REGULARIZATION OF CONVOLUTIONAL NEURAL NETWORKS USING SHUFFLENODE

Yihao Chen[1,2], *Hanli Wang*[1,2,*] , *Yu Long*[1,2]

[1] Department of Computer Science and Technology, Tongji University, Shanghai, P. R. China
[2]Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University, Shanghai, P. R. China

ABSTRACT

Convolutional Neural Network (CNN) has recently achieved significant performances for visual computing, and a number of researches are made to explore advanced model structures to solve the problem of over-fitting. In this paper, a regularization technique named ShuffleNode is proposed, which shuffles feature map elements to achieve regularization functions during model training. Specifically, there are two shuffle ways including within-map shuffle and cross-map shuffle, which are suitable to be employed in convolutional layers. The method of within-map shuffle is used to provide the exchange of elements within one feature map, while the cross-map shuffle method offers the opportunity of information sharing across different feature maps. The experimental results on several benchmark image classification datasets demonstrate the efficiency of the proposed method.

Index Terms— Convolutional Neural Network, regularization, shuffle, Dropout.

1. INTRODUCTION

The recent years witness a great success of Convolutional Neural Network (CNN) for visual computing, such as object detection [1], image retrieval [2], image segmentation [3], action recognition [4], and so on. In order to improve CNN performances, a number of advanced techniques are designed, including nonlinear activation functions [5], pooling approaches [6], regularization methods [7–9], network architectures [10–12], to name a few.

Due to the high model capacity, CNN models usually encounter the general problem of over-fitting. To address this issue, a number of regularization techniques have been designed in the literature. In the early days, it is common to reduce model size and employ early stopping criteria [13] to avoid over-fitting. In [14], weight decay is developed by adding ℓ_2 regularization on weight parameters to enhance regularization functionalities. Data augmentation [15] becomes a popular solution to CNN network regularization through expanding training dataset.

As a promising regularization method, Dropout [7] stochastically sets a number of activations in a hidden layer to zeros for each training sample. Since hidden units can not well collaborate to each other by Dropout, they need to learn a better representation for inputs to achieve generalization. As the generalization of Dropout, DropConnect [8] sets a randomly selected subset of weights to zeros. In [9], DisturbLabel is proposed to regularize CNN in the loss layer by setting several randomly selected labels to be incorrect for each training batch. In [16], stochastic pooling is designed to enable the training of larger models for weakening over-fitting. The key idea of stochastic pooling is to make the pooling process in each convolutional layer a stochastic process based on multinomial distribution. Moreover, mixed pooling is proposed in [17] to regularize CNNs, which replaces deterministic pooling with a stochastic procedure by randomly using the conventional max pooling and average pooling methods.

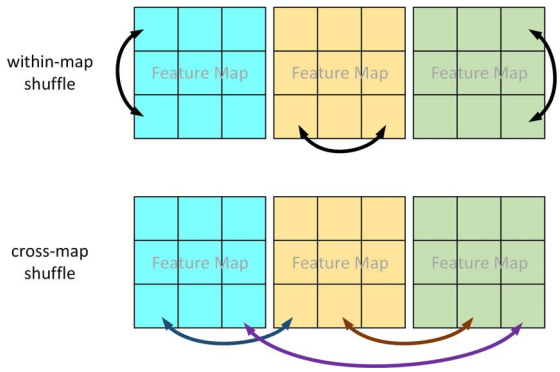

Fig. 1. Illustration of within-map shuffle and cross-map shuffle of the proposed ShuffleNode regularization technique.

Among the aforementioned regularization approaches, DisturbLabel, data augmentation and weight decay are not suitable for the convolutional layer, while Dropout and DropConnect are designed for full connection layers. Stochastic pooling and mixed pooling can only be used for pooling operations. Different from these regularization methods, we propose a novel regularization technique named ShuffleNode

*Corresponding author (H. Wang, E-mail: hanliwang@tongji.edu.cn).

This work was supported in part by the National Natural Science Foundation of China under Grants 61622115 and 61472281, and the Program for Professor of Special Appointment (Eastern Scholar) at the Shanghai Institutions of Higher Learning under Grant GZ2015005.

which is designed for the convolutional layer. ShuffleNode regularizes CNN by shuffling feature map elements during model training, and two shuffle ways can be chosen as illustrated in Fig. 1, including 1) within-map shuffle which shuffles feature map elements between different locations in a single feature map and 2) cross-map shuffle which shuffles feature map elements at the same location across different feature maps in one convolutional layer. Both of the within-map shuffle and cross-map shuffle methods randomly shuffle a selected subset of feature map elements to regularize CNN.

The rest of this paper is organized as follows. The proposed ShuffleNode regularization technique is introduced in Section 2. The experimental results and analyses are presented in Section 3. Finally, Section 4 concludes this work.

2. PROPOSED SHUFFLENODE REGULARIZATION

2.1. Motivation

It is well known that the information flow [18] of a CNN network is greatly influenced by its network structure. In CNN, the intermediate layers perform deterministic transformations to enhance the performance and it is desired that all information can be preserved after such operations. However, an amount of information may be lost due to these deterministic functions [18]. As a solution, it is beneficial for the intermediate layers to offer a variety of kinds of information exchange while preserving the deterministic functions, *e.g.*, cross-layer information flow is employed by the recent ResNet [12] to allow information exchange across layers.

Inspired by the above analysis, ShuffleNode provides information exchange in the convolutional layer of CNN models. Specifically speaking, the within-map shuffle method offers the opportunity of distant information exchange of feature map elements within one feature map, while the cross-map shuffle method bestows a convolutional layer with the advantage of information sharing across different feature maps.

Algorithm 1 Shuffle-Index

Require: n: the length of the elements to be shuffled
Ensure: S: the shuffled index vector
1: **for** i = 1 ; i $\leq$ n ; i = i + 1 **do**
2: S_i=i;
3: **end for**
4: **for** i = n ; i $\geq$ 1 ; i = i$-$1 **do**
5: $temp$ = random(i); // random(i) $\in$ [1,n]
6: swap(S_i, S_{temp});
7: **end for**
8: **return** S

2.2. Shuffle Operations

As aforementioned, ShuffleNode shuffles feature map elements to achieve regularization functionalities during CNN training. There are two shuffle ways including within-map shuffle and cross-map shuffle, which obey a similar shuffle process but apply to different subsets of feature map elements.

First, we start our introduction to ShuffleNode with the shuffle operation of 'Shuffle-Index'. Suppose there are n elements to be shuffled with their element indices denoted by $1, 2, 3, \cdots, n$, the Shuffle-Index operation shuffles these elements and then outputs the shuffled index vector of S. The pseudo code of Shuffle-Index is provided in Algorithm 1.

2.2.1. Within-Map Shuffle

The proposed within-map shuffle method exchanges the feature map elements between different locations within one feature map for each training batch independently. Assume there is a specific convolutional layer which contains C feature maps with the size of $W \times H$, let x_i^c, y_i^c represent the i^{th} input and output in the c^{th} feature map before and after the within-map shuffle operation, respectively. We then serialize the feature map elements to be a linear vector and the forward pass of within-map shuffle operation can be described as

$$S = \text{Shuffle-Index}(W \times H), \tag{1}$$

$$\begin{cases} y_i^c = x_i^c, & p_c = 0 \\ y_i^c = x_{S_i}^c, & p_c = 1 \end{cases}, \tag{2}$$

where p_c is generated by the following Bernoulli distribution as

$$P \sim Bernoulli(C, r), \tag{3}$$

where $r \in [0, 1]$ stands for the probability of each of the independent Bernoulli random variables being 1, which can also be considered as the shuffle ratio in the proposed ShuffleNode context. According to Eq. (2), the output of y_i^c will be updated as the shuffled element of $x_{S_i}^c$ if the corresponding Bernoulli sampled result $p_c = 1$; otherwise, the feature map elements in the c^{th} feature map keep unchanged when $p_c = 0$.

2.2.2. Cross-Map Shuffle

The proposed cross-map shuffle method shuffles the feature map elements at the same location of different feature maps in a convolutional layer. Let $x_{i,j,k}$ and $y_{i,j,k}$ represent the input and the output at the i^{th} row and the j^{th} column in the k^{th} feature map, where $0 \leq i < W, 0 \leq j < H, 0 \leq k < C$, the forward pass of cross-map shuffle operation can be formulated as

$$S = \text{Shuffle-Index}(C), \tag{4}$$

$$\begin{cases} y_{i,j,k} = x_{i,j,k}, & p_{i \times W + j} = 0 \\ y_{i,j,k} = x_{i,j,S_k}, & p_{i \times W + j} = 1 \end{cases}, \tag{5}$$

where $p_{i \times W + j}$ is generated from

$$P \sim Bernoulli(W \times H, r). \tag{6}$$

As seen from Eq. (5), for each output feature map element (*i.e.*, $y_{i,j,k}$), it will be updated with the shuffled element (*i.e.*, x_{i,j,S_k}) if the related Bernoulli sampled result is equal to 1 (*i.e.*, $p_{i\times W+j}=1$); otherwise it maintains unchanged.

3. EXPERIMENTS

The proposed ShuffleNode regularization technique is evaluated on four benchmark image classification datasets, including CIFAR-10 [19], CIFAR-100 [19], MNIST [20] and STL-10 [21]. Regarding the underlying CNN model architecture to deploy ShuffleNode, the improved LeNet model [9], VGG model [10] and ALL-CNN model [22] are employed as the representative shallow and deep CNN networks, respectively. All the experiments are conducted on the widely utilized open source deep learning framework of Caffe [23], mirroring and cropping are adopted for data augmentation.

3.1. Effect on Information Flow

First, the proposed ShuffleNode is evaluated from the perspective of information flow. In order to demonstrate the effectiveness of information exchange achieved by ShuffleNode, we develop the method of DropNode, a modified version of Dropout, which drops the feature map elements that will be shuffled by ShuffleNode. Due to the space limit, only the results on the dataset of CIFAR-10 [19] are presented on the improved LeNet model [9] and the ALL-CNN model [22].

The baseline configuration of the improved LeNet model [9] in this experiment is conv(32,5) – MP – conv(32,5) – MP – conv(64,5) – MP – FC(64) – DP – FC(10), where conv(n,s) denotes a convolutional layer consisting of n feature maps with the convolutional kernel size of $s\times s$, MP and DP stand for max pooling and Dropout, respectively, and FC(m) indicates a full connection layer with m nodes. As far as the baseline configuration of the ALL-CNN model [22] is concerned, it is set as conv(96,3) – conv(96,3) – conv(96,3) – DP – conv(192,3) – conv (192,3) – conv(192,3) – DP – conv(192,3) – conv(192,1) – conv(10,1) – GAP, where GAP is the abbreviation of global average pooling.

When applying the DropNode and ShuffleNode into the improved LeNet model, the modified network architecture is updated as conv(32,5) – MP – conv(32,5) – MP – conv(64,5) – $\mathcal{Q}(r)$ – MP – FC(64) – DP – FC(10), where $\mathcal{Q}(r)$ stands for the DropNode or ShuffleNode operation with the activation ratio equal to r. In this experiment, r is empirically set to 0.3 for both the DropNode and ShuffleNode operations. On the other hand, regarding the modified network architecture of the ALL-CNN model, it becomes conv(96,3) – conv(96,3) – conv(96,3) –$\mathcal{Q}(0.2)$ – DP – conv(192,3) – $\mathcal{Q}(0.2)$ – conv (192,3) – $\mathcal{Q}(0.2)$ – conv(192,3) – $\mathcal{Q}(0.2)$ – DP – conv(192,3) – conv(192,1) – conv(10,1) – GAP.

The comparison results with the classification error rate on CIFAR-10 are presented in Table 1, where it can be seen that both the DropNode and ShuffleNode methods can further improve the classification performance against the baseline model, and the proposed ShuffleNode technique is superior to DropNode for information exchange.

Table 1. Comparison of classification error rate (%) obtained by DropNode and ShuffleNode on CIFAR-10.

Method	**LeNet**	**ALL-CNN**
Baseline	22.20	11.65
DropNode	19.36	9.27
ShuffleNode	**18.24**	**8.70**

3.2. Effect on Inter-Class Separation

As suggested in [24], we compare the proposed ShuffleNode with Dropout from the viewpoint of inter-class separation, which is a measurement to evaluate the goodness of classification. To achieve this comparison, the improved LeNet model [9] is employed and the dataset of MNIST [20] is used. The baseline LeNet model is configured as conv(32,5) – MP – conv(64,5) – MP – FC(512) – FC(10), and the corresponding architectures of its competitors with Dropout and ShuffleNode are set as conv(32,5) – MP – conv(64,5) – MP – FC(512) – DP – FC(10) and conv(32,5) – MP – conv(64,5) – $\mathcal{Q}(0.3)$ – MP – FC(512) – FC(10), respectively.

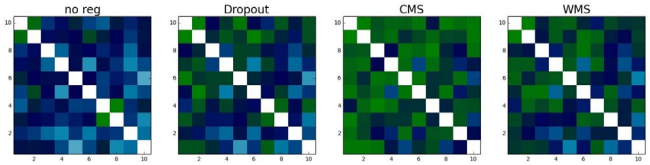

Fig. 2. Comparison of inter-class separation achieved by the proposed ShuffleNode regularization technique and Dropout on MNIST. Note 'no reg' indicates the effect obtained by the baseline LeNet model with Dropout or ShuffleNode disabled, 'WMS' and 'CMS' denote the proposed within-map shuffle and cross-map shuffle, respectively. Darker blue color indicates larger correlation between classes, while brighter green indicates smaller correlation.

The inter-class separation can be expressed as the correlation coefficient between classes following the computing protocol in [24]. The lower the correlation coefficient between classes is, the larger the inter-class separation becomes and thus the better the classification effect is. Figure 2 illustrates the comparison of inter-class separation achieved by the proposed ShuffleNode regularization technique and Dropout on the MNIST dataset. In Fig. 2, 'no reg' indicates the result by the baseline LeNet model, the results achieved by within-map shuffle and cross-map shuffle are marked as 'WMS' and 'CMS', respectively. As observed from this comparison, it can be

clearly seen that all of the regularization methods including Dropout, within-map shuffle and cross-map shuffle are able to achieve better inter-class separation effects than that of the baseline.

3.3. CIFAR-10

The CIFAR-10 [19] dataset is composed of 60,000 natural 32×32 color images that equally fall into ten classes. Each class has 5,000 samples for training and another 1,000 samples for testing. We implement the proposed ShuffleNode regularization technique with the improved LeNet model [9], VGG-19 model [10] and ALL-CNN model [22]. A number of network configurations are performed to evaluate the proposed ShuffleNode regularization technique.

Regarding the improved LeNet model [9], the following configurations are compared, including 1) baseline: conv(32,5) – MP – conv(32,5) – MP – conv(64,5) – MP – FC(64) – FC(10), 2) Dropout: conv(32,5) – MP – conv(32,5) – MP – conv(64,5) – MP – FC(64) – DP – FC(10), 3) ShuffleNode and 4) ShuffleNode + Dropout. The comparison of the classification error rate on CIFAR-10 achieved by these competitive methods is presented in Table 2, where both the conditions of data augmentation (denoted as 'DA') enabling and disabling (denoted as 'w/o DA') are tested. From the results, it can be seen that both the proposed within-map shuffle (WMS) and cross-map shuffle (CMS) obtain better classification performances than the baseline and Dropout. Moreover, the proposed ShuffleNode can be combined with Dropout to further improve the classification accuracy. Among these results, the best performance is achieved by CMS+Dropout with the network configuration as conv(32,5) – MP – conv(32,5) – MP – conv(64,5) – CMS(0.3) – MP – FC(64) – DP – FC(10).

Table 2. Comparison of classification error rate (%) with the improved LeNet model on CIFAR-10.

Method	w/o DA	DA
Baseline	22.20	15.90
Dropout	20.40	14.99
CMS	18.24	14.51
WMS	19.76	14.78
CMS + Dropout	**16.89**	**14.31**
WMS + Dropout	18.99	14.90

Regarding the VGG-19 model [10], the network configuration of CMS or WMS is conv(64,3) – conv(64,3) – MP – conv(128,3) – conv(128,3) – MP – conv (256,3) – $\mathcal{Q}(0.3)$ – conv (256,3) – conv (256,3) – conv (256,3) – MP – conv(512,3) – $\mathcal{Q}(0.3)$ – conv(512,3) – conv(512,3) – conv(512,3) – MP – conv(512,3) – $\mathcal{Q}(0.3)$ – conv(512,3) – conv(512,3) – conv(512,3) – MP – FC(4096) – FC(4096) – FC(10), where $\mathcal{Q}(0.3)$ stands for the corresponding CMS(0.3) or WMS(0.3). Table 4 shows the comparative results, where CMS + Dropout performs the best.

Table 3. Comparison of classification error rate (%) with the VGG-19 model on CIFAR-10.

Method	Error Rate (DA)
Baseline	8.93
Dropout	7.70
CMS	7.48
WMS	7.63
CMS + Dropout	**6.89**
WMS + Dropout	7.40

As far as the ALL-CNN model [22] is concerned, the baseline configuration is conv(96,3) – conv(96,3) – conv(96,3) – conv(192,3) – conv (192,3) – conv(192,3) – conv(192,3) – conv(192,1) – conv(10,1) – GAP. When Dropout is employed, the network configuration becomes conv(96,3) – conv(96,3) – conv(96,3) – DP – conv(192,3) – conv (192,3) – conv(192,3) – DP – conv(192,3) – conv(192,1) – conv(10,1) – GAP. The comparison of the classification error rate on CIFAR-10 achieved by ALL-CNN based models is presented in Table 4, where it can be seen that the best performances are achieved by CMS + Dropout and WMS + Dropout when data augmentation is enabled or disabled, respectively, with the network configuration of conv(96,3) – conv(96,3) – conv(96,3) –$\mathcal{Q}(0.2)$ – DP – conv(192,3) – $\mathcal{Q}(0.2)$ – conv(192,3) – $\mathcal{Q}(0.2)$ – conv(192,3) – $\mathcal{Q}(0.2)$ – DP – conv(192,3) – conv(192,1) – conv(10,1) – GAP, where $\mathcal{Q}(0.2)$ stands for the corresponding CMS(0.2) or WMS(0.2).

Table 4. Comparison of classification error rate (%) with the ALL-CNN model on CIFAR-10.

Method	w/o DA	DA
Baseline	11.65	7.95
Dropout	10.02	6.75
WMS + CMS	8.68	6.66
CMS	8.70	6.73
CMS + Dropout	**8.50**	6.69
WMS + Dropout	8.76	**6.60**

3.4. CIFAR-100

Similar to CIFAR-10, the collection of CIFAR-100 [19] has the same picture size and total number of images, except that there are a hundred classes and each class owns 500 images for training and 100 for testing. The network structures used herein are the same as that of the corresponding improved LeNet and ALL-CNN models for CIFAR-10. The

comparison results of classification error rate with the improved LeNet model and the ALL-CNN model are respectively shown in Table 5 and Table 6.

Table 5. Comparison of classification error rate (%) with the improved LeNet model on CIFAR-100.

Method	w/o DA	DA
Baseline	55.95	46.78
Dropout	51.83	44.24
CMS	48.76	42.38
WMS	51.01	44.02
CMS + Dropout	47.89	41.93
WMS + Dropout	**47.32**	**40.50**

Table 6. Comparison of classification error rate (%) with the ALL-CNN model on CIFAR-100.

Method	w/o DA	DA
Baseline	44.76	30.98
Dropout	33.52	29.93
WMS + CMS	32.61	29.30
CMS	31.45	30.21
CMS + Dropout	32.20	29.34
WMS + Dropout	**30.95**	**29.16**

As seen from the results, the proposed within-map shuffle and cross-map shuffle methods outperform Dropout for classification performance improvement. The fusion of the proposed within-map shuffle and Dropout is able to achieve the best performance with the network configurations on LeNet and ALL-CNN as conv(32,5) – MP – conv(32,5) – MP – conv(64,5) – WMS(0.1) – MP – FC(64) – DP – FC(10), conv(96,3) – conv(96,3) – conv(96,3) – WMS(0.2) – DP – conv(192,3) – WMS(0.2) – conv (192,3) – WMS(0.2) – conv(192,3) – WMS(0.2) – DP – conv(192,3) – conv(192,1) – conv(10,1) – GAP, respectively.

3.5. MNIST

The MNIST [20] dataset is consisted of a training set of 60,000 samples and a smaller testing set of 10,000 samples. Each sample is a labeled 28×28 gray image that appears as one handwritten digit from 0–9. The classification task on this dataset is not as hard as it is on those datasets consisted of real-world objects. As a result, it is only necessary to employ much smaller quantity of weights, leading to the improved LeNet model [9] utilized in this experiment. For comparison, the baseline architecture of LeNet is configured as conv(32,5) – MP – conv(64,5) – MP – FC(512) – FC(10), while the Dropout-based model configuration is conv(32,5) – MP – conv(64,5) – MP – FC(512) – DP – FC(10). The experimental results are presented comparatively in Table 7, from which it can be concluded that both the within-map shuffle and cross-map shuffle achieve better classification performances than that of Dropout and the best result is obtained by CMS + Dropout with the network configuration as conv(32,5) – MP – conv(64,5) – CMS(0.3) – MP – FC(512) – DP – FC(10).

Table 7. Comparison of classification error rate (%) obtained by LeNet on MNIST.

Method	Error Rate
Baseline	0.92
Dropout	0.75
CMS	0.65
WMS	0.74
CMS + Dropout	**0.59**
WMS + Dropout	0.66

3.6. STL-10

STL-10 [21] is also an image dataset for object recognition. It is composed of ten classes of objects and the image resolution is 96×96. For each class, apart from the unlabeled images for unsupervised learning, there are 500 training images and 800 test images. In our experiments, we use the same baseline structure of the ALL-CNN model for CIFAR-10 and train models with the labeled training samples. Table 8 shows the comparative results, where it is observed again that the best performance is achieved by CMS + Dropout with the network configuration as conv(96,3) – conv(96,3) – conv(96,3) – CMS(0.2) – DP – conv(192,3) – CMS(0.2) – conv(192,3) – CMS(0.2) – conv(192,3) – CMS(0.2) – DP – conv(192,3) – conv(192,1) – conv(10,1) – GAP.

Table 8. Comparison of classification error rate (%) obtained by ALL-CNN on STL-10.

Method	Error Rate (DA)
Baseline	21.59
Dropout	19.41
WMS + CMS	18.85
CMS	18.25
CMS + Dropout	**17.88**
WMS + Dropout	18.09

4. CONCLUSION

In this work, a novel technique ShuffleNode is proposed for regularization on convolutional layers. By ShuffleNode, two shuffle operations including within-map shuffle and

cross-map shuffle are performed to exchange information among feature maps in convolutional layers. The extensive experiments have demonstrated the superiority of the proposed ShuffleNode over the benchmark Dropout regularization method. Moreover, the fusion of ShuffleNode and Dropout can further boost the CNN performance.

5. REFERENCES

[1] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. CVPR'14*, Jun. 2014, pp. 580–587.

[2] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: an astounding baseline for recognition," in *Proc. CVPR'14*, Jun. 2014, pp. 806–813.

[3] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected CRFs," *arXiv preprint arXiv:1412.7062*, 2014.

[4] H. Qiao, X. Xi, Y. Li, W. Wu, and F. Li, "Biologically inspired visual model with preliminary cognition and active attention adjustment," *IEEE Trans. Cybernetics*, vol. 45, no. 11, pp. 2612–2624, Dec. 2015.

[5] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. Workshop ICML'13*, Jun. 2013.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *Proc. ECCV'14*, Sept. 2014, pp. 346–361.

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

[8] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proc. ICML'13*, Jun. 2013, pp. 2095–2103.

[9] L. Xie, J. Wang, Z. Wei, M. Wang, and Q. Tian, "Disturblabel: Regularizing CNN on the loss layer," in *Proc. CVPR'16*, Jun. 2016, pp. 4753–4762.

[10] K. Simonyan and A. Zisserman., "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR'15*, May 2015.

[11] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. CVPR'15*, Jul. 2015, pp. 1–9.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR'16*, Jun. 2016, pp. 770–778.

[13] D. C. Plaut, S. J. Nowlan, and G. E. Hinton, "Experiments on learning by back propagation," CMU, Tech. Rep. CMU-CS-86-126, Jun. 1986.

[14] A. S. Weigendas, D. E. Rumelhart, and B. A. Huberman, "Generalization by weight-elimination with application to forecasting," in *Proc. NIPS'90*, Oct. 1990, pp. 875–882.

[15] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. NIPS'12*, Dec. 2012, pp. 1097–1105.

[16] M. D. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," *arXiv preprint arXiv:1301.3557*, 2013.

[17] D. Yu, H. Wang, P. Chen, and Z. Wei, "Mixed pooling for convolutional neural networks," in *Proc. RSKT'14*, Oct. 2014, pp. 364–375.

[18] G. Alain and Y. Bengio, "Understanding intermediate layers using linear classifier probes," *arXiv preprint arXiv:1610.01644*, 2016.

[19] A. Krizhevsky, "Learning multiple layers of features from tiny images," Master's thesis, Department of Computer Science, University of Toronto, 2009.

[20] Y. L. Cun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[21] A. Coates, A. Y. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proc. AISTATS'11*, Apr. 2011, pp. 215–223.

[22] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," in *Proc. ICLR'15*, May 2015.

[23] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. ACM MM'14*, Nov. 2014, pp. 675–678.

[24] S. Shankar, D. Robertson, Y. Ioannou, A. Criminisi, and R. Cipolla, "Refining architectures of deep convolutional neural networks," in *Proc. CVPR'16*, Jun. 2016, pp. 2212–2220.