Weighted Channel Dropout for Regularization of Deep Convolutional Neural Network

Saihui Hou, Zilei Wang
Department of Automation, University of Science and Technology of China
saihui@mail.ustc.edu.cn, zlwang@ustc.edu.cn

Abstract

In this work, we propose a novel method named Weighted Channel Dropout (WCD) for the regularization of deep Convolutional Neural Network (CNN). Different from Dropout which randomly selects the neurons to set to zero in the fully-connected layers, WCD operates on the channels in the stack of convolutional layers. Specifically, WCD consists of two steps, *i.e.*, Rating Channels and Selecting Channels, and three modules, *i.e.*, Global Average Pooling, Weighted Random Selection and Random Number Generator. It filters the channels according to their activation status and can be plugged into any two consecutive layers, which unifies the original Dropout and Channel-Wise Dropout. WCD is totally parameter-free and deployed only in training phase with very slight computation cost. The network in test phase remains unchanged and thus the inference cost is not added at all. Besides, when combining with the existing networks, it requires no re-pretraining on ImageNet and thus is well-suited for the application on small datasets. Finally, WCD with VGGNet-16, ResNet-101, Inception-V3 are experimentally evaluated on multiple datasets. The extensive results demonstrate that WCD can bring consistent improvements over the baselines.

Introduction

Recent years have witnessed the great bloom of deep Convolutional Neural Network (CNN), which has significantly boosted the performance for a variety of visual tasks (He et al. 2016; Liu et al. 2016; Wang et al. 2016). The success of deep CNN is largely due to its structure of multiple nonlinear hidden layers, which contain millions of parameters and thus are able to learn the complicated relationship between input and output. However, when only limited training data is available, *e.g.*, in the field of Fine-grained Visual Categorization (FGVC), overfitting is very likely to occur, which would incur the performance drop.

In the previous literatures, many methods have been proposed to reduce the overfitting when training CNN, such as data augmentation, early stopping, L1 and L2 regularization, Dropout (Srivastava et al. 2014) and DropConnect (Wan et al. 2013). Among these methods, Dropout is one of the most popular which has been adopted in many classical network architectures, including AlexNet (Krizhevsky, Sutskever, and Hinton 2012), VGGNet (Simonyan and Zisserman 2014), and Inception (Szegedy et al. 2015; 2016). In Dropout, the output from the previous layer is flattened as a one-dimensional vector and a randomly selected subset of neurons is set to zero for the next layer (Figure 1). In most cases, Dropout is used to regularize the fully-connected layers within CNN and not very suitable for the convolutional layers. One of the main reasons is that Dropout operates on each neuron, while in the convolutional layers each channel consisting of multiple neurons is a basic unit that corresponds to a specific pattern of input image (Zhang et al. 2016). In this work, we propose a novel regularization technique for the stack of convolutional layers[1] which randomly selects channels for the next layer.

Another inspiration of this work comes from the observation that in the stack of convolutional layers within CNN, all the channels generated by the previous layer are treated equally for the next layer. This is not optimal especially for high layers where the features have greater specificity (Zeiler and Fergus 2014; Yosinski et al. 2014; Zhang et al. 2016). For each input image, only a few channels in high layers are activated while the neuron responses in the other channels are close to zero (Zhang et al. 2016). So instead of totally random selection, we propose to select the channels according to the relative magnitude of activation status, which can be treated as a special way to model the interdependencies across the channels. To some extent, our work is similar in spirit to the recent SE-Block (Hu, Shen, and Sun 2017). The detailed comparison between our work and SE-Block will be provided below.

In summary, the main contribution of this work lies in a novel method named Weighted Channel Dropout (WCD) for the regularization of convolutional layers within CNN, which is illustrated in Figure 2. Notably the basic operation unit of WCD is the channel rather than the neuron. Specifically, we first rate the channels output by the previous layer and assign a *score* for each channel. The *score* is obtained by Global Average Pooling, which can acquire a global view of activation status in each channel. Second, regarding to the channel selection, a binary *mask* is generated to indicate

[1]The stack of convolutional layers means the layers before the fully-connected layers, mainly consisting of convolution layers (with ReLU) optionally followed by pooling layers, such as *conv1-pool5* in VGGNet-16.

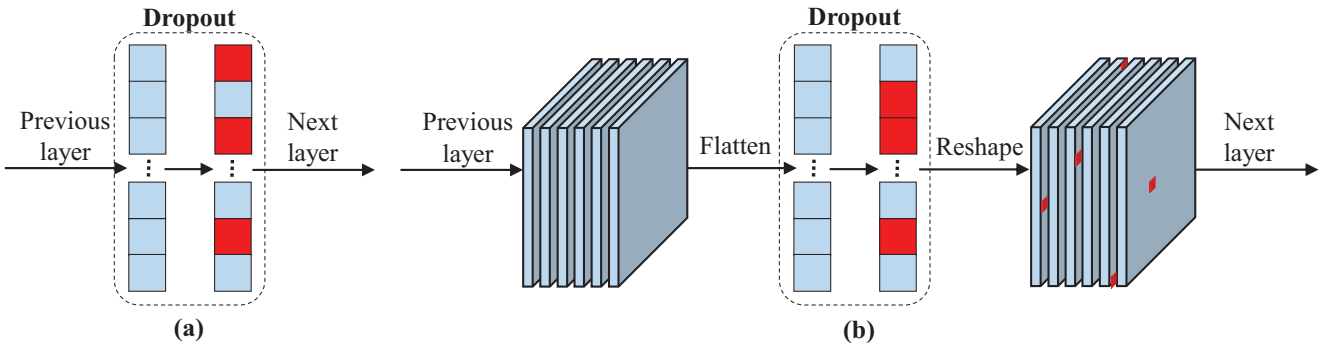

Figure 1: Illustration of Dropout. (a) Dropout in the fully-connected layers. (b) Dropout in the convolutional layers. The neurons in red are randomly selected and set to zero referring to the implementation in Caffe (Jia et al. 2014).

whether each channel is selected or not, and the channels with relatively high *scores* are kept with high probability. We find that the process of generating *mask* according to *score* can be boiled down to a special case of Weighted Random Selection (Efraimidis and Spirakis 2006) and an efficient algorithm is adopted for our purpose. Finally, a Random Number Generator is further attached to *mask* to filter channels for the next layer. It is worth noting that, WCD is parameter-free and only added to the network in training phase with very slight computation cost. The network in test phase remains unchanged and thus the inference cost is not added at all. Besides, WCD can be plugged into any existing networks already pretrained on large-scale ImageNet and requires no re-pretraining, making it appealing for the application on small datasets.

The motivation of WCD is exactly to alleviate the overfitting of finetuning CNN on small datasets, *e.g.*, the bulk of datasets for FGVC (Wah et al. 2011; Khosla et al. 2011; Krause et al. 2013; Maji et al. 2013), through adding more regularization to the stack of convolutional layers besides the fully-connected layers. For example, CUB-200-2011 (Wah et al. 2011) collects only about 30 training images for each class. By introducing more regularization, WCD can help the network learn more robust features from input. For the experiments, we evaluate WCD combining with the classical networks including VGGNet-16 (Simonyan and Zisserman 2014), ResNet-101 (He et al. 2016), Inception-V3 (Szegedy et al. 2016) on CUB-200-2011 (Wah et al. 2011) and Stanford Cars (Krause et al. 2013). Besides, for a thorough evaluation, we also evaluate WCD on Caltech-256 (Griffin, Holub, and Perona 2007) which focuses on generic classification. The extensive results demonstrate that WCD can bring consistent improvements over the baselines.

Related Work

Due to the availability to large training data and GPU accelerated computation, multiple efforts have been taken to enhance CNN for greater capacity since the massive improvement shown by AlexNet (Krizhevsky, Sutskever, and Hinton 2012) on ILSVRC2012. These efforts mainly consist of increased depth (Simonyan and Zisserman 2014; He et al. 2016), enlarged width (Zagoruyko and Komodakis 2016), nonlinear activation (Maas, Hannun, and Ng 2013; He et al. 2015), reformulation of the connection between layers (Huang et al. 2017) and so on. Our method falls into the scope of adding regularization to neural networks. In this field, the previous works include Dropout (Srivastava et al. 2014), DropConnect (Wan et al. 2013), Batch Normalization (Ioffe and Szegedy 2015), DisturbLabel (Xie et al. 2016), Stochastic Depth (Huang et al. 2016) and so forth. Specifically, Dropout randomly selects a subset of neurons and sets them to zero, which is widely used for designing novel networks. DropConnect instead randomly sets the weights between layers to zero. Batch normalization improves the gradient propagation through network by normalizing the input for each layer. DisturbLabel randomly replaces some labels with incorrect values to regularize the loss layer. Stochastic Depth randomly skips some layers in the residual networks. In addition, (Park and Kwak 2016) analyze the effect of Dropout on max-pooling and convolutional layers. (Morerio et al. 2017) extend the standard Dropout by introducing a time schedule to adaptively increase the ratio of dropped neurons.

Among these works, Dropout (Srivastava et al. 2014) is the most popular and more related to our method. Both work by randomly setting a portion of responses of hidden layers to zero in the training. However, there exist clear differences between WCD and Dropout. First, WCD is proposed to regularize the stack of convolutional layers while Dropout is usually inserted between the fully connected layers. Second, WCD differs from Dropout in that it operates on the channels other than the neurons. While in the stack of convolutional layers, each channel is a basic unit. Finally, the neuron selection in Dropout is completely random, in contrast, WCD selects the channels according to their activation status. Actually, Dropout can be seen as a special case of WCD, which will be discussed below. Both WCD and Dropout can be combined in one network, which is exactly the practice in our experiments.

Besides, WCD is similar to the recent SE-Block (Hu, Shen, and Sun 2017) in some degree, both of which share the similar structures (please refer to the paper for details). As aforementioned, due to the selection according to rela-

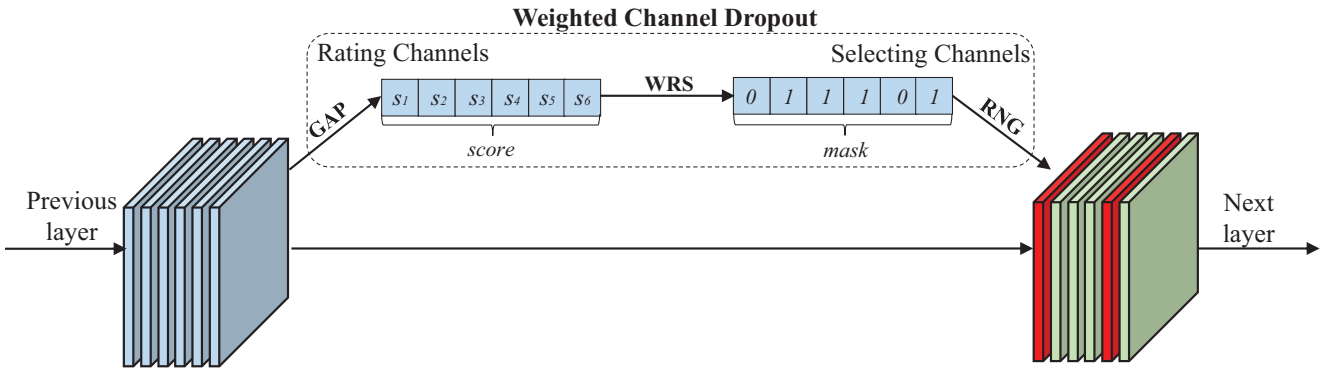

Figure 2: Illustration of Weighted Channel Dropout. **GAP**: Global Average Pooling, **WRS**: Weighted Random Selection, **RNG**: Random Number Generator. The channels are selected according to the activation status. The red ones are set to zero and the green are rescaled from the corresponding input channels.

tive magnitude of activation status, WCD can be treated as a special way to explore the channel relationship, which is the focus of SE-Block. However, compared to SE-Block, first of all, WCD is totally parameter-free and added only in training phase, thus leaving the inference cost not added at all. Second, when combining SE-Block with the existing networks, it requires re-pretraining the whole model on large-scale ImageNet before finetuning on other datasets. Unlike SE-Block, WCD can be plugged into any existing networks and does not need to re-pretrain the whole model, thus fitting well for the application on small datasets. Finally, WCD is usually applied to high convolutional layers within CNN while SE-Block mainly affects early convolutional layers. From this perspective, WCD and SE-Block are complementary to each other.

Our Approach

WCD is designed to provide regularization to the stack of convolutional layers within CNN. Taking VGGNet (Simonyan and Zisserman 2014) as an example, Dropout is inserted among the last three fully-connected layers, while no regularization is deployed in the layers before $pool5$. When finetuning VGGNet on small datasets, more regularization is wished since the overfitting is severe. Here we present some notations which will be used in the following. Formally, in two consecutive convolutional layers, $X = [x_1, x_2, \cdots, x_N]$ denotes the output of previous layer, $\widetilde{X} = [\widetilde{x}_1, \widetilde{x}_2, \cdots, \widetilde{x}_{\widetilde{N}}]$ denotes the input to next layer, N and $\widetilde{N}$ are the number of channels, x_i and $\widetilde{x}_i$ are the i-th channel. In almost all cases, $\widetilde{X} = X$, *i.e.*, there are $\widetilde{N} = N$ and

$$\widetilde{x}_i = x_i, i = 1, 2, \cdots, N \tag{1}$$

Differently, WCD randomly selects the channels from X to construct $\widetilde{X}$ according to the activation status in each channel. It is worth noting that the previous or the next layer could also be pooling layer and so on. Without loss of generality, here we discuss the case of two consecutive convolutional layers. Next we will present the details of WCD, which consists of two steps (*i.e.*, Rating Channels and Selecting Channels) and three modules (*i.e.*, Global Average Pooling, Weighted Random Selection and Random Number Generator). In our method, we still have $\widetilde{N} = N$.

Step 1: Rating Channels

According to the observation in previous works (Zeiler and Fergus 2014; Yosinski et al. 2014; Zhang et al. 2016), the features in high layers of CNN have great specificity (*i.e.*, class-specific in the context of image classification) and only a small fraction of channels are activated for each input image (Zhang et al. 2016). Thus, instead of treating all channels equally and conducting random selection, we propose to first rate the channels and assign each channel a *score*, which is used as the guidance for channel selection in the following step.

Global Average Pooling (GAP). In deep CNN, a neuron in high layers corresponds to a local patch in the input image and a channel consisting of multiple neurons represents a specific pattern (Zhang et al. 2016). In order to rate each channel, we adopt a simple but effective method, *i.e.*, Global Average Pooling, to acquire a global view of activation status in each channel. Formally,

$$score_i = \frac{1}{W \times H} \sum_{j=1}^{W} \sum_{k=1}^{H} x_i(j, k) \tag{2}$$

where W and H are the shared width and height of all channels. There are more sophisticated strategies (Sánchez et al. 2013; Yang et al. 2009; Lin, RoyChowdhury, and Maji 2015; Gao et al. 2016) to obtain the *score* which need further exploration. It is normal to assume that $score_i > 0$ since ReLU is appended behind each convolutional layer in modern CNNs (Simonyan and Zisserman 2014; He et al. 2016; Szegedy et al. 2016).

Step 2: Selecting Channels

After obtaining a $score$ for each channel, it comes to how to select the channels to construct $\widetilde{X}$. Here We use a binary $mask_i$ to indicate whether x_i is kept or not. The probability

Algorithm 1 Weighted Random Selection

Input: $score_i > 0$, $mask_i = 0$, $i = 1, 2, \cdots, N$, wrs_ratio.

Output: $mask_i$, $i = 1, 2, \cdots, N$. The probability of $mask_i = 1$ is $p_i = \frac{score_i}{\sum_{j=1}^{N} score_j}$.

1: For each i, $r_i = random(0, 1)$ and $key_i = r_i^{\frac{1}{score_i}}$.
2: Select the $M = N * wrs\_ratio$ items with the largest key_i and set the corresponding $mask_i$ to 1.

p_i of keeping the channel x_i is set to

$$p_i = \frac{score_i}{\sum_{j=1}^{N} score_j} \tag{3}$$

That is to say, $mask_i$ has the probability p_i to be set to 1, and the channels with relatively high $scores$ are more likely to be kept. In the following we will present the algorithm to achieve this goal, taking the computation cost and efficiency into account.

Weighted Random Selection (WRS). We find that the process of generating $mask$ according to $score$ can be boiled down to a special case of Weighted Random Selection (Efraimidis and Spirakis 2006). An efficient algorithm is adopted for our purpose, which is illustrated in Algorithm 1. Specifically, first, for channel x_i with $score_i$, a random number $r_i \in (0, 1)$ is generated and a key value key_i is computed as

$$key_i = r_i^{\frac{1}{score_i}} \tag{4}$$

Then M items with the largest key values are selected and the corresponding $mask_i$ are set to 1. Here $wrs\_ratio = \frac{M}{N}$ is a hyper-parameter of WCD, indicating how many channels are kept after WRS. The algorithm is computationally efficient, which can set $mask_i$ to 1 with the probability p_i shown in Equation 3. Please refer to (Efraimidis and Spirakis 2006) for more details about the algorithm.

Random Number Generation (RNG). Going further, for small datasets, the training usually starts from the models pretrained on ImageNet instead of from scratch. In high convolutional layers within the pretrained model, the disparity between channels is large, *i.e.*, only a few channels are assigned relatively high activation values with the others close to zero (Zhang et al. 2016). If we totally select channels according to $score$ in these layers, it is possible that for a given image, the sequence of selected channels is basically the same in each forward process[2], which is not desired. To alleviate this, we further propose to add a binary Random Number Generator rng with the parameter q to $mask_i$. Thus in the case that $mask_i$ is set to 1, x_i still has the probability $1 - q$ to be not selected.

[2]For example, suppose that $p_i \gg p_j, \forall j \neq i$, $mask_i$ is almost certain to be set to 1

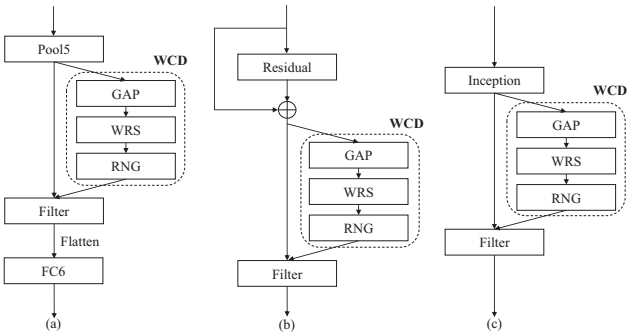

Figure 3: The schema to combine WCD with the existing networks. (a) WCD with VGGNet. (b) WCD with ResNet. (c) WCD with Inception.

Summary

On the whole, $\widetilde{X}$ is constructed as follows:

$$\widetilde{x}_i = \begin{cases} \alpha x_i & if\ mask_i = 1\ and\ rng = 1 \\ 0 & otherwise \end{cases} \tag{5}$$

where $mask_i$ has the probability p_i to be set to 1, rng generates 1 with the probability q, $\widetilde{x}_i = 0$ means that all the neurons in its $W \times H$ zone are set to zero. The coefficient α is used to reduce the bias between training and test data. In our implementation, α is empirically set to

$$\alpha = \frac{\sum_{j=1}^{N} score_j}{\sum_{j \in \widetilde{M}} score_j} \tag{6}$$

where $\widetilde{M}$ denotes the set of channels which are finally selected, the numerator is to sum the $scores$ of all channels and the denominator is to sum the $scores$ of selected channels. WCD is added only in training phase, while at inference time, all channels are sent into the next layer.

Moreover, let $keep\_ratio$ denote the ratio of how many channels are kept for $\widetilde{X}$ in training[3], and we have

$$keep\_ratio = \frac{|\widetilde{M}|}{N} \approx wrs\_ratio \times q \tag{7}$$

where $|\widetilde{M}|$ denotes the number of elements in $\widetilde{M}$, wrs_ratio and q are two hyper-parameters of WCD. Usually we have $0 < wrs\_ratio < 1$ and $0 < q < 1$. Besides, WCD also unifies the following special cases:

1. $wrs\_ratio = 1, 0 < q < 1$. $\widetilde{X}$ is constructed by totally random selection from the channels in X, which is denoted as Channel-Wise Dropout. When $W = H = 1$, it is equivalent to the original Dropout.
2. $0 < wrs\_ratio < 1, q = 1$. The channels with $mask_i = 1$ will be surely kept.

The cases that either wrs_ratio or q is set to 0 make no sense. WCD can be also treated as a more generic version

[3]The $keep\_ratio$ is set for the convenience of description in the following, which is not directly used to select channels in WCD.

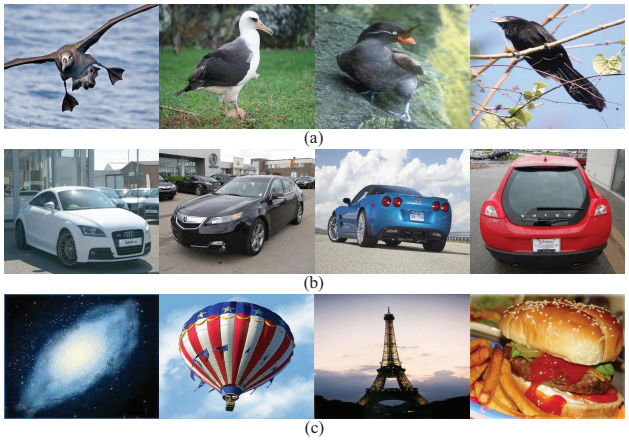

Figure 4: Example images. (a) CUB-200-2011. (b) Stanford Cars. (c) Caltech-256.

of Channel-Wise Dropout and brings additional flexibility. For instance, since the channels with high *scores* are more likely to be selected, the selected channels remain discriminative even with a low $keep\_ratio$ and would not hinder the convergence of network.

Application

WCD can be theoretically plugged into any two consecutive layers within CNN. In practice, it is usually used to regularize the stack of convolutional layers. The schema to integrate WCD with the classical networks including VGGNet (Simonyan and Zisserman 2014), ResNet (He et al. 2016) and Inception (Szegedy et al. 2016) are displayed in Figure 3. Specifically, inspired by the observation that the channels in early convolutional layers are more related to each other (Tompson et al. 2015) , we mainly deploy WCD behind the high layers, such as *pool4* and *pool5* in VGGNet-16, *res5a* and *res5c* in ResNet-101. To stress again, as a lightweight parameter-free module, WCD is added to these networks only in training phase and the network in test phase is unchanged. Besides, it requires no re-pretaining on ImageNet and thus can be easily deployed when finetuning these networks on small datasets.

Experiment

Experimental Setup

In this section, we evaluate WCD combining with VGGNet-16 (Simonyan and Zisserman 2014), ResNet-101 (He et al. 2016), Inception-V3 (Szegedy et al. 2016) on CUB-200-2011 (Wah et al. 2011), Stanford Cars (Krause et al. 2013) and Caltech-256 (Griffin, Holub, and Perona 2007). On these datasets, the scale of training set is rather small and thus overfitting is more likely to occur.

All the models are implemented with Caffe (Jia et al. 2014) on Titan-X GPUs. WCD is added to the networks in training phase with the original layers remaining unchanged. The hyper-parameters including wrs_ratio and q are set by cross validation and keep consistent on the similar datasets such as CUB-200-2011 and Stanford Cars. The training starts from the model pretrained on ImageNet. Stochastic gradient descent (SGD) is used for the optimization. The initial learning rate is set to 0.001 and reduces to its 1/10 three times until convergence. In all experiments, the images are randomly flipped and cropped before passing into the networks, and no other data augmentation is used. The inference is done with one center crop of the test images. Finally, the top-1 accuracy is taken as the metric for evaluation.

Table 1: The parameter settings of WCD on CUB-200-2011 and Stanford Cars.

	VGGNet-16	ResNet-101	Inception-V3
layer_1	*pool4*	*res5a*	*reduction_b*
wrs_ratio_1	0.8	0.8	0.8
q_1	0.9	0.9	0.9
layer_2	*pool5*	*res5c*	*inception_c2*
wrs_ratio_2	0.9	0.8	0.8
q_2	0.5	0.9	0.9

Table 2: The performance comparison on CUB-200-2011 and Stanford Cars.

	CUB	Cars
VGGNet-16	72.31%	86.11%
WCD-VGGNet-16	76.10% (**+3.79%**)	88.28% (**+2.17%**)
ResNet-101	76.45%	87.84%
WCD-ResNet-101	77.22% (**+0.77%**)	88.37% (**+0.53%**)
Inceptin-V3	83.98%	93.17%
WCD-Inception-V3	84.52% (**+0.54%**)	93.41% (**+0.24%**)

CUB-200-2011 & Stanford Cars

CUB-200-2011 (Wah et al. 2011) is a widely-used fine-grained dataset which collects images in 200 birds species. For each class, there are about 30 images for training. Some example images are shown in Figure 4(a).

VGGNet-16, ResNet-101, Inception-V3 are adopted as the baselines. The parameters of integrating WCD with these networks are shown in Table 1. For all three networks, WCD is deployed behind the last two layers conducting the feature dimension reduction in the stack of convolutional layers, *e.g.*, *pool4* and *pool5* in VGGNet-16, *res5a* and *res5c* in ResNet-101. Please refer to (Simonyan and Zisserman 2014), (He et al. 2016) and (Szegedy et al. 2016) for the details of network structures. As shown in Table 2, WCD can bring consistent improvements over the baselines. For VGGNet-16, WCD achieves a significant 3.79% improvement over the base model.

Compared to CUB-200-2011, Stanford Cars (Krause et al. 2013) is a fresh domain concentrating on the categorization of cars, such as Make, Model and Year, as shown in Figure 4(b). There are about 40 training images for each of 196 classes. The parameter settings of WCD are the same

Table 3: The parameter settings of WCD and performance comparison on Caltech-256. *-reported on the reduced test set consisting of 20 images per class.

	VGGNet-16	ResNet-101	Inception-V3
layer_1	*pool5*	*res5c*	*inception_c2*
wrs_ratio_1	0.8	0.8	0.8
q_1	0.9	0.9	0.9
Baseline	72.31%	78.00%	79.52%
With WCD	72.86% (**+0.55%**)	78.30% (**+0.30%**)	80.61% (**+1.09%**)
Baseline*	70.91%	77.28%	78.81%
With WCD*	71.83% (**+0.92%**)	77.94% (**+0.66%**)	79.92% (**+1.11%**)

as those on CUB-200-2011 shown in Table 1, and the performance comparison is also shown in Table 2. This dataset is more distinguishable than CUB-200-2011 and the baselines are relatively high, while the models with WCD still consistently outperform the baselines.

Discussion. The recent work (Zheng et al. 2017) lists the performance reported on CUB-200-2011 and Stanford Cars, where the methods can be roughly divided into two categories. The first one is to encode CNN features for more discriminative representation, such as Bilinear CNN (Lin, RoyChowdhury, and Maji 2015) and Compact Bilinear CNN (Gao et al. 2016). The second is to exploit the attention mechanism, such as RA-CNN (Fu, Zheng, and Mei 2017) and MA-CNN (Zheng et al. 2017). Our method does not belong to either of the above categories and the focus of this paper is not to report state-of-the-art performance. WCD is a fairly generic method to alleviate the overfitting when finetuning CNN on small datasets, which can be integrated into these existing models. We take some preliminary experiments to combine WCD with Compact Bilinear CNN (Gao et al. 2016), and find that WCD can help outperform this strong baseline (84.88% *vs.* 84.01%)[4].

Caltech-256

Both CUB-200-2011 and Stanford Cars belong to the field of fine-grained visual categorization. In order to obtain a thorough evaluation of WCD, here we further evaluate it on Caltech-256 (Griffin, Holub, and Perona 2007) which focuses on generic classification. Some example images of Caltech-256 are shown in Figure 4(c), which display larger inter-class difference than fine-grained datasets. There is no split way provided in the dataset, and for each class we randomly select 20 images for training with the rest as test set. The experimental settings are a little different from those on CUB-200-2011 and Stanford Cars. Specifically, WCD is appended behind the last layer conducting the feature dimension reduction in the stack of convolutional layers, such as *pool5* in VGGNet-16, *res5c* in ResNet-101. The performance comparison as well as the parameter details is shown in Table 3. It can be seen that, WCD also works well on Caltech-256 and helps achieve superior performance over the base model.

[4]WCD is added after *pool5* with the other settings unchanged.

Table 4: The ablation study of WCD. The results are reported on CUB-200-2011.

	Approach	Settings	Accuracy
A	VGGNet-16	*baseline*	72.31%
B	VGGNet-16+ *pool5*_Dropout	$dropout\_ratio = 0.5$	73.90%
C	VGGNet-16+ *pool5*_SE-Block	SE-Block not pretrained	72.71%
D		SE-Block pretrained	73.05%
E	VGGNet-16+ *pool5*_WCD	$wrs\_ratio = 0.9$ $q = 0.5$	**75.60%**
F	VGGNet-16+ *pool5*_WCD	$wrs\_ratio = 1$ $q = 0.45$	75.02%
G		$wrs\_ratio = 0.45$ $q = 1$	73.28%
H		$wrs\_ratio = 0.9$ $q = 0.5$	**75.60%**
I	VGGNet-16+ *pool5*_WCD	$wrs\_ratio = 1$ $q = 0.25$	73.16%
J		$wrs\_ratio = 0.25$ $q = 1$	72.16%
K		$wrs\_ratio = 0.5$ $q = 0.5$	**75.33%**

Discussion. We notice that Caltech-256 (Griffin, Holub, and Perona 2007) exhibits long-tail distribution where the number of images for each class largely varies from each other. Thus we further report the results on the reduced test set containing the same number of images for each class. Specifically, the training set remains unchanged, and we randomly select another 20 images from each class as the test set which has no overlap with the training set. The performance comparison is shown in the last two rows of Table 3, which further validate the effectiveness of WCD.

Ablation Study

In this subsection, we take extensive experiments to analyze the behaviors of WCD.

Channel dropout analysis. WCD is based on the observation that in high convolutional layers of CNN (pretrained on ImageNet or finetuned on the specific dataset), for an input image, only a few channels are activated with relatively high values while the neuron responses in the other channels are close to zero. The phenomenon is validated by the experiments in (Zhang et al. 2016). Noticing that the experiments in (Zhang et al. 2016) were performed on image patches, we conduct the similar experiments on full images and get the same observations. For example, for the input images randomly selected from CUB-200-2011, nearly half of channels in the *pool5* of VGGNet-16 hold the responses that are zero or very close to zero.

Comparison with Dropout and SE-Block. As shown in the first part of Table 4, a Dropout layer is inserted af-

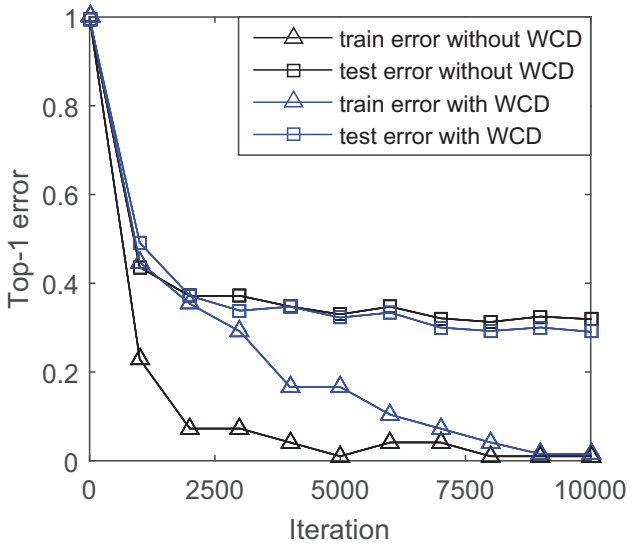

Figure 5: Effect of WCD on network training. The experiments are conducted on CUB-200-2011 with VGGNet-16 as the base model. Best viewed in color.

ter *pool5* of VGGNet-16 (Row B), and then a SE-Block is added at the same place (Row C and Row D). For SE-Block, VGGNet-16 with it is first directly finetuned on the specific dataset from the pretrained model on ImageNet. Then, given that SE-Block contains two fully-connected layers, we further re-pretrain the whole model on ImageNet before finetuning on small datasets. The resulting accuracy is a little better than that without re-pretraining, but still inferior to that with WCD (Row E).

Special cases of WCD. Row F and Row G of Table 4 are the two special cases of WCD as aforementioned, *i.e.*, either wrs_ratio or q is set to 1. The comparison between Row F and Row H shows the effectiveness of WCD compared to Channel-wise Dropout, while the comparison between Row G and Row H indicates the necessity of RNG in WCD. Besides, the results in the last three rows of Table 4 further demonstrates the superiority of WCD, indicating that WCD enables a low *keep_ratio* ($\approx wrs\_ratio \times q = 0.25$). It may suit for the cases where the training images are very rare, such as in medical image analysis (Qayyum et al. 2017).

Effect on network training. Figure 5 shows the effect of WCD on training with the settings shown in Table 1. It can be seen that, the curve of training error with WCD drops more slowly while the resulting test error is lower, proving that WCD can reduce overfitting in the training phase.

Computation cost of WCD. Table 5 provides some complexity statistics of WCD, which indicates that the computation cost introduced by WCD is negligible. Besides, these additional cost is introduced only in training phase and the inference cost is not increased at all.

Conclusion

In this work, we deal with the regularization of CNN by proposing a novel method named WCD for the stack of convolutional layers. Specifically, WCD filters the channels according to their activation status for the next layer. It consists of two steps, *i.e.*, Rating Channels and Selecting Channels, and three modules, *i.e.*, Global Average Pooling, Weighted Random Selection and Random Number Generator. As a whole, WCD is a lightweight component which can be integrated into any existing models with negligible computation cost introduced only in training phase. Its characteristics, *e.g.*, parameter-free and no need to re-pretrain on ImageNet, make it well-suited for the application on small datasets. Finally, the experimental results with VGGNet-16, ResNet-101, Inception-V3 on multiple datasets show the robustness and superiority of WCD. For the future work, we plan to apply WCD to other types of visual tasks, such as object detection.

Table 5: Computation cost introduced by WCD. The statistics are obtained on a Titan GPU with $batch\_size = 32$. The training time is averaged over the first 100 iterations.

	GPU memory (MB)	Training time (sec/iter)
VGGNet-16	5753	0.74
WCD-VGGNet-16	5817	0.83

Acknowledgment

This work is supported by the National Natural Science Foundation of China under Grant 61673362 and 61836008, Youth Innovation Promotion Association CAS (2017496), and the Fundamental Research Funds for the Central Universities.

References

Efraimidis, P. S., and Spirakis, P. G. 2006. Weighted random sampling with a reservoir. *Information Processing Letters* 97(5):181–185.

Fu, J.; Zheng, H.; and Mei, T. 2017. Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition. In *CVPR*.

Gao, Y.; Beijbom, O.; Zhang, N.; and Darrell, T. 2016. Compact bilinear pooling. In *CVPR*.

Griffin, G.; Holub, A.; and Perona, P. 2007. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*.

Hu, J.; Shen, L.; and Sun, G. 2017. Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*.

Huang, G.; Sun, Y.; Liu, Z.; Sedra, D.; and Weinberger, K. Q. 2016. Deep networks with stochastic depth. In *ECCV*.

Huang, G.; Liu, Z.; van der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *CVPR*.

Ioffe, S., and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*.

Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R.; Guadarrama, S.; and Darrell, T. 2014. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*.

Khosla, A.; Jayadevaprakash, N.; Yao, B.; and Li, F.-F. 2011. Novel dataset for fine-grained image categorization: Stanford dogs. In *CVPR Workshop*.

Krause, J.; Stark, M.; Deng, J.; and Fei-Fei, L. 2013. 3d object representations for fine-grained categorization. In *ICCV Workshop*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*.

Lin, T.-Y.; RoyChowdhury, A.; and Maji, S. 2015. Bilinear cnn models for fine-grained visual recognition. In *ICCV*.

Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; and Berg, A. C. 2016. Ssd: Single shot multibox detector. In *ECCV*.

Maas, A. L.; Hannun, A. Y.; and Ng, A. Y. 2013. Rectifier nonlinearities improve neural network acoustic models. In *ICML*.

Maji, S.; Rahtu, E.; Kannala, J.; Blaschko, M.; and Vedaldi, A. 2013. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*.

Morerio, P.; Cavazza, J.; Volpi, R.; Vidal, R.; and Murino, V. 2017. Curriculum dropout. In *ICCV*.

Park, S., and Kwak, N. 2016. Analysis on the dropout effect in convolutional neural networks. In *ACCV*.

Qayyum, A.; Anwar, S. M.; Majid, M.; Awais, M.; and Alnowami, M. 2017. Medical image analysis using convolutional neural networks: A review. *arXiv preprint arXiv:1709.02250*.

Sánchez, J.; Perronnin, F.; Mensink, T.; and Verbeek, J. 2013. Image classification with the fisher vector: Theory and practice. *International Journal of Computer Vision* 105(3):222–245.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *CVPR*.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the inception architecture for computer vision. In *CVPR*.

Tompson, J.; Goroshin, R.; Jain, A.; LeCun, Y.; and Bregler, C. 2015. Efficient object localization using convolutional networks. In *CVPR*.

Wah, C.; Branson, S.; Welinder, P.; Perona, P.; and Belongie, S. 2011. The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology.

Wan, L.; Zeiler, M.; Zhang, S.; Cun, Y. L.; and Fergus, R. 2013. Regularization of neural networks using dropconnect. In *ICML*.

Wang, L.; Xiong, Y.; Wang, Z.; Qiao, Y.; Lin, D.; Tang, X.; and Van Gool, L. 2016. Temporal segment networks: towards good practices for deep action recognition. In *ECCV*.

Xie, L.; Wang, J.; Wei, Z.; Wang, M.; and Tian, Q. 2016. Disturblabel: Regularizing cnn on the loss layer. In *CVPR*.

Yang, J.; Yu, K.; Gong, Y.; and Huang, T. 2009. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*.

Yosinski, J.; Clune, J.; Bengio, Y.; and Lipson, H. 2014. How transferable are features in deep neural networks? In *NIPS*.

Zagoruyko, S., and Komodakis, N. 2016. Wide residual networks. In *BMVC*.

Zeiler, M. D., and Fergus, R. 2014. Visualizing and understanding convolutional networks. In *ECCV*.

Zhang, X.; Xiong, H.; Zhou, W.; Lin, W.; and Tian, Q. 2016. Picking deep filter responses for fine-grained image recognition. In *CVPR*.

Zheng, H.; Fu, J.; Mei, T.; and Luo, J. 2017. Learning multi-attention convolutional neural network for fine-grained image recognition. In *ICCV*.