Reducing Overfitting in Deep Convolutional Neural Networks Using Redundancy Regularizer

Bingzhe Wu[1,2(✉)], Zhichao Liu[1], Zhihang Yuan[1,2], Guangyu Sun[1], and Charles Wu[2]

[1] CECA, Peking University, Beijing 100871, China
wubingzhe@pku.edu.cn
[2] Otureo Technologies (Beijing) Co., Ltd., Beijing 100080, China

Abstract. Recently, deep convolutional neural networks (CNNs) have achieved excellent performance in many modern applications. These high performance models normally accompany with deep architectures and a huge number of convolutional kernels. These deep architectures may cause overfitting, especially when applied to small training datasets. We observe a potential reason that there exists (linear) redundancy among these kernels. To mitigate this problem, we propose a novel regularizer to reduce kernel redundancy in a deep CNN model and prevent overfitting. We apply the proposed regularizer on various datasets and network architectures and compare to the traditional L2 regularizer. We also compare our method with some widely used methods for preventing overfitting, such as dropout and early stopping. Experimental results demonstrate that kernel redundancy is significantly removed and overfitting is substantially reduced with even better performance achieved.

1 Introduction

Recently, various deep CNN architectures have been widely proposed for modern applications, such as image classification and semantic segmentation. Most state-of-the-art CNN models tend to employ a lot of stacked layers along with a huge number of parameters, such as deep residual network [3], which has included more than 100 layers. Normally, it is easier to result in overfitting with more parameters included, especially on small training datasets. Thus, it has become one obstacle to apply these complicated models on many practical problems.

In the past few years, research works have been proposed to prevent overfitting by reducing Co-dependence in deep CNNs. Hinton et al. [4] introduced "dropout" to prevent overfitting. Srivastava et al. [10] show that dropout has a regularizing effect, leading to less correlated features. Cogswell et al. [1] find correlation between the cross-covariance of hidden unit activations. Then, they propose a loss function termed *DeCov* to prevent overfitting, which is based on the covariance matrix of the activation. Inspired by these works, we propose a novel regularizer to prevent overfitting in this work. Different from prior approaches, we concentrate on reducing the kernel redundancy instead of Co-dependence in deep networks. More specifically, we propose a new regularizer

A. Lintas et al. (Eds.): ICANN 2017, Part II, LNCS 10614, pp. 49–55, 2017.
https://doi.org/10.1007/978-3-319-68612-7_6

called $correlation_{loss}$, which encourages kernels that have less (linear) correlation. In addition, this regularizer is applied to convolutional layers rather than fully-connected layers.

We apply the proposed regularizer on different network architectures using various datasets, which include CIFAR10/100 and ImageNet. Experimental results show that kernel redundancy is significantly removed and overfitting is substantially reduced. Comparison with L2 penalty demonstrate the advantage of using our approach over traditional approaches. In addition, we even achieve a higher accuracy on CIFAR100 dataset than the previous state-of-the-art result (81.03 % vs. 75.72 %).

The rest of this paper is organized as follows. In Sect. 2, we present how to calculate correlation coefficients between two kernels. Based on this, we visualize features and convolutional kernels to show kernel redundancy. In Sect. 3, we propose our novel regularizer termed $correlation_{loss}$. In Sect. 4, we provide comprehensive experimental results over a range of datasets, followed by a conclusion in Sect. 5.

2 Exploring Kernel Redundancy in Deep CNNs

In this section, we first quantitatively define redundancy between two convolution kernels using correlation coefficient. Then, we demonstrate kernel redundancy in real CNN models.

2.1 Correlation Between Two Kernels

First, we explain how to compute the correlation coefficient between two vectors denoted as x, y. Let $f_{cor}(x, y)$ denote the Pearson's correlation coefficient [7] of two vectors,

$$f_{cor}(x,y) = \frac{\sum_i (x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_i (x_i - \overline{x})^2 \sum_i (y_i - \overline{y})^2}}, \tag{1}$$

where $\overline{x}$ and $\overline{y}$ denote the average of elements in vector x and y, respectively.

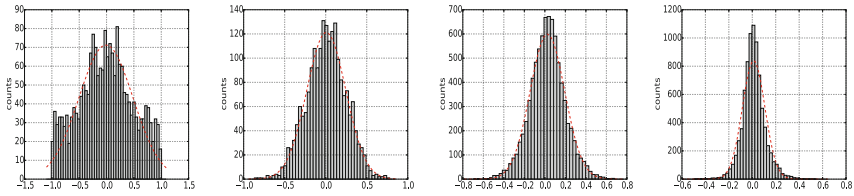
Here we denote two given convolutional kernels as $S1$ and $S2$. We first flatten $S1$, $S2$ to $S1_{flatten}$, $S2_{flatten}$. The correlation is computed between two flattened vectors. We use the computed correlations as a metric of similarity between two kernels. For simplicity, we denote $cor(S1, S2)$ as the similarity metric. The computing formula is listed as follows:

$$cor(S1, S2) = f_{cor}(S1_{flatten}, S2_{flatten}) \tag{2}$$

2.2 Distribution of Correlations in a Real CNN Model

We apply the proposed $cor(Si, Sj)$ on a VGG19 [9] model trained on ImageNet. We calculate the correlation coefficients among all pairs of kernels i and j from

the same layer in VGG19. The histogram of correlation coefficients for all kernel pairs is shown in Fig. 1. For the first convolution layer, we can observe that the distribution of $cor(Si, Sj)$ is zero centered. But a number of kernel pairs gather at both ends in the histogram. This indicates that there are many kernel pairs, which have high correlation coefficients in the first convolution layer of the VGG19 model. While going deeper into the network, we find that the numbers of high correlation kernel pairs decrease sharply. An interesting observation is that this trend is similar to that found in Shang's work [8]. Shang et al. propose a new activation to eliminate this phenomenon. In this work, we will address this issue by minimizing the $correlation_{loss}$ (will be introduced in Sect. 3).



(a) conv1_1 (b) conv1_2 (c) conv2_1 (d) conv2_2

Fig. 1. Histograms of the correlation coefficients from conv1_1 to conv2_2 in VGG19. There are many kernel pairs with high correlation coefficient in conv1_1. Numbers of high-correlation pairs gradually decrease when going deeper into the network.

2.3 Visualizing High-Similarity Kernels and Features

Based on the above observation, we select some high-correlation kernel pairs for visualization in Fig. 2. We put kernel pairs with high-correlation together and we find that they have a similar pixel distributions. For comparison, we select some high-similarity kernel pairs from VGG19 [9] and AlexNet [6] in Figs. 3 and 4, respectively.

Fig. 2. Illustration of kernel pairs with high-correlation.

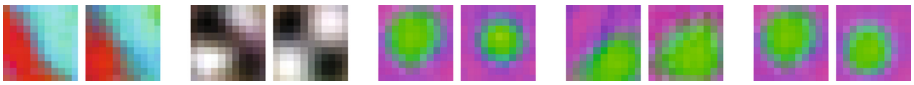


Fig. 3. Some high-correlation pairs of kernels from the same layer of AlexNet.

In Fig. 4, column 2 and column 3 demonstrate the features that extracted by the kernels from high-similarity pairs. It is difficult for human to distinguish two generated features from the same image.

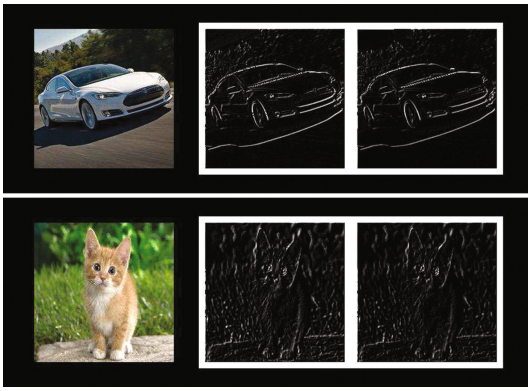


Fig. 4. Demonstrate the features extracted by high-similarity pair of kernels.

Inspired by these observations, we can treat these pairs with high correlation as a (linear) redundancy in deep CNNs. To address this issue, we minimize the $correlation_{loss}$ to eliminate the (linear) redundancy of convolution layer in deep CNNs. The $correlation_{loss}$ will be introduced in next section.

3 $Correlation_{loss}$

We collect correlation coefficients among all pairs of kernels from same convolution layer and form a matrix G listed as follows. Si and Sj represent the i-th kernel and j-th kernel of a convolution layer in a deep CNN.

$$G_{i,j} = cor(Si, Sj) \tag{3}$$

As discussed in last section, we try to minimize the correlations among these kernels. Thus, we can treat the Frobenius norm of G as a regularizer. Since the diagonal of G is the self-correlation coefficients, we can subtract the term from the matrix norm to calculate a final penalty term as follows:

$$correlation_{loss} = \frac{1}{2}(\|G\|_F^2 - \|diag(G)\|_2^2) \tag{4}$$

where $||\cdot||_F$ is the Frobenius norm.

We can find that $correlation_{loss}$ is different from L_{DeCov} introduced in [1]. Instead, it is similar to the L_2 or L_1 regularizers because it is a function purely based on weight vectors. We can simply implement the regularizer and apply it to any layer in a deep CNN. We add this loss with classification loss and get total loss with following equation. l represents the l-th layer in a deep CNN model.

$$total_{loss} = L_{classification} + \lambda * \sum_{l} correlation_{loss} \tag{5}$$

4 Experiment Results

4.1 CIFAR10 and CIFAR100

CIFAR10 and CIFAR100 datasets each consist of 50,000 training and 10,000 testing images evenly drawn from 10 and 100 classes respectively. For preprocessing, we subtracted the mean and divide by the variance. We also use random horizontal flip and change the contrast for data augmentation.

We conduct experiments with VGG16 [9] and 34 layers Res-Net [3] on these two datasets. We add batch normalization [5] after each convolutional layers. For $correlation_{loss}$, we set $\lambda = 0.01$ in Eq. (5).

We refer cor_{loss} to $correlation_{loss}$ in the result table for simplicity. In this paper, we use the gap between train and test accuracy for evaluating overfitting. We conduct a serial of experiments to compare different strategies for preventing overfitting. Firstly, we compare our proposed regularizer with the traditional L2 regularizer. In Tables 1 and 2, we observe a significant improvement when using $correlation_{loss}$ on these two models. We see that using $correlation_{loss}$ instead of $L2$ weight decay has obviously decreased the gap (between train accuracy and test accuracy) while obtaining a better test accuracy.

Table 1. Results on CIFAR10

Model	cor_{loss}	L2	Train	Test	Train-test
VGG16	No	Yes	98.78	85.45	13.33
VGG16	Yes	No	95.24	90.28	4.96
ResNet-34	No	Yes	100	91.13	8.87
ResNet-34	Yes	No	99.45	93.45	6.00

To compare our method with previous methods for preventing overfitting, We conduct experiments with 34 layers ResNet [5] on cifar10 dataset. For fair comparison, we add L2 regularizer in all these experiments. Table 3 shows the results.

From the experimental results in Table 3, our method has a slight improvement compared to dropout. We find that using both our method and dropout can further improve the accuracy.

We also combine early stopping with our method and dropout to see if it can provide improvement. In our experiment, 10% of the original training set is split into a validation set. We find that using early stopping doesn't improve the generation of such a complicated deep convolutional neural network. In [10], the authors also mentioned this issue.

Table 2. Results on CIFAR100

Model	cor_{loss}	L2	Train	Test	Train-test
VGG16	No	Yes	99.45	75.50	8.01
VGG16	Yes	No	85.34	81.03	4.31
ResNet-34	No	Yes	100	66.6	33.4
ResNet-34	Yes	No	95.43	72.24	23.19

Table 3. Comparative Experiments with dropout based on Resnet-34.

Dataset	cor_{loss}	Dropout	Early stopping	Train	Test	Train-test
Cifar10	No	Yes	No	98.97	93.02	5.95
Cifar10	Yes	No	No	98.22	93.20	5.02
Cifar10	Yes	Yes	No	97.32	94.18	3.14
Cifar10	Yes	Yes	Yes	97.12	93.41	3.71

4.2 ImageNet

Imagenet [2] is a large labeled dataset. In our experiment, we select the validation set of ImageNet2012 as testing dataset and the ILSVRC12's training data as the training dataset. We test our new regularizer on VGG19 [9] and find once again that the cor_{loss} gives improvement over the baseline model with L2 regularizer.

Table 4. ImageNet Benchmark set results

Models	Top1	Top5	Train-val (Top1)
VGG19 with L2	68.44	88.37	15.27
VGG19 with cor_{loss}	69.52	88.47	11.84

Even more interesting thing is that we observe the phenomenon discussed in Sect. 3 vanishes when we use the $correlation_{loss}$ instead of the $L2$ regularizer. It proves that this new regularizer is effective for reducing the (linear) kernel redundancy.

5 Conclusion

In this paper, we propose a new regularizer called $correlation_{loss}$, which explicitly penalizes correlations among kernels in convolutional layers. Our new regularizer has demonstrated a strong ability to prevent overfitting. We show that using $correloation_{loss}$ achieves better performance than traditional regularizer with different datasets and model architectures.

Acknowledgments. This work is supported by National Natural Science Foundation of China (No. 61572045).

References

1. Cogswell, M., Ahmed, F., Girshick, R., Zitnick, L., Batra, D.: Reducing overfitting in deep networks by decorrelating representations. ICLR (2016)
2. Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L.: Imagenet: a large-scale hierarchical image database. In: IEEE Conference on Computer Vision and Pattern Recognition, 2009, CVPR 2009, pp. 248–255. IEEE (2009)
3. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CoRR, abs/1512.03385 (2015)
4. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580 (2012)
5. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
6. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
7. Pearson, K.: Note on regression and inheritance in the case of two parents. Proc. R. S. London **58**, 240–242 (1895)
8. Shang, W., Sohn, K., Almeida, D., Lee, H.: Understanding and improving convolutional neural networks via concatenated rectified linear units. arXiv preprint arXiv:1603.05201 (2016)
9. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. Eprint Arxiv (2014)
10. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)