2016 IEEE 16th International Conference on Data Mining

Regularizing Deep Convolutional Neural Networks with a Structured Decorrelation Constraint

Wei Xiong*, Bo Du*, Lefei Zhang†, Ruimin Hu‡, and Dacheng Tao§
*Computer School of Wuhan University, National Engineering Research Center for Multimedia Software, Wuhan, China
and Collaborative Innovation Center of Geospatial Technology, China
Corresponding author: Bo Du, remoteking@whu.edu.cn
Email: wxiong@whu.edu.cn, remoteking@whu.edu.cn
†Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China
Email: cslfzhang@comp.polyu.edu.hk
‡National Engineering Research Center for Multimedia Software, Computer School of Wuhan University, China
and Collaborative Innovation Center of Geospatial Technology, China
Email: hrm1964@163.com
‡Centre for Quantum Computation & Intelligent Systems
University of Technology, Sydney, NSW 2007, Australia
Email: dacheng.tao@uts.edu.au

***Abstract*—Deep convolutional networks have achieved successful performance in data mining field. However, training large networks still remains a challenge, as the training data may be insufficient and the model can easily get overfitted. Hence the training process is usually combined with a model regularization. Typical regularizers include weight decay, Dropout, etc. In this paper, we propose a novel regularizer, named *Structured Decorrelation Constraint* (SDC), which is applied to the activations of the hidden layers to prevent overfitting and achieve better generalization. SDC impels the network to learn structured representations by grouping the hidden units and encouraging the units within the same group to have strong connections during the training procedure. Meanwhile, it forces the units in different groups to learn non-redundant representations by minimizing the cross-covariance between them. Compared with Dropout, SDC reduces the co-adaptions between the hidden units in an explicit way. Besides, we propose a novel approach called *Reg-Conv* that can help SDC to regularize the complex convolutional layers. Experiments on extensive datasets show that SDC significantly reduces overfitting and yields very meaningful improvements on classification performance (on CIFAR-10 6.22% accuracy promotion and on CIFAR-100 9.63% promotion).**

***Keywords*-Convolutional Networks, Overfitting, Decorrelation.**

I. Introduction

Deep networks provide a fresh eye to mine large scale datasets, especially the convolutional neural networks (CNNs) [1, 2]. With elaborately designed networks, representative features, which contain the inherent structures of the input data, can be efficiently mined out from the complicated raw data for various data mining tasks such as multimedia retrieval [3], feature engineering [4], recommendation system [5, 6], classification [7], information estimation [8], etc. However, efficiently training deep networks remains a big challenge. Deep networks are usually composed of multiple layers containing a tremendous amount of trainable parameters, leading to some serious problems like vanishing gradient in back propagation [9], diminishing feature reuse in forward propagation [10] and overfitting [11, 12]. Among these problems, overfitting makes the model hard to fully use the trainable parameters, greatly limiting the final performance of the networks.

To reduce overfitting, several regularizers, optimization methods and network structures have been proposed, including Early Stopping [13], Weight Decay [14], Model Ensemble [15–17], Data Augmentation [18], Dropout [19, 20], DropConnect [21], Maxout [22], Batch Normalization [23], Residual Networks [10], etc. These methods enhance the network's generalization capacity in different ways. Data augmentation deals with the input data; Batch Normalization and Residual Networks are novel network structures or normalization of the medium layers; Early Stopping focuses on the way the parameters are updated; Model Ensemble copes with averaging the results of multiple models. These methods are reported to have an effect of improving the generalization of the network, but all in an implicit way. Besides these methods, some methods aim directly to reduce the networks' overfitting, including Weight Decay, Dropout, DropConnect, etc.

Hinton et al. [19] took a further investigation into the essence of neural networks' overfitting. They observed that overfitting can be relieved by reducing the co-adaption between neurons [19]. The neurons' co-adaption means one neuron may activate depending on the state of the other neurons in the same layer. It makes the feature representation

2374-8486/16 $31.00 © 2016 IEEE
DOI 10.1109/ICDM.2016.66

learned by these neurons redundant and inefficient. They proposed Dropout to reduce co-adaption. Dropout works on the neural units of the hidden layer. It reduces the co-adaption of neurons by randomly omitting a part of the activations during each training iteration. Though Dropout can effectively avoid the co-adaption of neurons, it works in an implicit way.

Can we reduce the co-adaptions between the neurons in an explicit way? In this paper, we take a further step on to reduce overfitting. We propose a novel regularizer, named Structured Decorrelation Constraint, imposed on the hidden layers of deep networks to improve their generalization. SDC depicts the co-adaptions of neurons within the same layer by an explicit formula, i.e., using the covariance/correlation of two units's responses to describe the units' co-adaption, hence the objective function of minimizing co-adaptions can be directly added to the cross-entropy loss of the network's classification layer. Though Dropout can prevent the neurons from getting co-adapted, it works in an implicit way by training a random portion of neurons each time. Its capability of reducing co-adaption cannot be well evaluated. However, by explicitly express the co-adaption, we can reduce overfitting more straight forwardly and can obtain more optimal solution for the parameters. As a type of loss function, our method can be easily implemented under the framework of mini-batch gradient descent [24–26]. By imposing SDC on the hidden layers, the regularized network is able to learn decorrelated representations, which contains much less redundancy compared with the network without our regularizer.

Another major challenge to reduce the co-adaptions exists when we re-consider the relationship between the hidden units within the same layer. For a certain layer, is the co-adaption between arbitrary hidden units in this layer harmful to the generalization of the network? Is it the optimal choice to decorrelate all the neurons within a layer for boosting the performance of the network? In some cases, neurons may still need to be correlated with each other to jointly perform a certain functionality. This insight is motivated by the performance of Dropout [19]. In Dropout, when dropping some hidden units during an iteration, other neurons may still have the chance to be co-adapted. The typical drop rate is 0.5. When the dropping rate increases, however, the performance of the network isn't continuously getting better. This phenomenon motivates us to think that the correlations between some of the neurons may help to construct a more promising regularizer.

To address these concerns, the proposed regularizer should not only express the co-adaptions explicitly, but also encourage the hidden units to learn a prior structure. Specifically, hidden units in the regularized layer are enforced to activate in groups, and neurons within the group are forced to have relatively high correlations and similarities, while neurons in different groups are quite decorrelated and dissimilar. To guarantee this, we first divide the units into groups, then maximize the correlations of units within each group and minimize the correlations of those in different groups simultaneously by optimizing a unified objective function. In this way, we enforce the hidden neurons to behave obeying the group structure. The network can take advantages of the correlations between the neurons while reducing the co-adaptions of a portion of the neurons, making the hidden layer of the network to learn structured and decorrelated representations that are beneficial for the network's performance.

The proposed regularizer can be easily performed on the fully connected layers of the convolutional neural networks. However, besides the fully connected layers, there is also a considerable amount of trainable parameters in the convolutional layers, which can also lead to overfitting. If we regard each feature in the feature map of a convolutional layer as a unique neuron, minimizing the co-adaptions of all the neurons will cost a very large amount of computation resource. In addition, features within the same convolutional feature map may have the inherent correlation, due to the unique properties of the convolution and pooling structures. We then propose a novel approach called Reg-Conv to help the explicit regularizers to regularize the convolutional and pooling layers much more efficiently and effectively. The main idea of this approach is to degrade the convolutional layer to a fully connected layer, then we regularize the resulting fully connected layer, in reverse the convolutional layer is regularized by propagating backwards the gradients of the fully connected layer.

The main contributions of our paper are as follows. 1) We propose a novel regularizer called *Structured Decorrelation Constraint*, which can reduce the co-adaptions of the hidden units in an explicit way. SDC divides the units in the hidden layer into groups, and minimizes the correlations of units in different groups so as to avoid co-adaptations between them, while preserving the correlation between hidden units within the same group to encourage them to have a grouped structure. 2) We propose a *Reg-Conv* method that can efficiently apply explicit regularizers to the convolutional layers during the learning process, so that the feature maps in the convolutional layers can also learn the structured and decorrelated features from complicated data, which is beneficial for boosting the performance of the network.

The remainder of this paper is organized as follows. Related works are introduced in Section II and details of the proposed method are provided in Section III. Experimental results and analysis are presented in Section IV, followed by our main conclusions in Section V.

II. Related Work

Typical Regularizers. The generalization of the network can be improved from various aspects, including augmenting

the training data, averaging the models, using more complicated layers, using more efficient training strategy, etc. In this part we overview some of the related regularizers used in the deep networks. Weight decay [14] is a way to constrain the weights of the net in a limited parameter space so as to decrease the model's complexity. It's widely adopted now. Dropout [19] is a very powerful regularizer imposed on the hidden units' responses. During each training iteration, it randomly omits part of the neurons, so that the hidden neurons won't activate based on the others' states. In this way, Dropout prevents the neurons from getting adapted and reduces overfitting. Similar to Dropout, DropConnect [21] omits part of the connections between neurons in adjacent layers during each training iteration, to prevent the co-adaptions between the weights and also achieve an effect of model averaging. DropConnect also regularizes the network in an implicit way, while our method works in an explicit way. DropConnect deals with the weights between the adjacent layers, which differs greatly from our method, which is imposed on the activations of certain hidden layers.

Recently, another regularizer that is quite related to our method has been proposed. Aiming to reduce overfitting, Cogswell et al. [27] proposed an interesting method called DeCov, which reduced the correlations between the neurons by minimizing the their cross-covariance over a batch of samples. DeCov and our method both work on the hidden layers of a deep network. However, our method differs a lot from DeCov in several vital aspects. DeCov constrains the cross-covariance of all the hidden units to be small in order to reduce the co-adaptions within a layer, while we argue that not all the correlations between the units have negative influence on the performance of the network. DeCov achieves a good performance in reducing overfitting, but may not be optimal for improving the performance of the network, as it may destroy some inherent connections between the units. On the contrary, Our method merely encourages a portion of the units to be decorrelated, while preserving the correlations between the other units. The remaining correlations may have positive effect on improving the network's performance. Another difference is that DeCov can't learn structured representation, while our method is capable of learning both decorrelated and structured representations, which are beneficial for boosting the performance of the network. DeCov simply reduces the co-adaptions between the hidden units, our method maintains a prior group structure in the regularized layer, making the network to learn better structures. The third difference between DeCov and our method lies in that DeCov can only be imposed on the fully connected layers, while in this paper we propose an approach that can extend both our regularizer and DeCov to regularizing the convolutional layers. In this way, all types of layers in any convolutional layers can be regularized for better performance.

Correlation Related Methods. Except DeCov and our method, there are other methods that are tightly related with correlations. Deep Canonical Correlation Analysis (Deep CCA) [28] learns flexible non-linear representations with deep networks by maximizing the correlations between the variables. Correlational Neural Networks (CorrNets) [29] apply a similar loss in addition to their original objective functions. It learns correlated features by maximizing the correlation of features from multiple views. CorrNet and CCA are methods that maximize the correlation between the neurons or the latent variables, while our method minimizes the covariance between the units within a hidden layer to learn decorrelated representations.

Bergstra et al. introduces a novel activation function in [30], aiming at learning decorrelated representation for pretraining complex cell-like networks. It's pretraining method provide a good initialization for the hidden units with a decorrelation criterion. In this work, decorrelation is used for initialization, while our method uses decorrelation for reducing overfitting of the network. Another correlation based method similar to ours is the cross-covariance penalty (XCov) proposed by Cheung et al. [31]. The XCov disentangle the class-relevant variables and the latent variables (such as form or style) in deep auto-encoder by minimizing the correlations between the hidden neurons and the neurons in the classification layer. Different from XCov, our method minimizes the covariance between the hidden neurons to better regularize the network.

Structures in CNNs. There are various types of structures in convolutional networks. Except convolution, pooling and fully connected layers, recently, there are other types of structures that can compose powerful networks. Szegedy et al. proposed the GoogLeNet [32], which is composed of multiple Inception blocks. The Inception block has filters of multiple size, hence it can learn multiple level features. He et al. [10] proposed a novel network called residual network, which is composed of layers of residual blocks. Each block is a combination of multiple convolutional layers, along with batch normalization and activation function. Though these structures are powerful, they are predefined. Our method is based on the simple convolutional and fully connected layers. We don't predefine the structure, we merely encourage the network to learn a prior structure. In addition, they are designed for training very deep networks, while our goal is to reduce overfitting.

Besides designing network structures, there are approaches that can learn structures automatically. Feng et al. [33] proposed a novel method for automatically learning aspects of the structure of a deep model in order to improve its performance, especially when labeled training data are scarce. It's similar to our method in that we both learn structures guided by a known prior. However, there method is used to design the network, while our method is used to regularize the hidden layers to reduce overfitting, and we obey different priors. Schulz et al. [34] proposed to adapt

a structured loss function for neural network training which directly maximizes overlap of the prediction with ground truth bounding boxes for object detection. Our method is more general, we aim to reduce overfitting of the networks, which can be used in many tasks.

III. Structured Decorrelation Constraint

In this section, we first use the SDC method to regularize the fully connected layers of a typical convolutional network, then we extend our method to the convolutional and pooling layers.

A. Regularize Fully Connected Layers

Model Definition. We first introduce the *total-loss* of the whole network J. We consider a certain training iteration under the minibatch gradient descent [24] scheme. The network's input is a batch with *N* samples. Without any regularizers imposed on the network, the network's *total-loss* J will be equal to the *cross-entropy-loss* between the network's output and the groundtruth, which is denoted as $J_Y = \frac{1}{N}\sum_n (\hat{y}^n)^{\mathrm{T}} \cdot \log(y^n)$, where $\hat{y}^n$ and y^n are the one-hot vector-form groundtruth and the network's output for the *n-th* sample, respectively.

Our method aims to reduce the co-adaption of neurons in the fully connected layer (FC layer). To this end, we use the covariance to represent the co-adaption, as the lower covariance of two variables tends to mean a less linear correlation between them. When it comes to the neurons's responses, neurons with higher covariance are likely to be highly correlated. So the covariance matrix of the hidden layer's output on a batch of data is calculated. Our goal is to maintain a lower covariance between each pair of hidden units during each iteration. Suppose the output of the fully connected layer is $H \in R^{N\times D}$, where *N* is the size of mini batch; *D* is the number of neurons in the FC layer. We use Cov_{ij} to denote the covariance between the *i-th* and the *j-th* neuron, then we have:

$$Cov_{ij} = \frac{1}{N}\sum_t (h_i^t - \bar{h}_i)(h_j^t - \bar{h}_j) \tag{1}$$

where h_i^t denotes the *t-th* sample of the the *i-th* neuron's response, $\bar{h}_i$ denotes the mean of the *i-th* neuron's response over the *N* samples, $\bar{h}_i = \frac{1}{N}\sum_t h_i^t$. The covariance is constrained to be small by minimizing its Frobenius norm. For the notation convenience, we call the covariance matrix over all the neurons in the hidden layer the *full-covariance-matrix*, denoted as Cov, and call the Frobenius norm of Cov as the *full-covariance-loss*, denoted as J_C, which can be formulated as:

$$J_C = \frac{1}{2}(\|Cov\|_F^2 - \|diag(Cov)\|_2^2). \tag{2}$$

We subtract the diagonal elements of the covariance matrix as the goal is to constrain the covariance between the responses of two different neurons, not to constrain the variance of each neuron. For notation convenience, we let $Cov_{ii} = 0, i = 1, 2, ..., D$. Then we express the *full-covariance-loss* as: $J_C = \frac{1}{2}\|Cov\|_F^2$. By minimizing J_C, neurons in the hidden layer are forced to be uncorrelated, thus the network is less likely to get overfitted.

To constrain the covariance of all pairs of hidden units to be small, we will have $J = J_Y + \gamma \cdot J_C$, where γ is utilized to balance the cross-entropy loss and the covariance loss. However, this objective function ignores the problem that correlations between some neurons may have positive effects on the network. Our method takes advantages of correlations between the hidden units. To realize this, we partition the neurons into groups, our goal is that the neurons in the same group are encouraged to activate simultaneously in correlated manners, and the neurons in different groups are encouraged to behave in very uncorrelated ways.

Suppose we partition the neurons of the FC layer into *K* groups in order and each group has *m* ($m = D/K$) neurons. The *k-th* group is represented as $H^k \in \mathrm{R}^{N\times m}$, which contains the responses of the $(m \times (k-1)+1)$-*th* to the $(m \times k)$-*th* neurons. To achieve our goal, we constrain the covariance of neurons in different groups to be small, while relaxing the constraint upon the covariance of neurons in the same group. We use Cov^{uv} to represent the cross-covariance matrix between group H^u and H^v, where $u, v \in \{1, \ldots, K\}$. The relationship between the matrix Cov^{uv} and the *full-covariance-matrix* Cov can be formulated as: $Cov_{ij}^{uv} = Cov_{(m\times(u-1)+i)(m\times(v-1)+j)}$, where Cov_{ij}^{uv} is the covariance between the *i-th* neuron in group *u* and the *j-th* neuron in group *v*. We minimize each element in the cross-covariance between group H^u and H^v by minimizing the Frobenius norm of the matrix, denoted as $J_C^{uv} = \frac{1}{2}\|Cov^{uv}\|_F^2$. Hence the penalty for regularizing the covariance between all the pairs of groups can be formulated as:

$$J_{between} = \sum_{u\neq v} J_C^{uv} = \frac{1}{2}\sum_{u\neq v}\|Cov^{uv}\|_F^2. \tag{3}$$

Some prior studies demonstrated that without the constraint on the FC layers, the covariance between each two neurons still increases as the network updates using SGD [27]. This phenomenon indicates that neurons in the hidden layers are easily getting co-adapted, which has been empirically verified by Hinton etc. [19]. Hence even if we don't impose any constraint on the neurons inside the group, the neurons belonging to the same group may still have a high covariance. Using J_C to constrain the FC layer leads to very small correlations between neurons inside each group, as the covariance losses between the groups and inside the groups have the same weight (1:1). To preserve the correlations of neurons inside the same group, we introduce a balance factor to reach a trade off. We penalize the correlation of

neurons inside group H_u by minimizing its Frobenius norm, denoted as $J_C^u = \frac{1}{2}\|Cov^{uu}\|_F^2$. We sum this penalty over each group, and call the sum the *inside-covariance-loss*, denoted as J_{inside}:

$$J_{inside} = \sum_u J_C^u = \sum_u \|Cov^{uu}\|_F^2. \tag{4}$$

The total penalty that SDC imposed on the FC layer is a weighted sum of the penalties inside each group and between each group. We call the loss of SDC the *grouped-covariance-loss*, which can be formulated as:

$$J_G = \lambda \cdot J_{inside} + J_{between} \tag{5}$$

where $0 \leq \lambda \leq 1$. When $\lambda = 0$, we don't impose any constraint on the covariance of the neurons inside each group; when $\lambda = 1$, we have $J_G = J_C$. Then the *total-loss* of the network regularized with SDC can be formulated as $J = J_Y + \gamma \cdot J_G$. Fig. 1 illustrates the grouped structure of the FC layer.

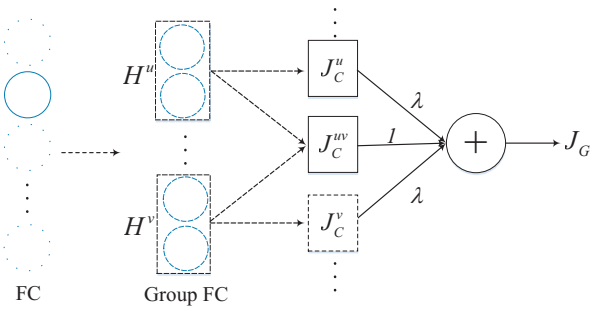


Figure 1. Illustration of the grouped structure in the fully connected layer. The FC layer is firstly divided into groups, then the covariance loss inside the groups and between the groups are calculated then summed.

Instead of just dividing the neurons in the FC layer into groups, we can relax this setting by switching non-overlapping grouping to overlapping grouping, i.e., each two adjacent groups may include the same neurons. We can realize this structure by simply changing the group mask *M*. Suppose each group contains m neurons, and the adjacent groups shares *s* neurons ($s < m$), we will have $K = \frac{D-s}{m-s}$ groups. By overlapping the groups, the adjacent groups are allowed to be correlated to some degree.

Efficient Implementation. Our structured loss J_G looks to be time-consuming, as we have to first divide the neurons, then calculate the covariance between each pair of groups. In this section, we provide an efficient way to implement our method.

In fact, the grouped-covariance-loss J_G can be interpreted as the Frobenius norm of a *grouped-covariance-matrix* Cov^G, which can be generated by element-wisely producting a mask matrix *M* to the *full-covariance-matrix* Cov. It can be formulated as $J_G = \frac{1}{2}\left\|Cov^G\right\|_F^2$, where $Cov^G = M \circ Cov$, $M \in \mathrm{R}^{D\times D}$, and "$\circ$" denotes the element-wise product. Then we have:

$$M_{ij} = \begin{cases} \sqrt{\lambda}, & if\ (k-1)\times m + 1 \leq i,j \leq k \times m, \\ & k = 1, 2, ..., K \\ 1, & otherwise \end{cases} \tag{6}$$

where *m* is the number of neurons in each group, *K* is the number of groups. This transformation makes the *grouped-covariance-loss* much more efficient to implement, as element-wise product consumes much less time to implement than calculating the cross-covariance between each group one by one.

B. Reg-Conv: Regularize Convolutional Layers

It should be noted that the related method DeCov can also regularize the fully connected layers with a definite objective function. However, one drawback of DeCov is that it can only regularize the fully connected layers of the convolutional networks. DeCov can't be applied to the convolutional layers directly, as there are too many features in the convolutional layer. Calculating the *grouped-covariance-loss* J_G or the *full-covariance-loss* J_C on the convolutional feature maps directly will lead to a very large covariance matrix if we take each feature as an individual variable. Another reason that we shouldn't impose the constraint on the convolutional features directly is that it may break the inner structure of the convolutional layers, as features within a feature map are supposed to be correlated. The same problem goes with SDC.

In this section, we propose a novel approach named Reg-Conv to effectively apply DeCov and SDC to the features in the convolutional layers, without breaking the inherent correlations between the features in the meanwhile. Considering a specific convolutional layer $S \in R^{N\times c\times a\times b}$ to be regularized, where *a*, *b*, *c*, *N* are the height, width, number of channels of the feature maps, and *N* denotes to the size of the mini batch. We first reduce the number of features in this layer by pooling the layer with global average pooling (GAP) [35], the resulting features are $P \in R^{N\times c}$. We regard *P* as *N* observations of a vector composed of *c* possibly correlated variables. Then similar to the approach in the FC layer, we impose the proposed SDC on *P*. The whole process is illustrated in Fig. 2.

Reg-Conv approach can not only be applied to the convolutional layers, but all the layers that is in the form of convolutional feature maps, including the pooling layer, the activation layer, etc. Note that the way we use the global average pooling is quite different from the previous methods. In the literature, the GAP is usually used as one of the layers embedded into the network connecting the previous and next layers. However, we use the GAP layer just to condense the convolutional layers and common pooling layers, not to use it to connect the networks. To the best of our knowledge, it

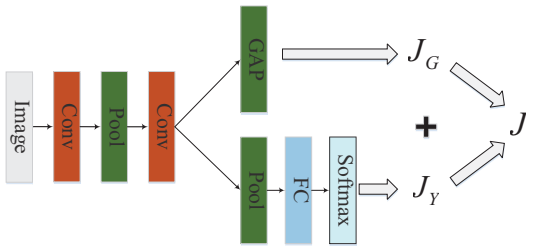


Figure 2. The structure of constraining the convolutional layer. During training, the convolutional layer (Conv) is mapped to a average pooling layer (GAP), then we decorrelate the GAP layer, which in turn has an effect of decorrelation on the Conv layer. J_G denotes the group-covariance-loss of the GAP layer, J_Y is the cross-entropy loss of the classification layer.

is the first attempt to use the global average pooling method to regularize the convolutional layer of large networks.

Interpretation. The convolutional feature maps *S* can be decorrelated by imposing the decorrelation constraint on the features of the GAP layer *P*. It's effective because if two feature maps in the convolutional layer are highly correlated, the corresponding neurons in the GAP layer should also be correlated. We further study this conclusion by calculating the derivation of features in the convolutional layer. First, we consider the derivation of the GAP layer. We consider the derivative Δp_i^n on the *i-th* neuron's response for the *n-th* sample p_i^n in the pooling layer:

$$\Delta p_i^n = \frac{\partial J_G}{\partial p_i^n} = \frac{2}{N} \sum_j Cov_{ij}^G \left(p_j^n - \bar{p}_j \right). \tag{7}$$

Then the derivative is propagated back to the convolutional layer through an upsampling [36] operation. The derivative of feature in the *i-th* channel for the *n-th* sample is calculated as:

$$\Delta p_i^n(y,z) = \frac{1}{a \times b} \frac{\partial J_G}{\partial p_i^n} = \frac{2}{N \times a \times b} \sum_j Cov_{ij}^G \left(p_j^n - \bar{p}_j \right). \tag{8}$$

By decorrelating the pooling layer, we can propagate the relevant gradient back to the convolutional layer, which has the same form compared with the derivative of features in the pooling layer (the only difference is the scale). The gradients are supposed to have some positive effects on decorrelating the convolutional layer, since gradients with the same form can decorrelate the pooling layer effectively.

IV. Experiments

In this section, we conduct a series of experiments with a convolutional neural network on several datasets: CIFAR-10, CIFAR-100 [37] and SVHN [38]. We evaluate the performance of the proposed regularizer by imposing it on the convolutional layers and fully connected layers. We compare our methods with the most related regularization methods: Dropout and DeCov, as both methods aim directly to reduce the co-adaption between the hidden neurons and force the network to learn non-redundant representations.

In the following parts, we first introduce the datasets and implementation details, then evaluate the regularizers' influence on the fully connected layers of the CNN. After that we evaluate the regularizers by comparing the classification performance of the network directly, then compare the performance of regularizing the convolutional layers.

A. Datasets

CIFAR datasets. The CIFAR-10 dataset consists of 60000 natural scene images in 10 classes with 6000 images per class. It has been split to 50000 training images and 10000 test images. Each image is an RGB image with a size of 32 $\times$ 32. The CIFAR-100 dataset is just like the CIFAR-10 dataset, except that it has 100 classes containing 600 images each. There are 500 training images and 100 testing images per class.

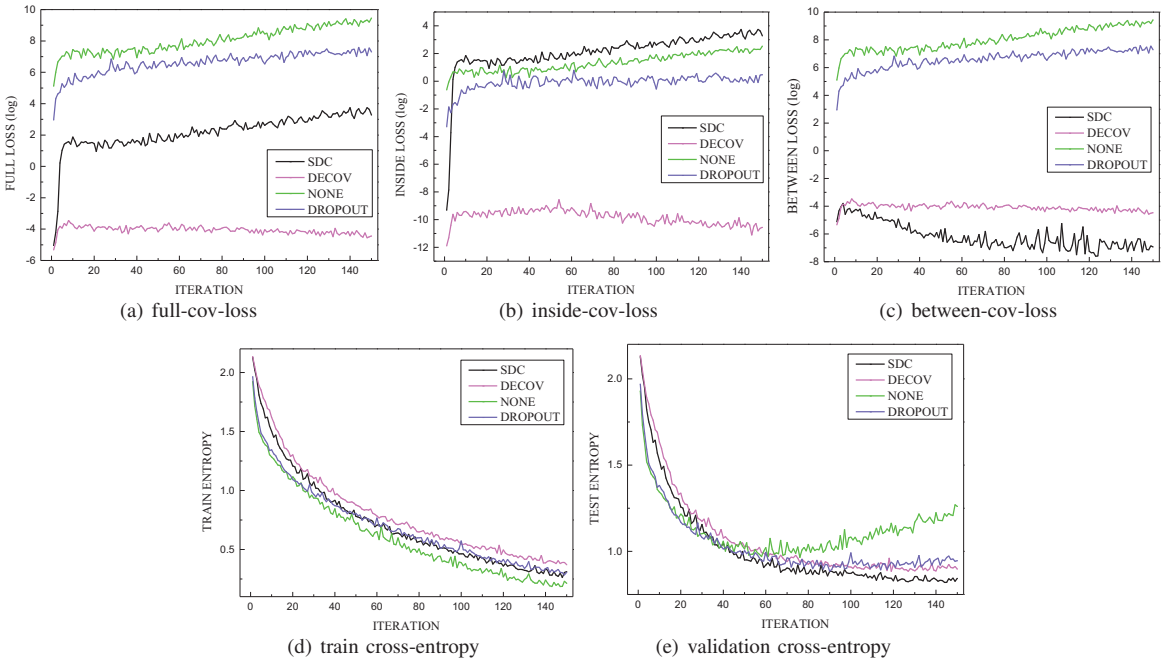
SVHN datasets. The SVHN dataset is a much larger dataset. It contains 73257 training samples, 26032 testing samples, and 531131 extra training samples. We preprocess the data as in the previous methods [38], i.e., selecting 400 samples per category from the training set as well as 200 samples per category from the extra set, using these 6000 images for validation, and the remaining 598388 images as training samples.

B. Implementation Details

For CIFAR-10 and SVHN datasets, we use Caffe's quick CIFAR-10 architecture, i.e., 3 convolutional layers, each convolutional layer is followed with a pooling layer, 1 hidden fully connected layer (FC layer) which has 64 neurons and a softmax layer. The activation function of the FC layer is linear. The only difference of the network on CIFAR-100 is that it uses 100 nodes in the softmax layer instead of 10. We don't use a very large network since we are evaluating the performance of the regularizers, not aiming to achieve the state of the art performance. We use mini-batch gradient descent with momentum to train the network. The initial learning rate is 0.01 for CIFAR-10 and CIFAR-100, and 0.001 for SVHN. The momentum is set as 0.9. We use a mini batch size of 64 for all the experiments. For all the datasets, we use only global contrast normalization as the preprocessing. For the CIFAR datasets, 5% of the training samples are used as the validation set for model selection. The experiments are implemented with Theano [39] and Lasagne.

C. Regularize FC layer

First, we evaluate the regularizers' effects on reducing the co-adaption in the fully connected layers on CIFAR-10 dataset. We divide the 64 hidden neurons in the FC layer into 8 equal groups without overlapping. After each training epoch, we calculate the average *full-covariance-loss* J_C, the



(a) full-cov-loss (b) inside-cov-loss (c) between-cov-loss

(d) train cross-entropy (e) validation cross-entropy

Figure 3. Network's losses with different regularizers. Top three figures demonstrate the covariance losses, bottom two figures demonstrate the network's cross-entropy loss.

inside-covariance-loss J_{inside} and the *between-covariance-loss* $J_{between}$ of the FC layer on the validation set. The results are demonstrated in Fig. 3(a), 3(b) and 3(c). In this figure, we compare the losses with no regularizers(only weight decay is used), only Dropout, only DeCov and only SDC imposed on the FC layer. The Dropout rate is set to 0.5. The weight of the covariance loss is set to 1. In our method, the additional hyper-parameter λ is set to 0. Weight decay is used for all the parameters in the network and the coefficient of this term is kept the same for all methods. Except weight decay and the three regularizers, no more regularizers are added to the network.

Fig. 3(a) shows that the covariance loss of the hidden units increases rapidly as the model iterates, indicating that neurons can easily become co-adapted during training. The three regularization methods significantly reduce the covariance of the hidden neurons, while DeCov and our SDC perform much better than Dropout as they work in an explicit way.

We focus on the performance of our SDC method. The *full-covariance-loss* achieved by the SDC is larger than DeCov, but lower than Dropout. This is because we have relaxed the covariance constraint inside each neuron group. From Fig. 3(b), the SDC achieves the largest *inside-covariance-loss*, which is even larger than the loss without any regularizer. On the contrary, it achieves the smallest *between-covariance-loss*, as shown in Fig. 3(c). These results indicate that: 1). the proposed SDC has the best performance for reducing the cross-correlation between the neurons in different groups among all the compared methods; 2). SDC effectively keeps the correlation between neurons inside each group.

We further evaluate the regularizers' influence on the cross-entropy loss of the network. After each training epoch, we calculate the average cross-entropy loss of the network on the train and validation samples, which are correspondingly demonstrated in Fig. 3(d) and Fig. 3(e). The network without any regularizers on the FC layer overfits easily, as it has the smallest cross-entropy loss on the training data, while the cross-entropy loss on the validation set is very large. The SDC method has both smaller train cross-entropy loss and validation cross-entropy loss compared to DeCov. The reason may be that the group structure behind the network has been effectively learned with the help of our SDC method. The neurons in the same group are forced to be correlated and activate in similar ways, and they can learn structured and decorrelated features. Note that the DeCov and Dropout methods decrease the validation cross-entropy loss by a large margin compared to the results with no regularizers. Their performances on reducing overfitting is impressive. However, our method takes a step on to dig the inner structures in the hidden neurons, continuing decreasing the validation loss by a considerable margin.

D. Classification Performance

Though we have proved that the proposed SDC method has a good capacity on reducing the validation cross-entropy loss, its influence on classification tasks haven't been directly

evaluated. We utilize a CNN with different regularizers to classify the image datasets CIFAR-10, CIFAR-100 and SVHN. Besides imposing the single regularizer on the FC layer, we simultaneously apply Dropout along with SDC or DeCov to evaluate the combined performance of these regularizers. To evaluate the effect of overlapping grouping, we also show the results with overlapped SDC. On all datasets, each pair of adjacent groups overlap two hidden units. This hyper-parameter is chosen by many trials on the validation set.

Results on CIFAR Datasets. Table I demonstrates the classification results of the CNN on CIFAR-10 and CIFAR-100. On both datasets, we show the classification accuracy on the training data and the standard test data. We also show the gap between train and test accuracy to compare the performance of reducing overfitting. We run 5 times for each method and average the results. Note that for CIFAR-100 dataset, we don't show the standard deviation, as we adopt the results reported in [27] directly for None regularizers (denoted as "NONE" in the table), only Dropout, only DeCov and DeCov+Dropout. [27] didn't provide the standard deviation on CIFAR-100 dataset.

On CIFAR-10 dataset, our method promotes the test classification accuracy by **6.22**% compared to the performance without any regularizers. On the much more challenging CIFAR-100 dataset, there is even a **9.63**% promotion in accuracy. These results demonstrate the effectiveness of our method on reducing overfitting of the network. DeCov and Dropout also achieve very impressive performance on decreasing the training accuracy and promoting the test accuracy. DeCov method achieves the best train-test accuracy gap on CIFAR-100 dataset, larger than that of our method. This may be because the goal of DeCov is to decorrelate the hidden neurons, while our method sacrifice some decorrelation ability to enforce the structures among the neurons. On both CIFAR-10 and CIFAR-100 datasets, the test accuracies of our method are better than that of DeCov. Compared with DeCov, our method has a 1.7% and 3% promotion in test accuracy on CIFAR-10 and CIFAR-100 datasets respectively, the results further prove the superiority of the proposed method. Note that combined with Dropout, both DeCov and our method achieve a slightly better performance (in most cases) in promoting the test accuracy, indicating the flexibility of our regularizer. The overlapping of groups also boosts the performance of SDC on CIFAR-10 and CIFAR-100, but not by a large margin.

Results on SVHN. We then conduct the same experiments on the much larger SVHN dataset, and the results are shown in Table II. Classifying SVHN is a more challenging task since it contains more than 500000 training samples. On this dataset, our regularizer achieves both the highest test accuracy and the largest train-test gap. Though the CNN has achieved a very high accuracy on the test dataset without any regularizers, i.e., 95.50%, we can still improve it to 97.31%, which is a 1.81% absolute margin. The train-test gap is also largely improved. The network with no regularizers obtains a gap of 4.40%, while our SDC method reduces this gap to 1.65%. All these results indicate that the proposed regularizer has a consistent superiority over the conventional methods.

Results on Small Dataset. To evaluate the performance of the regularizers more efficiently, we reduce the number of training samples, as the large network is more likely to get over-fitted when there is a lack of labeled training data. On CIFAR-10 and CIFAR-100, we randomly select $\{2000, 4000, 6000, 8000, 10000\}$ images from the 50000 *train* data to train the same CNN used in the previous sections and test it on the standard test set. The number of samples in each class are kept equal. Fig. 4 shows the classification results on data of different sizes. The SDC presents an obvious superiority over the other methods especially DeCov on smaller datasets of almost all sizes, owing to the structures learned in the neurons — the neurons inside the same group are correlated and similar while different groups of neurons act uncorrelated to avoid co-adaption.

E. Regularize Convolutional Layers

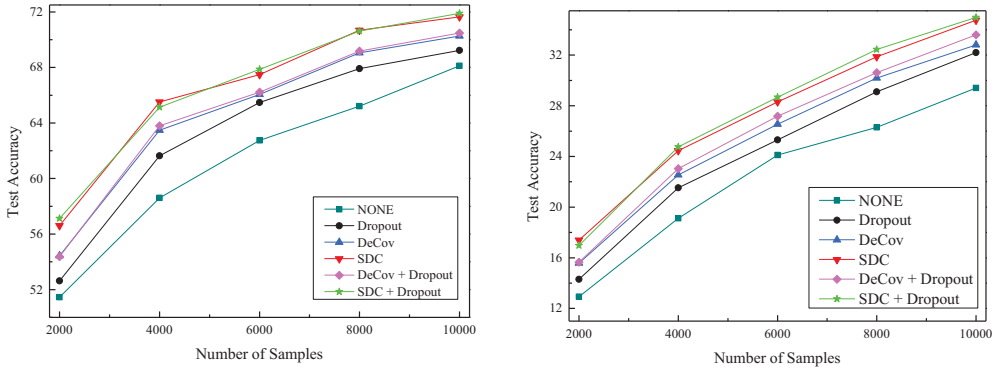
We further evaluate the performance of the proposed RegConv method that can efficiently apply SDC and DeCov to the convolutional layers. We train the Caffe's quick CIFAR-10 model on the full training samples of CIFAR-10 dataset and test the model on the test set. We impose a single regularizer on the second (denoted as *pool 2* layer) and the third pooling layer (denoted as *pool 3* layer). Specifically, we first pool the 3-Dimension feature maps of the pooling layer into a vector (in fact the size of each feature map is 1×1) using the Global Average Pooling. Then we impose the regularizer on the pooled vector. The FC layer is not imposed on any regularizer. Since the weight decay term has an impact on the training and testing accuracy, we fix the coefficient of this term to 0.001 for all methods. The other hyper-parameters such as γ, λ, number of groups and number of neurons overlapped between groups are carefully chosen for each method based on the *val* set.

Table III shows the classification results. Imposing the regularizers on the global average pooling layer of *pool 2* layer or *pool 3* layer successfully boosts the classification performance on the test data and decreases the training accuracy. The results indicate that both DeCov and SDC can be imposed on the convolutional layers using our approach. Though the global average pooling layer is not followed by any layers nor connected to the final output of the network, by regularizing the global average pooling layer, the parameters in the convolutional and pooling layers related to the GAP layer are still successfully regularized.

Compared to the results without regularizer, imposing the SDC constraint on the GAP layers following pool 2 layer and the pool 3 layer boosts the test performance by 5.50% and

Table I
THE CLASSIFICATION RESULTS ON CIFAR-10 AND CIFAR-100 DATASETS.

Methods	CIFAR-10			CIFAR-100		
	train	test	train-test	train	test	train-test
NONE [27]	100.0 ± 0.00	75.24 ± 0.27	24.77 ± 0.27	99.77	38.52	61.25
Dropout [27]	99.10 ± 0.17	77.45 ± 0.21	21.65 ± 0.22	87.35	43.55	43.80
DeCov [27]	88.78 ± 0.23	79.72 ± 0.14	9.06 ± 0.22	77.92	40.34	37.58
DeCov+Dropout [27]	87.78 ± 0.08	79.75 ± 0.17	8.04 ± 0.16	72.53	45.10	**27.43**
SDC	91.21 ± 0.40	80.80 ± 0.33	10.41 ± 0.35	82.11	47.60	34.51
SDC+Dropout	89.81 ± 0.30	81.32 ± 0.20	8.48 ± 0.22	78.74	47.93	30.81
Overlapped SDC	90.90 ± 0.27	81.07 ± 0.23	9.83 ± 0.25	81.42	47.86	33.56
Overlapped SDC+Dropout	89.11 ± 0.45	**81.46 ± 0.31**	**7.64 ± 0.38**	77.39	**48.15**	39.24



(a) CIFAR-10 (b) CIFAR-100

Figure 4. The classification results on data of different sizes.

Table II
THE CLASSIFICATION RESULTS ON SVHN.

Method	train	test	train-test
NONE	99.90 ± 0.02	95.50 ± 0.13	4.40 ± 0.06
Dropout	99.64 ± 0.08	95.71 ± 0.11	3.93 ± 0.09
DeCov	98.91 ± 0.05	96.20 ± 0.09	2.71 ± 0.07
DeCov+Dropout	98.78 ± 0.06	96.25 ± 0.07	2.53 ± 0.06
SDC	99.03 ± 0.10	96.97 ± 0.08	2.06 ± 0.09
SDC+Dropout	98.85 ± 0.07	97.20 ± 0.07	**1.65 ± 0.07**
Overlapped SDC	99.15 ± 0.11	97.13 ± 0.10	2.02 ± 0.10
Overlapped SDC+Dropout	98.99 ± 0.05	**97.31 ± 0.07**	1.68 ± 0.06

Table III
CLASSIFICATION RESULTS BY REGULARIZING THE POOLING LAYERS.

Method	train	test	train-test
NONE [27]	100.0 ± 0.00	75.24 ± 0.27	24.76 ± 0.27
Dropout on pool 2	87.16 ± 0.15	78.80 ± 0.22	**8.36 ± 0.18**
DeCov on pool 2	89.85 ± 0.32	78.76 ± 0.20	11.09 ± 0.25
SDC on pool 2	91.08 ± 0.22	**80.74 ± 0.19**	10.34 ± 0.20
Dropout on pool 3	86.29 ± 0.17	78.62 ± 0.14	**7.67 ± 0.15**
DeCov on pool 3	90.66 ± 0.54	78.45 ± 0.26	12.21 ± 0.33
SDC on pool 3	91.17 ± 0.16	**80.03 ± 0.12**	11.14 ± 0.14

4.79%, which are significant promotions for classification tasks. The proposed SDC beats DeCov by approximately 2% in accuracy, indicating that the enforced structures between the neurons are beneficial for regularizing the convolutional layers of the network.

V. CONCLUSION

In this paper, we propose a novel regularizer called structured decorrelation constraint (SDC) to reduce overfitting of the large network in an explicit way. The performance of SDC on the challenging datasets is much better than that of the conventional regularizers, indicating that: 1) exploring the correlations between neurons within the same layer can improve the network's performance; 2) SDC can encourage the neurons to form the grouped structures to learn structured and decorrelated representations; 3) the proposed Reg-Conv approach that enables the DeCov and SDC to regularize the convolutional layers is effective and efficient.

ACKNOWLEDGMENT

The authors would like to thank the handing editor and the anonymous reviewers for their careful reading and helpful remarks, which have contributed in improving the quality of this paper. This work was supported in part by the National Natural Science Foundation of China under Grants

61471274, 61302111, 61401317, 61231015, 91338111 and 61671336, National High Technology Research and Development Program of China (863 Program) under Grants 2015AA016306 and Australian Research Council Projects under Grant DP-140102164 and Grant FT-130101457.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[3] H. Wang, Y. Cai, Y. Zhang, H. Pan, W. Lv, and H. Han, "Deep learning for image retrieval: What works and what doesn't," in *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*. IEEE, 2015, pp. 1576–1583.

[4] L. Zhang, Q. Zhang, L. Zhang, D. Tao, X. Huang, and B. Du, "Ensemble manifold regularized sparse low-rank approximation for multiview feature embedding," *Pattern Recognition*, vol. 48, no. 10, pp. 3102–3112, 2015.

[5] X. Li, S. Qian, F. Peng, J. Yang, X. Hu, and R. Xia, "Deep convolutional neural network and multi-view stacking ensemble in ali mobile recommendation algorithm competition: The solution to the winning of ali mobile recommendation algorithm," in *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*. IEEE, 2015, pp. 1055–1062.

[6] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 1235–1244.

[7] J. Li, S. Gao, N. Han, Z. Fang, and J. Liao, "Music mood classification via deep belief network," in *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*. IEEE, 2015, pp. 1241–1245.

[8] L. Ge, J. Gao, X. Li, and A. Zhang, "Multi-source deep learning for information trustworthiness estimation," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2013, pp. 766–774.

[9] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, Mar 1994.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[11] D. M. Hawkins*, "The problem of overfitting," *Journal of Chemical Information and Computer Sciences*, vol. 44, no. 1, pp. 1–12, 2004.

[12] H. Wang, J. Yin, J. Pei, P. S. Yu, and J. X. Yu, "Suppressing model overfitting in mining concept-drifting data streams," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 736–741.

[13] Y. Yao, L. Rosasco, and A. Caponnetto, "On early stopping in gradient descent learning," *Constructive Approximation*, vol. 26, no. 2, pp. 289–315, 2007.

[14] J. Moody, S. Hanson, A. Krogh, and J. A. Hertz, "A simple weight decay can improve generalization," *Advances in Neural Information Processing Systems*, vol. 4, pp. 950–957, 1995.

[15] J. A. Hoeting, D. Madigan, A. E. Raftery, and C. T. Volinsky, "Bayesian model averaging: a tutorial," *Statistical science*, pp. 382–401, 1999.

[16] R. Caruana, A. Munson, and A. Niculescu-Mizil, "Getting the most out of ensemble selection," in *International Conference on Data Mining*. IEEE, 2006, pp. 828–833.

[17] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2003, pp. 226–235.

[18] R. Reed, S. Oh, R. J. Marks *et al.*, "Regularization using jittered training data," in *International Joint Conference on Neural Networks*, vol. 3, 1992, pp. 147–152.

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[20] O. Rippel, M. Gelbart, and R. Adams, "Learning ordered representations with nested dropout," in *Proceedings of the 31st International Conference on Machine Learning*, 2014, pp. 1746–1754.

[21] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proceedings of the 30th International Conference on Machine Learning*, 2013, pp. 1058–1066.

[22] I. Goodfellow, D. Warde-farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, vol. 28, no. 3, May 2013, pp. 1319–1327.

[23] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of The 32nd International Conference on Machine Learning*, 2015, pp. 448–456.

[24] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of Computer Statistics*, 2010, pp. 177–186.

[25] M. Li, T. Zhang, Y. Chen, and A. J. Smola, "Efficient mini-batch training for stochastic optimization," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 661–670.

[26] X. Sun, H. Kashima, T. Matsuzaki, and N. Ueda, "Averaged stochastic gradient descent with feedback: An accurate, robust, and fast training method," in *IEEE 10th International Conference on Data Mining*. IEEE, 2010, pp. 1067–1072.

[27] M. Cogswell, F. Ahmed, R. Girshick, L. Zitnick, and D. Batra, "Reducing overfitting in deep networks by decorrelating representations," *4th International Conference on Learning Representations*, 2016.

[28] G. Andrew, R. Arora, J. Bilmes, and K. Livescu, "Deep canonical correlation analysis," in *Proceedings of the 30th International Conference on Machine Learning*, 2013, pp. 1247–1255.

[29] S. Chandar, M. M. Khapra, H. Larochelle, and B. Ravindran, "Correlational neural networks," *Neural computation*, 2015.

[30] Y. Bengio and J. S. Bergstra, "Slow, decorrelated features for pretraining complex cell-like networks," in *Advances in neural information processing systems*, 2009, pp. 99–107.

[31] B. Cheung, J. A. Livezey, A. K. Bansal, and B. A. Olshausen, "Discovering hidden factors of variation in deep networks," *arXiv preprint arXiv:1412.6583*, 2014.

[32] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *The IEEE Conference on Computer Vision and Pattern Recognition*, June 2015.

[33] J. Feng and T. Darrell, "Learning the structure of deep convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2749–2757.

[34] H. Schulz and S. Behnke, "Structured prediction for object detection in deep neural networks," in *Artificial Neural Networks and Machine Learning*, 2014, pp. 395–402.

[35] M. Lin, Q. Chen, and S. Yan, "Network in network," *International Conference on Learning Representations*, 2014.

[36] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *The 14th European Conference on Computer Vision*, 2014, pp. 818–833.

[37] A. Krizhevsky, V. Nair, and G. Hinton, "The cifar-10 and cifar-100 datasets," 2009. [Online]. Available: http://www. cs. toronto. edu/˜ kriz/cifar. html

[38] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS workshop on deep learning and unsupervised feature learning*, vol. 2011, 2011, p. 4.

[39] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio, "Theano: new features and speed improvements," Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.