ 0.1 & \text{if } l \in \{5,\ldots,9\} \end{cases}. \tag{5}$$

To generate one example we first sample the left digit using $P(l)$ then the right using $P(r|l)$. As shown in Appendix A, we can compute the conditional entropies of one digit given the other to get $H(l|r) = 2.0868$ and $H(r|l) = 1.9360$. Since $H(l|r) > H(r|l)$, the left digit is more informative of the right than the right is of the left. There is no cross-digit signal at test time, so features for

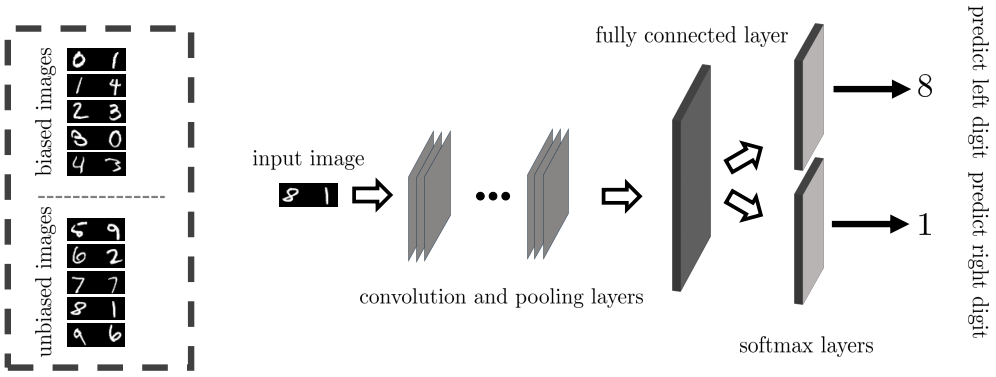


Figure 2: We consider the task of simultaneously predicting two MNIST digits placed side by side. By biasing right digits more than left digits at train time, we create a controlled scenario with the type of problem we expect DeCov to solve.

试验设置的目的.

the right and left digits should be completely decorrelated to generalize, but learned features will have some correlation between left and right. Intuitively, DeCov should help generalization in this scenario. Our experiments support this.

We use Caffe's (Jia, 2013) reference version of LeNet (LeCun et al., 1995). It has two convolution layers, each followed by pooling, then a fully connected layer with 500 hidden units which are shared between the two softmax layers. We apply DeCov and/or Dropout to the 500 hidden units of the fully connected layer.

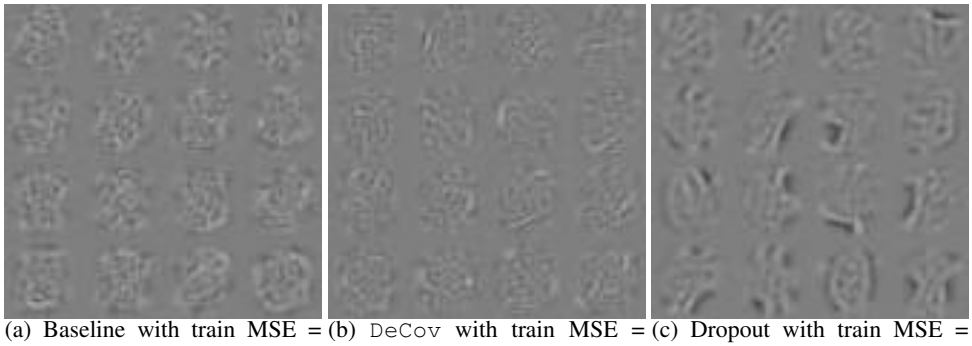
		Left Digit			Right Digit		
DeCov	Dropout	train	test	train - test	train	test	train - test
no	no	99.98 ± 0.01	97.94 ± 0.18	2.05 ± 0.19	100.00 ± 0.00	96.75 ± 0.24	3.25 ± 0.24
no	yes	99.99 ± 0.00	98.45 ± 0.04	1.54 ± 0.04	99.99 ± 0.00	97.39 ± 0.20	2.61 ± 0.20
yes	yes	99.97 ± 0.01	98.59 ± 0.12	1.38 ± 0.12	99.99 ± 0.00	97.81 ± 0.07	2.18 ± 0.06
yes	no	99.99 ± 0.00	$\mathbf{98.74 \pm 0.03}$	$\mathbf{1.25 \pm 0.04}$	99.99 ± 0.00	$\mathbf{97.99 \pm 0.12}$	$\mathbf{2.00 \pm 0.12}$
weight decay		99.97	97.86	2.11	99.97	96.21	3.76

Table 1: **MNIST side by side results.** As expected, biasing right digits at train time so that they are weakly informed by left digits leads to lower performance on an unbiased test set. More importantly, DeCov provides greater improvements over the baselines on the right, confirming that it leads to better features when decorrelation is extremely likely to improve performance.

Results. Table 1 reports the accuracy of left and right digit classifiers. Our injected dataset bias can be clearly seen in the lower test accuracy and higher train-test gap of the right classifier, indicating that all of our networks incorporate the train time bias into their predictions. We report mean accuracies across 4 trials, along with the standard deviation. We also compare the effect of Dropout.

The main result is that the gaps between the performance of DeCov and the baselines are larger for the biased right digit (*e.g.*, right digit test accuracy shows a $\sim$0.6% improvement when switching from Dropout-alone to DeCov-alone while the improvement for left digits is just $\sim$0.3%). This suggests that the baselines pick up on the false bias and that DeCov does the best job of correcting for it. DeCov also improves generalization for both classifiers since test accuracy is higher in the bottom two rows and the train - test gap is lower in those rows. Combining Dropout with our DeCov loss hurts slightly, but we note that the error bars overlap in some cases, so this is not a statistically significant difference.

One skeptical hypothesis is that the DeCov loss is simply enforcing something akin to an L2 penalty on the weights. The experiments with DeCov and Dropout already use an L2 penalty, so this is unlikely, but a grid search over weights on this term shows it makes little difference. The best accuracies are reported in the last row of Table 1.



(a) Baseline with train MSE = 1.47 and test MSE = 1.47 (b) DeCov with train MSE = 0.98 and test MSE = .98 (c) Dropout with train MSE = 3.08 and test MSE = 3.03

Figure 3: Weights learned by the first layer of a 2 layer autoencoder are reshaped into images and visualized for a model with no DeCov or Dropout (Fig. 3a), a model with DeCov (Fig. 3b), and a model with Dropout (Fig. 3c).

4.2 MNIST AUTOENCODER

To offer a more qualitative point of comparison, we visualized learned features using the 2 layer autoencoder experiment from (Srivastava et al., 2014) (section 7). In this experiment an autoencoder is trained on raw pixels of single MNIST digits using an encoder with 1 layer of 256 ReLU units and a decoder (untied weights) that produces 784 (28×28) ReLU outputs. Fig. 3 shows the weights learned by the autoencoder (reshaped to align with the input image) and mean-square reconstruction errors.

Weight initialization turned out to be an important factor for the visualizations. Initializing all weights by sampling from $U[-\sqrt{\frac{3}{n}}, \sqrt{\frac{3}{n}}]$ (based on Glorot & Bengio (2010); as implemented in Caffe) led to visualizations as seen in Srivastava et al. (2014) (the baseline looks like noise), but sampling weights from a Gaussian with mean 0 and standard deviation 0.001 led to baseline visualizations with faint digit outlines. The latter initialization was used in Fig. 3.

One take-away is that MSE is significantly lower for DeCov than others. However, the key take-away is the qualitative difference between representations learned with Dropout and those learned with DeCov. Recall from Section 1 that Dropout reduces cross-covariance while DeCov explicitly minimizes it. Despite this intuitive similarity, the two lead to different learned representations.

4.3 IMAGE CLASSIFICATION

4.3.1 CIFAR10

CIFAR10 contains 60,000 32x32 images sorted into 10 distinct categories (Krizhevsky & Hinton, 2009). We training on the 50,000 given training examples and testing on the 10,000 specified test samples. Hyper-parameters (loss weights for DeCov and weight decay) are chosen by grid search on the standard train/val split.

We use Caffe's quick CIFAR10 architecture, which has 3 convolutional layers followed by a fully connected layer with 64 hidden units and a softmax layer. The hidden fully connected layer is not followed by a non-linearity. The DeCov loss is added only to the 64 hidden units in the hidden fully connected layer. All reported results are average performance over 4 trials with the standard deviation indicated alongside.

DeCov	Dropout	train	test	train - test
no	no	100.0 ± 0.00	75.24 ± 0.27	24.77 ± 0.27
no	yes	99.10 ± 0.17	77.45 ± 0.21	21.65 ± 0.22
yes	yes	87.78 ± 0.08	$\mathbf{79.75 \pm 0.17}$	$\mathbf{8.04 \pm 0.16}$
yes	no	88.78 ± 0.23	79.72 ± 0.14	9.06 ± 0.22
weight decay		100.0	75.29	24.71

Table 2: CIFAR10 Classification. We can see that DeCov with Dropout leads to the highest test performance and the lowest train-test gap.

Results. In Table 2, we again observe significant improvements when using the DeCov loss – there is a ∼4.5% improvement in test accuracy (over no regularization). Moreover, the DeCov loss reduces the gap between train and val accuracies by ∼15% (without Dropout) and ∼16% (with Dropout)!

Comparing the four combinations, we see that using DeCov alone provides a larger improvement than using Dropout. Using both DeCov and Dropout further improves the generalization (as measured by the gap in train and test accuracies), but the improvement in absolute test performance does not seem statistically significant.

We again test if L2 weight decay can provide similar improvements and find once again that the best setting gives little improvement over the baseline.

One promise of regularization is the ability to train larger networks, so we increase the size of our CIFAR10 network. We add another fully connected layer to the network used in the previous experiment, double the number of filters in each convolutional layer, and double the number of units in the fully connected layers. This larger network performs better than the smaller version – all accuracies are higher than corresponding entries in Table 2. However, there are the stronger indications of overfitting in this network – specifically, the train accuracies are much higher than test accuracies (when compared to the previous network). Table 3 shows the results. We observe similar trends as the previous experiment – there are significant gains from using DeCov alone compared to Dropout alone, and there is a further slight improvement in combining both. Using Dropout alone gives a ∼1.5% boost in test accuracy, while using DeCov alone provides a ∼4% increase in test accuracy. Using both yields roughly the same test performance, but the trainval and test gap is further reduced.

DeCov	Dropout	(train+val)	test	(train+val) - test
no	no	100.00	77.38	22.62
no	yes	100.00	79.93	20.07
yes	yes	96.76	**81.68**	**15.08**
yes	no	98.15	81.63	16.52

Table 3: CIFAR10 Classification with a bigger version of the base network

4.3.2 CIFAR100

To scale up our experiments, we move to CIFAR100 (Krizhevsky & Hinton, 2009). We use the same architecture as the base architecture for CIFAR10 and hold out the last 10,000 of the 50,000 train examples for validation. Table 4 shows that Dropout alone highest higher test performance than DeCov alone, but DeCov leads to a smaller train-test gap. Using both regularizers not only achieves the highest test accuracy, but also the smallest train-test gap (∼34% smaller than using neither regularizer). This suggests that the two regularizers may have complementary effects.

DeCov	Dropout	train	test	train - test
no	no	99.77	38.52	61.25
no	yes	87.35	43.55	43.80
yes	yes	72.53	**45.10**	**27.43**
yes	no	77.92	40.34	37.58

Table 4: CIFAR100 Classification Accuracies

One more problem comes with the question of how to weight the DeCov loss. All of our experiments use grid search to pick this hyper-parameter. The optimal weight varies across datasets, but we have found consistency across variations in architecture. We varied both the DeCov weight and the number of hidden units in the fully connected layer to which DeCov is applied, training a new network for each setting. The best DeCov weight (0.1) is consistent for a range of hidden activation sizes in this dataset, though it is different in other experiments.

4.3.3 IMAGENET

Now we explore results for networks trained for ImageNet classification, starting by applying DeCov to fc6 and fc7 in AlexNet (Krizhevsky et al., 2012). The last 50,000 of the ILSVRC 2012 train images are held out for validation. Our implementation comes from Caffe. In particular, it

uses a fixed schedule that multiplies the learning rate by 1/10 every 100,000 iterations (see jumps in Fig. 4). We do not use early stopping and do not perform color augmentation.

In Fig. 4 we notice that when neither of the two regularizers – Dropout or DeCov– are applied (blue line), the network overfits (it even gets 100% train accuracy), and the DeCov loss (hidden activation redundancy) is higher than with any other combination of the regularizers. Applying either of the regularizers also causes a synchronous drop in both losses. Explicitly minimizing the DeCov loss naturally leads to much lower DeCov losses, and we notice that this coincides with significantly reduced overfitting. Interestingly, Dropout results in relatively lower DeCov loss too, even when DeCov is not optimized for. This is further indication of the link between redundant activations and overfitting.

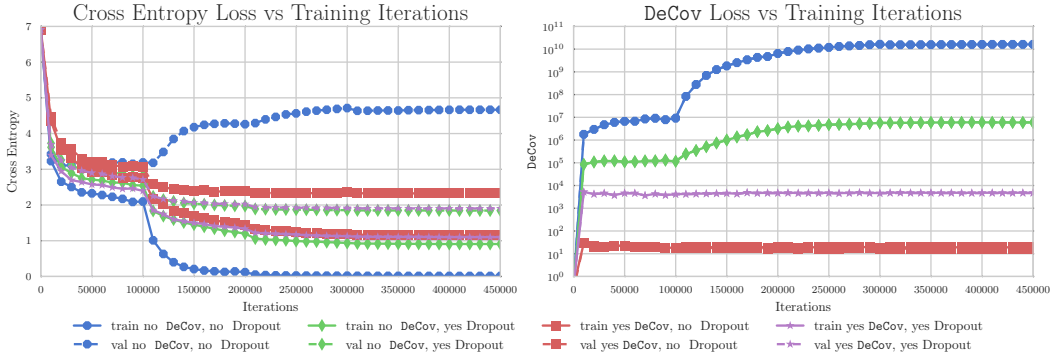


Figure 4: Cross Entropy and DeCov losses over the course of training AlexNet with 256x256 images. Note that the DeCov val curves are hidden by the train curves. Interestingly, DeCov is reduced even by Dropout, though not nearly as much as when it is explicitly minimized.

Fig. 5 shows accuracies across different image resolutions we used to train AlexNet. AlexNet is typically trained with 256x256 images, but training with smaller images is faster [1] *and* reduces the number of parameters in the network. Smaller images (we use 128x128, 160x160, 192x192, and 224x224) lead to smaller feature maps output by pool5, so the dense connection between pool5 and fc6 has fewer parameters, the model has less capacity, and it's less likely to overfit. For example, images scaled to 256x256 (taking 227x227 crops [2]) lead to a weight matrix with 38 million parameters while 128x128 images (with 99x99 crops) result in a 4 million parameter matrix. Generally, accuracies (left plots) and the train-val gap (right plots) have a slight positve slope, confirming that performance and overfitting increase with resolution and model capacity. Note that the DeCov loss weight was tuned using grid search at each resolution both with and without Dropout.

We see that Dropout alone (green) usually has the best val accuracy, which is slightly higher than the two losses combined (purple) and a couple points higher than DeCov alone (red) at higher resolutions. At the lowest resolution Dropout alone is tied with DeCov alone. Dropout also reduces overfitting more than DeCov, though both independently reduce overfitting by a large margin – from 59.35% to 14.7% in the case of DeCov @ 128x128.

Finally, we test our new regularizer on ILSVRC 2012 with one more architecture – the Network in Network (Lin et al., 2014a).[3] This architecture is fully convolutional: it contains 4 convolutional layers, with 96, 256, 384, and 1024 feature maps, respectively. Between each of these layers and after the last are two convolutional layers which have 1x1 kernels, which further process each feature map output by the main convolutional layers before being fed into the next layer. To produce 1000 softmax activations, 1000 feature maps are averaged over spatial locations to produce one feature vector. We applied DeCov to these average pooled feature vectors.

Interestingly, this architecture has much less overfitting than AlexNet. However, adding a DeCov loss still decreases overfitting substantially and improves validation accuracy. There is a small boost in performance on validation accuracies and a significant decrease of $\sim$3% (for top 1) and $\sim$2% (for top 5) in the train - val gap.

[1] Using CuDNNv3, AlexNet with 128x128 inputs takes 103ms averaged over 50 runs to compute a forward and backward pass. For 256x256 images this time is 449ms.

[2] At train time crops are sampled and mirrored randomly. At test time only the center 227x227 crop is used.

[3] This is the model provided in the Caffe Model Zoo: https://gist.github.com/mavenlin/d802a5849de39225bcc6

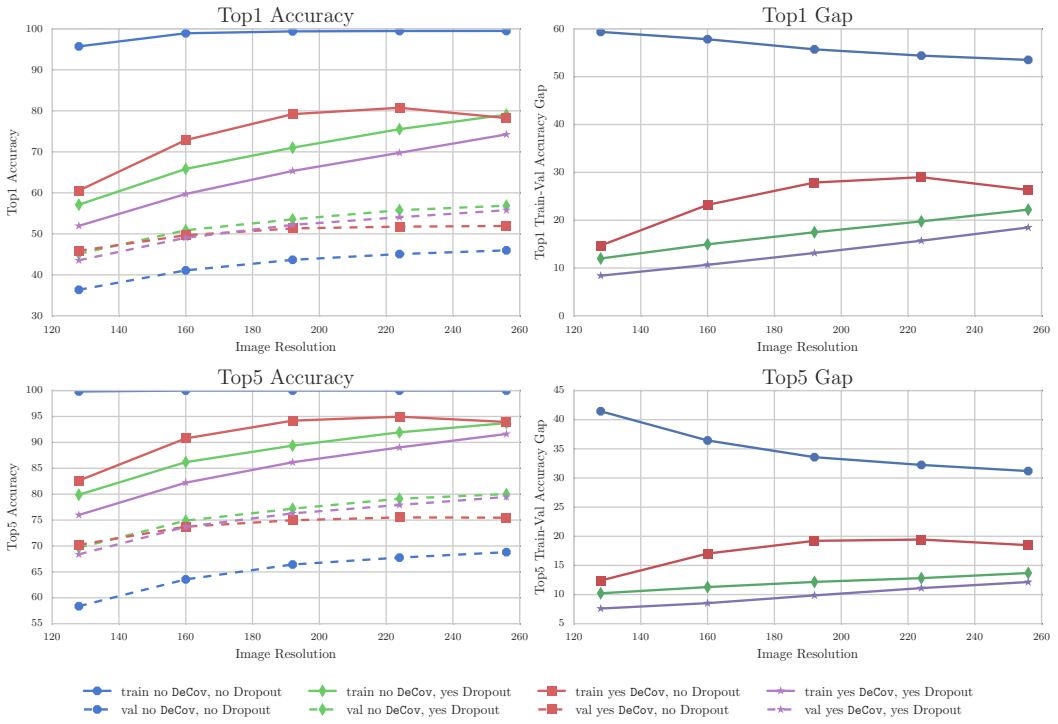


Figure 5: ImageNet classification performance using AlexNet. Plots on the left show training and validation (ILSVRC 2012 validation set) accuracy at different resolutions. Note how all curves have a much lower train-val gap than the (blue) baseline.

DeCov	Dropout	ILSVRC 2012 train top 1	ILSVRC 2012 val top 1	train - val
no	no	71.68	58.67	13.01
no	yes	71.32	58.95	12.37
yes	yes	68.28	**59.08**	**9.20**
yes	no	68.33	58.85	9.48
DeCov	Dropout	ILSVRC 2012 train top 5	ILSVRC 2012 val top 5	train - val
no	no	89.91	81.18	8.73
no	yes	89.63	81.53	8.10
yes	yes	87.99	**81.94**	**6.05**
yes	no	87.88	81.57	**6.05**

Table 5: ImageNet Classification Accuracies with Network in Network

5 Discussion and Conclusion

Fine-tuning. In the experiments we presented, networks were always trained from scratch, but we also tried fine-tuning networks in different scenarios. During our ImageNet experiments we fine-tuned both the Network in Network and AlexNet architectures initialized with parameters that weren't trained with a DeCov loss, but were trained with Dropout. In both cases performance either stayed where it was at fine-tuning initialization or it decreased slightly (within statistical significance). We found similar results when fine-tuning for other tasks like attribute classification (fine-tuning AlexNet) and object detection (Fast RCNN (Girshick, 2015)).

This, along with some cases where combining Dropout and DeCov decreases performance slightly suggest that the DeCov loss may possibly be acting adversarially to activations learned by Dropout. Fine-tuning with DeCov is an interesting direction for future work.

Trends. All of our experiments strongly indicate two clear trends:

1. DeCov reduces overfitting as measured by the gap between train and test performance.
2. DeCov acts as a regularizer: performance with DeCov is always better than performance without either DeCov or Dropout.

To be clear, the results do not support that Dropout can be completely replaced by DeCov, but simply that in a number of scenarios DeCov is a useful alternative and their combination almost always works the best. Our loss clearly has desirable regularization properties at the expense of one extra hyper-parameter to tune.

In this work, we proposed a new `DeCov` loss which explicitly penalizes the covariance between the activations in the same layer of a neural network in an unsupervised fashion. This loss acts as a strong regularizer for deep neural networks, where overfitting is a major problem and Dropout has been required to get large models to generalize well. We show that `DeCov` competes well against Dropout over a range of experiments which investigate different scales, datasets and architectures.

Acknowledgements. This work was supported in part by the following awards to DB: National Science Foundation CAREER award, Army Research Office YIP award, Office of Naval Research grant N00014-14-1-0679, AWS in Education Research Grant, and GPU support by NVIDIA. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Government or any sponsor.

REFERENCES

Andrew, Galen, Arora, Raman, Bilmes, Jeff, and Livescu, Karen. Deep canonical correlation analysis. In *Proceedings of the 30th International Conference on Machine Learning*, pp. 1247–1255, 2013. 4

Antol, Stanislaw, Agrawal, Aishwarya, Lu, Jiasen, Mitchell, Margaret, Batra, Dhruv, Zitnick, C. Lawrence, and Parikh, Devi. Vqa: Visual question answering. In *International Conference on Computer Vision (ICCV)*, 2015. 1

Barlow, Horace B. Possible principles underlying the transformations of sensory messages. 1961. 3

Bengio, Yoshua and Bergstra, James S. Slow, decorrelated features for pretraining complex cell-like networks. In *Advances in neural information processing systems*, pp. 99–107, 2009. 3

Breiman, Leo. Bagging predictors. *Machine learning*, 24(2):123–140, 1996. 2

Chandar, Sarath, Khapra, Mitesh M, Larochelle, Hugo, and Ravindran, Balaraman. Correlational neural networks. *arXiv preprint arXiv:1504.07225*, 2015. 4

Chen, Xinlei and Zitnick, C Lawrence. Mind's eye: A recurrent visual representation for image caption generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2422–2431, 2015. 1

Cheung, Brian, Livezey, Jesse A., Bansal, Arjun K., and Olshausen, Bruno A. Discovering hidden factors of variation in deep networks. *Proceedings of the International Conference on Learning Representations (ICLR)*, abs/1412.6583, 2014. URL http://arxiv.org/abs/1412.6583. 4

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009. 1, 2

Donahue, Jeff, Jia, Yangqing, Vinyals, Oriol, Hoffman, Judy, Zhang, Ning, Tzeng, Eric, and Darrell, Trevor. Decaf: A deep convolutional activation feature for generic visual recognition. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2014. 1

Girshick, Ross. Fast r-cnn. In *International Conference on Computer Vision (ICCV)*, 2015. 9

Girshick, Ross, Donahue, Jeff, Darrell, Trevor, and Malik, Jagannath. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pp. 580–587. IEEE, 2014. 1

Glorot, Xavier and Bengio, Yoshua. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pp. 249–256, 2010. 6

Goodfellow, Ian J, Warde-Farley, David, Mirza, Mehdi, Courville, Aaron, and Bengio, Yoshua. Maxout networks. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2013. 1

Hansen, Lars Kai and Salamon, Peter. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10):993–1001, 1990. 2

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pp. 448–456, 2015. URL http://jmlr.org/proceedings/papers/v37/ioffe15.html. 4

Jia, Yangqing. Caffe: An open source convolutional architecture for fast feature embedding. http://caffe.berkeleyvision.org/, 2013. 5

Krizhevsky, Alex and Hinton, Geoffrey. Learning multiple layers of features from tiny images. *Computer Science Department, University of Toronto, Tech. Rep*, 1(4):7, 2009. 2, 6, 7

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoff. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. URL http://books.nips.cc/papers/files/nips25/NIPS2012_0534.pdf. 1, 2, 7

LeCun, Yann, Jackel, LD, Bottou, L, Brunot, A, Cortes, C, Denker, JS, Drucker, H, Guyon, I, Muller, UA, Sackinger, E, et al. Comparison of learning algorithms for handwritten digit recognition. In *International conference on artificial neural networks*, volume 60, pp. 53–60, 1995. 2, 5

Lin, Min, Chen, Qiang, and Yan, Shuicheng. Network in network. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014a. 2, 8

Lin, Tsung-Yi, Maire, Michael, Belongie, Serge, Hays, James, Perona, Pietro, Ramanan, Deva, Dollï£¡r, Piotr, and Zitnick, C. Lawrence. Microsoft COCO: Common objects in context, 2014b. 1

Linsker, Ralph. Self-organization in a perceptual network. *Computer*, 21(3):105–117, 1988. 3

Pearlmutter, Barak A and Hinton, Geoffrey. G-maximization: An unsupervised learning procedure for discovering regularities. In *AIP conference proceedings*, volume 151, pp. 333–338. American Institute of Physics, 1986. 3

Perrone, Michael P. and Cooper, Leaon N. When networks disagree: Ensemble methods for hybrid neural networks. In *Tech Report*, pp. 126–142. Chapman and Hall, 1993. 2

Schmidhuber, Jürgen. Learning factorial codes by predictability minimization. *Neural Computation*, 4(6): 863–879, 1992. 3

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15 (1):1929–1958, 2014. 1, 2, 4, 6

Tibshirani, Robert. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267 – 288, 1996. 1

Tikhonov, Andrey Nikolayevich. On the stability of inverse problems. *Dokl. Akad. Nauk SSSR*, pp. 195–198, 1943. 1

Vinyals, Oriol, Toshev, Alexander, Bengio, Samy, and Erhan, Dumitru. Show and tell: A neural image caption generator. In *Computer Vision and Pattern Recognition*, 2015. URL http://arxiv.org/abs/1411.4555. 1

Wan, Li, Zeiler, Matthew, Zhang, Sixin, Cun, Yann L, and Fergus, Rob. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp. 1058–1066, 2013. 1

Yosinski, Jason, Clune, Jeff, Bengio, Yoshua, and Lipson, Hod. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, pp. 3320–3328, 2014. 1

Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., and Oliva, A. Learning deep features for scene recognition using places database. In *NIPS*, 2014. 1

APPENDICES

A DETAILS OF THE BIAS IN THE MNIST EXPERIMENT

Recall that in Section 4.1 we generate biased pairs of MNIST digits by defining

$$P(l) = 0.1 \text{ and } P(r|l) = \begin{cases} 0 & \text{if } l \in \{0,\dots,4\} \text{ and } r \in \{0,\dots,4\} \\ 0.2 & \text{if } l \in \{0,\dots,4\} \text{ and } r \in \{5,\dots,9\} \\ 0.1 & \text{if } l \in \{5,\dots,9\} \end{cases} \tag{6}$$

and sampling left then right digits. To show that this creates a larger bias on the right than on the left, we show there is more uncertainty about left digits given right ones than right ones given left ones. That is, we show the conditional entropy $H(l|r)$ is greater than $H(r|l)$.

To compute the conditional entropies, we first derive

$$P(r) = \sum_l P(r|l)P(l) = \begin{cases} 0.05 & \text{if } r \in \{0,\dots,4\} \\ 0.15 & \text{if } r \in \{5,\dots,9\} \end{cases} \tag{7}$$

and

$$P(l|r) = \frac{P(r|l)P(l)}{P(r)} = \begin{cases} 0 & \text{if } l \in \{0,\dots,4\} \text{ and } r \in \{0,\dots,4\} \\ \frac{2}{15} & \text{if } l \in \{0,\dots,4\} \text{ and } r \in \{5,\dots,9\} \\ \frac{3}{15} & \text{if } l \in \{5,\dots,9\} \text{ and } r \in \{0,\dots,4\} \\ \frac{1}{15} & \text{if } l \in \{5,\dots,9\} \text{ and } r \in \{5,\dots,9\} \end{cases}. \tag{8}$$

Using the convention $0 \log 0 = 0$, we can now compute

$$H(l|r) = -\sum_r P(r) \sum_l P(l|r) \log P(l|r) \approx 2.0868 \tag{9}$$

$$H(r|l) = -\sum_l P(l) \sum_r P(r|l) \log P(r|l) \approx 1.9560 \tag{10}$$

$$\tag{11}$$