IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 18, 2020, accepted March 23, 2020, date of publication March 27, 2020, date of current version April 15, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2983774

Sequence-Dropout Block for Reducing Overfitting Problem in Image Classification

LEDAN QIAN[1], LIBING HU[2], LI ZHAO[3], TAO WANG[3], AND RUNHUA JIANG[3]

[1]College of Mathematics and Physics, Wenzhou University, Wenzhou 325035, China
[2]Training Center, Zhejiang College of Security Technology, Wenzhou 325016, China
[3]College of Computer Science and Artificial Intelligence, Wenzhou University, Wenzhou 325035, China

Corresponding authors: Libing Hu (fox2000hu@163.com) and Li Zhao (lizhao@wzu.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB1004904, in part by the Zhejiang Provincial Natural Science Foundation under Grant LQ19F020005, and in part by the Project of Science and Technology Plans of Wenzhou under Grant G20180011 and Grant G20190031.

ABSTRACT Overfitting is a common problem for computer vision applications It is a problem that when training convolution neural networks and is caused by lack of training data or network complexity. The novel sequence-dropout (SD) method is proposed in this paper to alleviate the problem of overfitting when training networks. The SD method works by dropping out units (channels of feature) from the network in a sequence, replacing the traditional operation of random omitting. Sophisticated aggregation strategies are used to obtain the global information of feature channels, and channel-wise weights are produced by gating mechanism. The SD method then selectively drops out the feature channels according to the channel-wise weights that represent the importance degree of each channel. The proposed SD block can be plugged into state-of-the-art backbone CNN models such as VGGNet and ResNet. The SD block is then evaluated on these models, demonstrating consistent performance gains over the baseline model on widely-used benchmark image classification datasets including MNIST, CIFAR-10, CIFAR-100, and ImageNet2012. Experimental results demonstrate that the superior performance of the SD block compared to other modern methods.

INDEX TERMS Convolutional networks, image classification, overfitting, sequence-dropout.

I. INTRODUCTION

Overfitting is a typical problem for computer vision applications [1]–[5]. It is a problem that artificial neural networks with non-linear hidden layers produce complex co-adaptations on the training data. It leads to the predictions on the testing set are worse than on the training sets. A variety of techniques [6]–[11] have been developed to tackle overfitting problem. However, overfitting still remains a major challenge when training large neural networks or having very small amounts of data, and tackling this challenging well can benefit many computer vision tasks, such as classification [12], [13], denoising [14] and tracking [49], [50].

Over past decades, several researches focus on functional solutions [15], trying to extend convolutional neural networks (CNNs) for application on smaller datasets to reduce overfitting. For example, a typical technique is Dropout [16], [17], which improves the generalization ability of networks by stopping the feature detectors with random probability p at each training epoch. Batch normalization (BN) [18] is another technique, it has been applied to modern CNNs to enhance the generalization ability by normalizing the set of activations in a layer. In addition, many other strategies for reducing overfitting concentrate on the network architecture. Therefore, a sequence of progressively complex architectures are proposed, such as AlexNet [2], VGG-16 [3], ResNet [19], Inception-V3 [5], and DenseNet [13]. CNNs with more complex architectures can learn various feature representations of the input image by using different filters in a convolution layer, and the layer output (feature map) indicates the importance (measured by *matching degree*) of the learned feature to identify the input image. For instance, SENet [20] just exploits the characteristic of convolution to selectively enhance and suppress features according to channel-wise weights acquired by *Global Average Pooling* (GAP) [21], embedded in CNNs to improve the generalization performance.

Through investigating the characteristics of Dropout [17] and SENet [20], we construct a novel structure unit named as "*sequence-dropout*" (SD) block. It is introduced into CNNs

The associate editor coordinating the review of this manuscript and approving it for publication was Jeon Gwanggil.

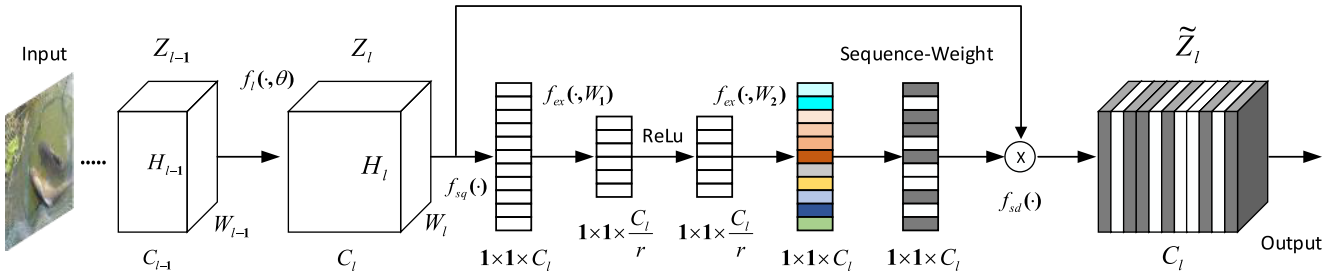


FIGURE 1. The SD building block is placed between the convolution layers. Z_l is the feature map of layer l with spatial dimensions $W_l \times H_l$ and C_l channels. r is a positive integer. The section of $1 \times 1 \times C_l$ with different colors symbolizes the different importance degree of feature map channels, closely following is *sequence-weight*. Black denotes the reassignment to 1 and white symbolizes reassignment to 0. In the output $\tilde{Z}_l$, black features show that the feature map channels are preserved and white features show that the feature map channels are temporarily removed.

to reduce the problem of overfitting on image classification task. The proposed SD block establishes a novel mechanism of feature recalibration which highlights the most critical aspect of the proposed method, to generate the values representing the *matching degree* of feature maps, selectively dropping the feature map channels or neurons with low *matching degree* during training to improve the model generalization performance. The method inherits the advantages of Dropout, simplifying the model with residual architecture [13], [19], and improves the robustness of the model in cooperation with other regularization methods (*e.g.* L2, BN).

The architecture of the proposed SD block is illustrated in Fig. 1. For any given transformation $f_l(\cdot, \theta)$: $Z_{l-1} \rightarrow Z_l$, $Z_{l-1} \in \mathbb{R}^{W_{l-1} \times H_{l-1} \times C_{l-1}}$, $Z_l \in \mathbb{R}^{W_l \times H_l \times C_l}$, (e.g. a convolution or a set of convolutions), the constructed SD block includes three calculation steps (see Section III for further details): *squeeze* $f_{sq}(\cdot)$, *excitation* $f_{ex}(\cdot)$ and *sequence-dropping* $f_{sd}(\cdot)$, which are elaborated as follows. Firstly, using $f_{sq}(\cdot)$ calculation (*squeeze* operation), the feature map Z_l of l layer is polymerized into a $1 \times 1 \times C_l$ channel descriptor across spatial dimensions $W_l \times H_l$. The obtained descriptor contains the information of global distribution for channel-wise feature responses, which makes the information from the global receptive domain of the network to be utilized by its lower layers. Then, the descriptor is passed through *excitation* operation including $f_{ex}(\cdot, W_1)$, $f_{ex}(\cdot, W_2)$ and *Sequence-Weight*, which utilizes a gating mechanism to capture channel-wise dependencies, generating the weights of importance for each feature channel. Finally, through $f_{sd}(\cdot)$ calculation, the weights are reassigned to get the *matching degree* and the feature map channels are dropped out by channel-wise multiplication of the *matching degree*.

The proposed SD block abandons randomness used in Dropout and preserves important features to the greatest extent, significantly reducing overfitting on image classification (the experiment results are detailed in Section V). Under the same training condition, comparing with L2, BN and Dropout, our method achieves the lower test classification error 0.64%. Further more, combining our method with L2 and BN further reduces the error to 0.56%.

In the following sections, related work is first introduced, focusing on the evolution of solutions to reduce overfitting in Section II followed by a detailed description of the proposed SD block in Section III. The architecture and computational complexity of network embedded the SD block is analyzed in Section IV and our experiments are reported in Section V. Finally, the Section VI concludes this paper.

II. RELATED WORK

Overfitting is a common problem in neural networks. To solve this issue, a wide range of works have been carried out in recent years. Several classical approaches are approximately summarized as follows.

A. DATA AUGMENTATION

Increasing the amount of data is an effective way to prevent overfitting for networks. Because more training data means that a deeper network can be used for training to obtain superior results [22]. Thus, a lot of datasets are built such as ImageNet [23], COCO [24], and Objects365 [25]. However, it is laborious to substantially increase the data, and it is unable to estimate how much data is enough. Data augmentation [6] can augment data with certain rules on the existing data. It is based on basic image manipulation including Color Jittering [2], Random Crop [2], Scale Jittering [19], Flip [2], Random Erasing [26], Noise injection [27], Kernel filters [28], Mixing images [29], etc. In addition, neural networks are being utilized to further the enhancement of data augmentation techniques. It is divided into Feature space augmentation, Adversarial training, GAN-based, Neural Style Transfer, Meta learning Data Augmentations [15].

B. MODEL MODIFICATION AND SELECTION

Selecting an appropriate complexity network architecture can also reduce the overfitting problem. Weight sharing [30] is an effective way to reduce the weights that need to be learned. In addition, choosing the right activation function also improves the generalization ability of the model. For example, Rectified linear units (ReLUs) mentioned in [31] helps the model avoid the vanishing gradient problem and makes it has a better generalization. Maxout units [32] and Stochastic pooling [33] (a noisy version of max-pooling)

are designed for regularization. Batch normalization [18] and Ghost batch normalization [34] are other paradigms of regularization technique, which fix the activation input distribution to avoid the problem of ''Internal Covariate Shift'' in a layer.

In addition to modifying the model architecture, the generalization can also be improved by the selection of learning mode, loss function and optimization method. Multi-task learning, meta-learning and transfer learning make the model share representation information between related tasks, and achieve better generalization performance. The cost function can also have a regularizing effect. For instance, numerous research works [35], [36] introduce regularizer [8], [10], [11] into the cost function to achieve regularization. Essentially, overfitting is caused by over training.

C. MODEL COMBINATION

Model combination is also a way to reduce overfitting problem. It assembles several models, weakens the influence of outliers and the characteristics of each model, and maintains commonality between models. It trains several models and uses the average output of each model as a result to improve the model performance. For example, the approach of Bagging (bootstrap aggregating) [37] trains several different models separately, and then all the models vote on the output for test examples. Boosting [38] is an another example of model combination. It is achieved by training a series of simple neural networks to acquire weights and get the final results by averaging these weights. Different from above two techniques, The technique of Stacking [39] outputs a final prediction result according to the prediction results from different weak models (heterogeneous weak learner).

Although model combination can alleviate the problem of overfitting to some extent, it is usually hard to train the model with large amounts of parameters. Dropout [16] is proposed to solve this problem. It provides a method of approximately combining exponentially various neural network architectures efficiently to prevent overfitting. It randomly shields a certain proportion of neurons, makes the output uncertain about which features it is combining, and may mask the features that play an important role in the output results. Numerous research works [40]–[42] have carried out on basis of Dropout.

III. APPROACH

The constructed SD block can be applied to existing state-of-the-art deep architecture to reduce model overfitting. The calculation of SD block is divided into three parts: channel compression calculation, channel weight calculation, and sorting and dropping calculation. Similar to dropout involving modifications to the model architecture, the SD block can be placed in two locations in the model architecture: between the convolution layers and behind the FC layer. There is a small distinction between the calculation methods of the two locations, being that the features must be compressed on the channel dimension in the FC layer. A diagram of the SD building block is shown in Fig. 1. The calculation of the SD block embedded between convolution layers are discussed as follows.

The SD block is added after the convolution layer of ℓ in any given network model. The convolutional transformation f_l of ℓ layer in the model architecture ($\ell \subseteq [1, L]$): $f_l : Z_{l-1} \to Z_l, Z_{l-1} \in \mathbb{R}^{W_{l-1}\times H_{l-1}\times C_{l-1}}, Z_l \in \mathbb{R}^{W_l\times H_l\times C_l}$. Right here we set the convolutional filter involved in convolutional calculation to be $K_l = [k_1, k_2, \ldots, k_C]_l$, in which k_C is the convolutional filter of c channel. Through the convolutional transformation f_l, we will get $Z_l = [z_1, z_2, \ldots, z_C]_l$, z_C is calculated by convolution of $k_C = \left[k_C^1, k_C^2, \ldots, k_C^{C'}\right]$ and $Z_{l-1} = \left[z^1, z^2, \ldots, z^{C'}\right]_{l-1}$ (the bias term is omitted in the formula):

$$z_C = k_C * Z_{l-1} = \sum_{n=1}^{C'} k_c^n * Z_{l-1}^n \tag{1}$$

As illustrated in the above formula, the value of the current layer is generated through the sum of all channels in the upper layer, and the interdependencies between channels are implicitly embedded in it. The purpose is to obtain the values representing the importance of each channel, which serve as a basis for judgment of which low-importance feature map channels (or neurons) can be dropped in order to obtain network generalization. To achieve this, the interdependencies are explicitly modeled between channels before entering the next convolution and the output value is considered as the importance degree of each feature channel after feature selection. According to the output values, the weights of the non-important channels are set to the value of 0, which means that the channels are temporarily removed from the network. Important channels that must remain are given the value of 1, then an elimination matrix composed of 0 and 1 is achieved. Finally, the SD of the original feature on the channel dimension is achieved by using channel-wise multiplication between the elimination matrix and the previous feature map.

To summarize, the method is implemented in three steps: *squeeze* (see Section III-A), *excitation* (see Section III-B), and *sequence-dropping* (see Section III-C), as detailed below.

A. SQUEEZE: EXTRACTING GLOBAL INFORMATION OF EACH CHANNEL

The characteristics of CNNs show that each unit in output Z only perceives the local receptive fields, and cannot make use of relevant information outside the locality. In order to calculate the weights of each channel, the channel global information must first be obtained, which can be achieved by combining the information of local receptive field at higher layers. For this purpose, the output Z is minimized through the spatial dimension direction and the two-dimensional feature map of each channel to a real number by using global average pooling (or other more sophisticated aggregation

strategies). Formally, a statistic $S \in \mathbb{R}^c$ is generater by shrinking Z through spatial dimensions $W \times H$, where the c-th element of S is calculated by:

$$s_c = f_{sq}(z_c) = \frac{1}{W \times H} \sum_{i=1}^{W} \sum_{j=1}^{H} z_c(i, j) \tag{2}$$

The dimension of output S is consistent with the number of feature channels of input. To some extent, the real number S has a certain global receptive field, which represents the global distribution of response on feature channels, and enables the layer close to the input to obtain the global receptive field. This process is useful in many feature engineering works [3].

B. EXCITATION: CAPTURE CHANNEL-WISE DEPENDENCIES

The global information of each channel is generated by the *squeeze* operation, which can be used to explicitly model channel-wise dependencies. To achieve this, a gating mechanism similar to RNN [43] is utilized to generate the weights for each feature map by parameter learning. The formula is as follows:

$$E = f_{ex}(S, W) = g(S, W) = W_2 \delta(W_1 S) \tag{3}$$

where δ is ReLUs [31] function, and the parameters of W_1 and W_2 are learned to model the channel-wise dependencies, $W_1 \in \mathbb{R}^{\frac{C}{r} \times C}$, $W_2 \in \mathbb{R}^{C \times \frac{C}{r}}$. To reduce the parameter for computation complexity, and make the model block more nonlinear to better fit the complex correlation between channels, two FC layers are adopted to form a bottleneck architecture. This process is realized in three steps. Firstly, the feature dimension is reduced to $1/r$ through the first FC layer with parameter W_1. It is then activated by the ReLUs function, and finally, the second FC layer with parameter W_2 is adopted to return the dimension back to the original number. The final output E is the weight of importance for each feature channel whose dimension is consistent with the number of feature channels.

C. SEQUENCE-DROPPING: SEQUENCE WEIGHT AND DROPPING OUT FEATURES

The output E of Eq. (3) is the weight of feature channels, which reflects the importance degree of each channel. Judging by this value, in the last step of SD block, the feature map channels are dropped by setting a dropout ratio p. This thins the network to carry out the generalization. The dropout ratio is most commonly set to 0.5 [16], [17], meaning that half the feature channels (neurons) are dropped. In this stage, the weights of each feature channel must be sequenced (termed as "sequence-weight"). However, the *sequence-weight* does not actually change the location of each feature channel, it only obtains the index value of the sorted channels. Following this, the weights of each feature channel are recalibrated. Reassigning the weights of non-important feature channels to 0 indicates that the feature channels need to be dropped out, while assigning the weights of important feature channels to 1 indicates that no dropout is required. The pseudocode for the calculation process is as Algorithm 1.

Algorithm 1 Implementation of Sequence-Weight in SD Block

Input: The set of channel sequence weight, E; the features (or neurons) dropout ratio, p

Output: The final output is reassigned E. The order of the sequence remains invariant, while the value of its elements is set to 0 or 1 according to the filters by using *for* loop

```
Sort the values of each element in sequence E, E_sorted = E.sort();
Obtain the index c of each element of the E_sorted by using argsort(), c = E_sorted.argsort();
for i = 1; i ≤ c.size(); do
    if i ≤ int(len(E) * p) then
        set E(c) to 0;
    else
        set E(c) to 1;
Final ;
Return E;
```

After the *sequence-weight* calculation, the weight sequence E of the feature channels consists of 0 and 1. The final output of the SD block is then obtained by using reassigned E to recalibrate the feature of the original convolution layer:

$$\widetilde{z}_c = f_{sd}(z_c, E_c) = E_c \cdot z_c \tag{4}$$

Here $f_{sd}(\cdot)$ donates channel-wise multiply, feature mapping $z_c \in \mathbb{R}^{W \times H}$, the block output $\widetilde{z} = [\widetilde{z}_1, \widetilde{z}_2, \cdots, \widetilde{z}_c]$. Fig. 2 depicts the output of channel-wise multiply.

The spatial dimension of the final output in SD block remains invariant $W \times H$. However, for spatial channel, the proposed mechanism extracts the global information of spatial channel dimension to judge the importance degree of each channel, which removes the $C * p$ relatively non-important channels, and retains the $C * (1 - p)$ relatively important channels.

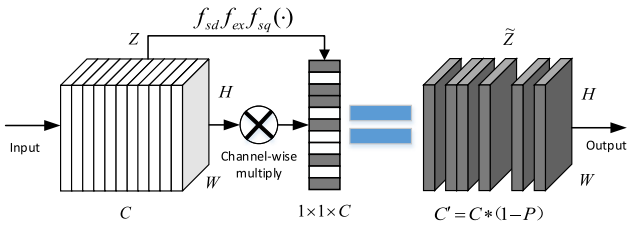


FIGURE 2. The final output of SD block. The black and white in $1 \times 1 \times C$ represent the value 1 and 0 respectively. Through ***Channel-wise multiply*** with input, the output channels corresponding to 0 are temporarily removed, and are a hidden in $\widetilde{z}$.

TABLE 1. Model architectures of VGG16 and ResNet50 with SD. Here VGG16 with SD is referred to as VGG16-SD, and ResNet50 with SD is ResNet50-SD. Both VGG16-SD and ResNet50-SD embody two different applications of SD in CNNs (as mentioned in Section III).

stage	output	VGG16	VGG16-SD	ResNet50	ResNet50-SD
conv1	112x112	$[3\times3, 64]\times2$		$7\times7, 64, s=2$	
		2x2 pool max,s=2			
conv2	56x56	$[3\times3, 128]\times2$		3x3 pool max, s=2	
		2x2 pool max, s=2		$\begin{bmatrix}1\times1,64\\3\times3,64\\1\times1,256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1,64\\3\times3,64\\1\times1,256\\SD,[16,256,128]\end{bmatrix}\times3$
conv3	28x28	$[3\times3, 256]\times2$		$\begin{bmatrix}1\times1,128\\3\times3,128\\1\times1,512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1,128\\3\times3,128\\1\times1,512\\SD,[32,512,256]\end{bmatrix}\times4$
		2x2 pool max, s=2			
conv4	14x14	$[3\times3, 512]\times2$		$\begin{bmatrix}1\times1,256\\3\times3,256\\1\times1,1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1,256\\3\times3,256\\1\times1,1024\\SD,[64,1024,512]\end{bmatrix}\times6$
		2x2 pool max, s=2			
conv5	7x7	$[3\times3, 512]\times2$		$\begin{bmatrix}1\times1,512\\3\times3,512\\1\times1,2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1,512\\3\times3,512\\1\times1,2048\\SD,[128,2048,1024]\end{bmatrix}\times3$
		2x2 pool max, s=2			
	1x1	FC 4096, ReLUs		7x7 global average pool, FC 1000, *softmax*	
		Dropout P=0.5	SD P=0.5		
		FC 4096, ReLUs			
		Dropout P=0.5	SD P=0.5		
		FC 1000, *softmax*			
#params		138.38×10^6	142.57×10^6	25.609×10^6	24.937×10^6
FLOPs		15.47×10^9	15.48×10^9	3.87×10^9	3.504×10^9

IV. MODEL AND COMPUTATIONAL COMPLEXITY

Previous studies have shown that dropout unit is generally placed behind the FC layer of model architecture to randomly drop out the neurons (such as VGGNet [3]). In contrast, the proposed SD can not only be placed behind FC layer, but between blocks of convolution layer to drop out the convolution feature channels. Table 1 respectively describes the VGG16 and ResNet50 module architecture of dropout and sequence-dropout with 224×224 pixel images as input.

In this section, the different model architecture embedding the SD block are first introduced in IV-A followed by the interpretation of the model complexity in IV-B, and then the computational cost of the networks are analyzed in IV-C. Finally, this section ends in the discussion.

A. MODEL ARCHITECTURE

As illustrated in Table 1, two dropout units exist in the VGG16 network, which are placed behind the 6th and 7th FC layer, respectively. Additionally, the number of neurons changes from 4096 to 2048 when using the dropout unit (the probability p of random elimination is set at 0.5). The third column in Table 1 shows that using the SD scheme in VGG16 network replaces the dropout units with SD blocks, and also shows that the input of SD block is FC layer neurons ($1\times1\times4096$) in VGG16-SD architecture, which means it is not required to aggregate the feature maps across spatial dimension in SD block. With elimination ratio 0.5, the neuron number changes from 4096 to 2048 by SD calculation, which is the same as dropout.

The ResNet50 network deepens the network through the residual module, which to a certain extent solves the problem that the deep network cannot be trained due to gradient disappearance or gradient explosion. The ResNet50 architecture is implemented by stacking residual modules together, and the input dimension (number of channels) of its stack block is expanded four times by 1×1 convolution kernel for output. Detailed description of residual module is provided in the row conv2 of column ResNet50 in Table 1 (green background) as follows. Firstly, a 1×1 convolution filter kernel is utilized, followed by a 3×3 ($c=64$) convolution operation. The dimension for output is then expanded with $1\times1(c=256)$ convolution. The concept of applying SD block to ResNet50 network is to imbed SD block in the original residual module to form a new stack block. In the column of ResNet50-SD, the numbers in the brackets after SD of each residual module respectively refer to the first FC layer output dimension, the second FC layer output dimension and the final SD block output dimension. Compared with the original ResNet50, the output dimension of stack block in ResNet50-SD network is reduced to half (elimination ratio is set at 0.5). The orange background area of Table 1 shows that the conv2 output dimension of ResNet50-SD is reduced from 256 to 128, the

TABLE 2. Comparison of MLP and CNN on MNIST dataset. ∼ indicates the model architecture of CNN.

Network	Architecture	Accuracy %
MLP	2 layers, FC+ReLUs	98.06
MLP+Dropout	3 layers, FC+ReLUs+Dropout ($p = 0.5$)	98.09
MLP+SD	3 layers, FC+ReLUs+SD ($p = 0.5, r = 16$)	**98.13**
CNN	5 layers, Conv+Pool+Conv+Pool+FC	99.20
CNN+Dropout	6 layers, ∼+Dropout ($p = 0.5$)	99.36
CNN+SD	6 layers, ∼+SD ($p = 0.5, r = 16$)	**99.47**

conv3 output dimension is reduced from 512 to 256, the conv4 output dimension is reduced from 1024 to 512, and the conv5 output dimension is reduced from 2048 to 1024.

B. MODEL COMPLEXITY

The complexity of the model is evaluated by calculating the number of parameters of the neural network. In the VGG16 network, the dropout unit only introduces probability p to remove the neurons without generating learning parameters in the calculation process. Therefore, in the VGG16-SD architecture, SD block is used to take the place of dropout module, and the additional network parameters are introduced on the SD block. The parameters of SD block are contained in two FC of gate mechanism [44], and the calculation of the number of parameters is given by the following formula:

$$\frac{2}{r}\sum_{l=1}^{L} N_l \cdot C_l^2 \tag{5}$$

where r denotes the reduction ratio, L refers to the number of stages (*e.g.* conv2, $\cdots$, conv5 in Table 1), N_l is the number of repetitions of SD block at the l stage, and C_l denotes the dimension of the output channels for l stage. The calculation in Eq. (5) shows that the number of parameters in the VGG16-SD network rises 3.028% (4.19M) compared to the VGG16 network. In the ResNet50-SD network, the parameters of the network increase with the addition of SD block, while the number of parameters introduced by the connection between stack blocks decreases. Thus, the final result calculated by Eq. (5) shows that the parameters of ResNet50-SD are reduced by 2.623% (0.672M) compared with ResNet50.

C. COMPUTATIONAL COMPLEXITY

A 224×224 pixel image is used as the input of network, and the computational cost of the network is evaluated by using floating-point operations per second (FLOPs). The VGG16 network requires ∼15.470 GFLOPs for a single forward propagation calculation. By comparison, the VGG16-SD network increases two FC layer floating-point operations of SD block, slightly increasing by 0.027% (0.00419 GFLOPs). This illustrates that the proposed SD block used in the VGG16 network is completely acceptable in terms of computational complexity. The original ResNet50 requires ∼3.87 GFLOPs for a single forward propagation calculation, while by imbedding SD block, ResNet50 adds floating-point operations of two FC layers and simultaneously changes the output dimension of each stack, ultimately reducing the overall GFLOPs of the network. By calculation, the ResNet50-SD network corresponds to the original ResNet50 network relatively decreasing by 10.22% (0.39566 GFLOPs). It can thus be seen that the computational cost of ResNet50 is significantly reduced by imbedding SD block. The training time of the network in a real environment is also affected by the bandwidth of GPU, code quality, and other factors.

Discussion

In practice, it is acceptable that the complexity and computational cost of the network increases slightly with SD block placed in VGG16. By adding a small amount of extra expenditure, the accuracy of the network is dramatically improved (see the experiment results in Section V). The SD block is placed in ResNet50 to form a new stack block for connection, changing the output dimension of each stack block, reducing the complexity and computational cost of the network, and improving accuracy.

V. EXPERIMENTS

In this section, the implementation of the proposed SD block is introduced and its effectiveness is demonstrated using benchmark datasets such as MNIST [45], CIFAR-10, CIFAR-100 [46], ImageNet 2012 [23], and Places365-Standard [47].

A. IMPLEMENTATION

The proposed SD block is implemented on the PyTorch framework. For data augmentation, 224×224 pixels are randomly cropped from training images. In addition, batch size is not fixed during training and the learning rate is diminished by segments for network training.

B. MNIST

The benchmark dataset MNIST is composed of handwritten digit images. All images are resized to 28×28 pixels. Among

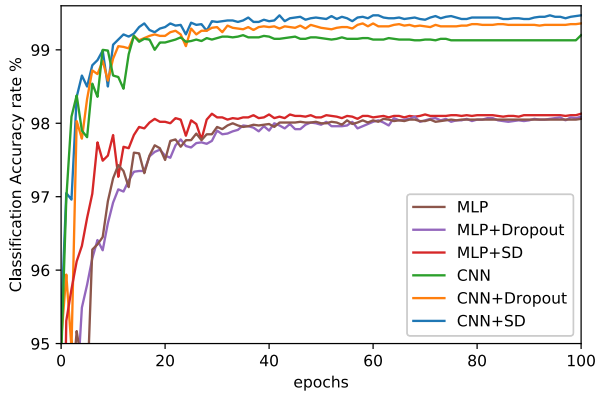


FIGURE 3. Test accuracy of different models on the MNIST dataset.

the resized images, 60,000 images are used as the training set, while others are used as the testing set. For fair comparison, the proposed SD block is embedded into multi-layer perceptron (MLP) and a simple convolution neural network (CNN). Additionally, the MLP is composed of a fully-connect layer and a ReLUs layer. The CNN contain two convolution layers followed by a pool layer and a fully connect layer. The models are trained by stochastic gradient descent (SGD) optimizer and the initial learning rate is set to 0.1 and decreased 10 times every 30 epochs.

As illustrated in Table 2, the standard dropout layer and the proposed SD block are employed in both MLP and CNN for comparison. The accuracy of MLP is 98.06%. After employing dropout layer, the accuracy is improved to 98.09%. In addition, by embedding the proposed SD block into MLP, the accuracy rises again by 0.04%. However, the accuracy of CNN+SD achieves the best performance among CNN, CNN+Dropout, and CNN+SD. Both experiments demonstrate the effectiveness of the proposed SD block. The accuracies of the six models are illustrated in Fig. 3. After adopting the SD block, the classification accuracy can be observed to reach the optimal value more quickly. In addition, the accuracy of the MLP+SD and CNN+SD changes more smoothly than other curves, indicating that the proposed SD block is effective.

C. CIFAR-10 AND CIFAR-100

Benchmark datasets CIFAR-10 and CIFAR-100 consist of 60,000 32 × 32 colour images in 10 and 100 classes respectively. For the former dataset, 6,000 images per class are divided in 5,000 training images and 1,000 testing images. Similarly, 600 images per class in CIFAR-100 are divided in 500 training images and 100 testing images. For comparison, the proposed SD block is employed in VGG16 and ResNet50 and both are trained by SGD optimization. The learning rate is initially set to 0.1, and decreased by a factor of 10 in epoch 90 and a factor of 100 in epoch 150.

As presented in Table 3, the proposed SD block assists the VGG16 and ResNet50 datasets to achieve the best performance. Specifically, after adding SD block, the accuracy of VGG16+SD rises to 89.04% and 57.23%, which is higher than the VG-G16+Dropout. However, the ResNet50+SD rises by 3.01% on CIFAR-10. The accuracy curves of the four models are provided in Fig. 4.

D. ImageNet2012

The ImageNet2012 dataset is composed of 1.2 million training images and 50,000 testing images. Both images are chosen from 1,000 classes of the ImageNet dataset. Following existing methods, the *Top-1* and *Top-5* accuracy rates are adopted for evaluation.

As illustrated in Table 4, four models are used for quantitative comparisons and contain the same architecture as the models in Table 3. Results demonstrate that the SD block is beneficial to model performance and efficiency. Specifically, after adopting the SD block, both VGG16 and ResNet50 achieve the best performance in terms of the *Top-1* and *Top-5* accuracy rates. Moreover, the model size and GFLOPs of VGG16+SD is slightly increased, while the model size and GFLOPs of ResNet50+SD is slightly decreased. The accuracy curves of the four models are provided in Fig. 5.

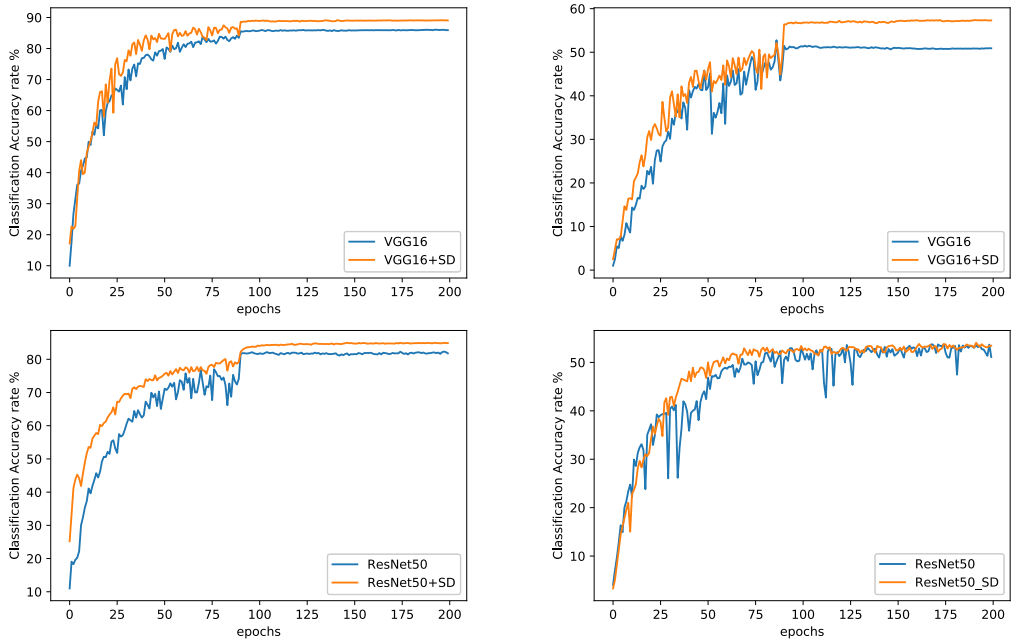
E. PLACES365-STANDARD

Places365-Standard is a benchmark dataset that consists of single object images. In this dataset, approximately 1.8 million images are split into the training set, and about 365,000 images are used for the validation set. The networks of VGG16, ResNet50, and ResNet152 are utilized to verify the effectiveness of the SD block and corresponding results are presented in Table 5.

As illustrated, VGG16+Dropout obtains the Top-1 accuracy of 55.24% and VGG16+SD reaches 56.31%. By

TABLE 3. Accuracy rates of different models on CIFAR-10 and CIFAR-100. The detail architectures of VGG16 and ResNet50 are provided in Table 1.

Network	Architecture	CIFAR-10 Accuracy %	CIFAR-100 Accuracy %
VGG16+Dropout	FC+Dropout	85.93	50.84
VGG16+SD	FC+SD ($p = 0.5, r = 16$)	**89.04**	**57.23**
ResNet50	residual block	81.82	52.6
ResNet50+SD	residual block+SD	**84.83**	**53.13**



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

FIGURE 4. Testing accuracy (%) of VGG16, ResNet50, VGG16+SD and ResNet50+SD on CIFAR-10 and CIFAR-100.

TABLE 4. Comparison of *Top-1* and *Top-5* accuracy rates (%) of VGG16, ResNet50, VGG16+SD and ResNet50+SD on ImageNet2012 dataset.

Network	Size	GFLOPs	*Top-1* Accuracy %	*Top-5* Accuracy %
VGG16+Dropout	531.5 MB	15.47	72.01	90.69
VGG16+SD	547.6 MB	15.48	**73.20**	**91.52**
ResNet50	98.4 MB	3.87	76.56	93.01
ResNet50+SD	95.8 MB	3.5	**78.02**	**93.82**

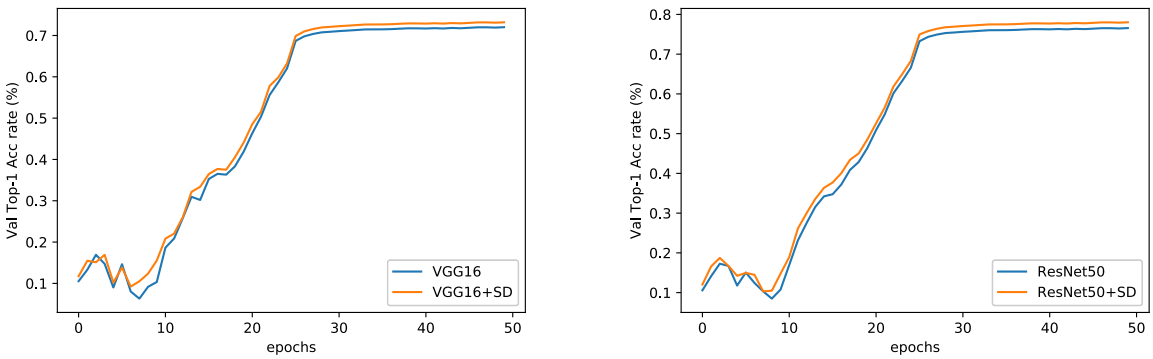


FIGURE 5. Curves of validation *Top-1* accuracy on ImageNet2012 dataset. (Left): VGG16 and VGG16+SD; (Right): ResNet50 and ResNet50+SD.

comparing the two metrics of VGG16+Dropout and VGG16+SD, it can be observed that the SD block can improve classification performance. ResNet50 obtain the Top-1 accuracy of 55.18%, ResNet152 achieves 54.74% and ResNet152+SD can reach 55.32%, which shows that the metrics of the ResNet152+SD are closer to ResNet50 than ResNet152. Comparing the last four models indicates that the SD block can help to mitigate the difficulty of training deeper networks to some extent.

F. COMPARING THE PERFORMANCE OF REDUCING OVERFITTING

In this section, we design two experiments to verify the performance of the proposed SD block on the overfitting

TABLE 5. Comparison of *Top-1* and *Top-5* accuracy rates (%) of VGG16, ResNet50, and ResNet152 and corresponding improved networks on Places365-Standard dataset.

Network	*Top-1* Accuracy %	*Top-5* Accuracy %
VGG16+Dropout	55.24	84.91
VGG16+SD	**56.31**	**85.82**
ResNet50	55.18	85.29
ResNet50+SD	**55.78**	**86.53**
ResNet152	54.74	85.08
ResNet152+SD	**55.32**	**86.33**

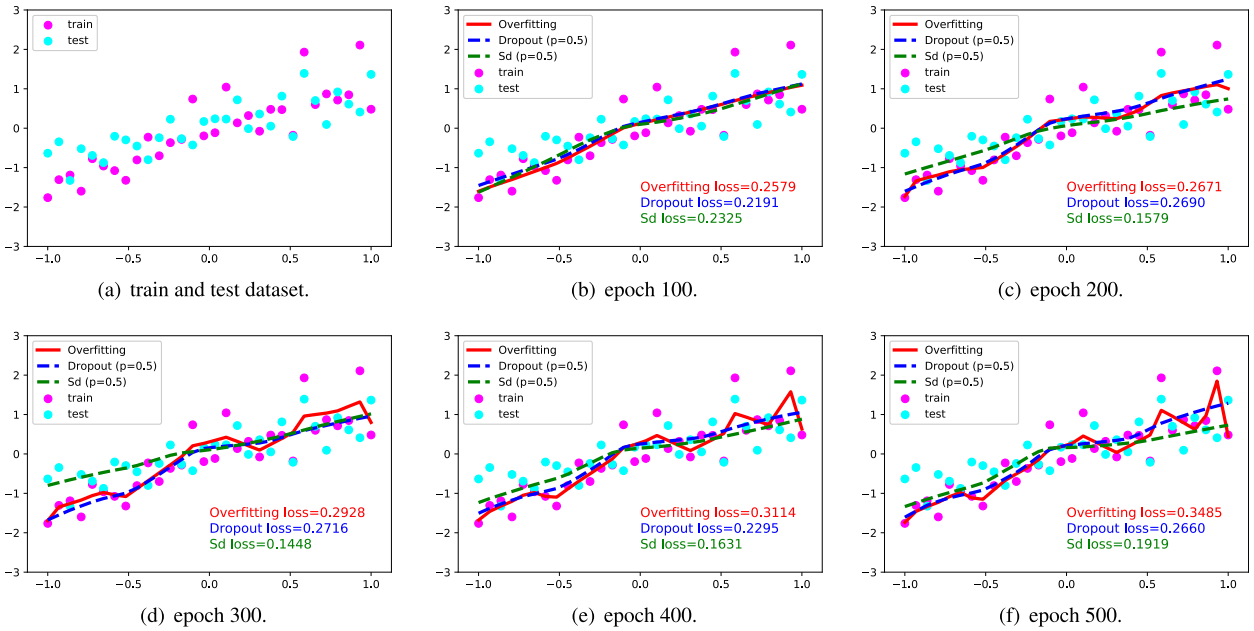
problem. In the first experiment, other three commonly used regularization methods such as L2 [9], BN [18], Dropout [17] are introduced to compare. We construct the same network architecture (28 × 28-32 × 24 × 24-64 × 12 × 12-1024-1024-2048-10) with ReLUs and train it with the four regularization methods (including SD) and the SGD optimizer by 100 epochs on the MNIST dataset. The initial learning rate is set to 0.1 and decreases to 0.05 in epoch 15, 0.01 in epoch 31, 0.005 in epoch 61 and 0.001 in epoch 71. Table 6 shows the validation results of different regularization methods. The results demonstrate that the proposed SD can achieve a lower generalization error than other three regularization methods (0.64% vs. 0.71%, 0.69% and 0.66%), and the combination of the SD with L2 and BN reaches the lowest error (0.56%).

Fewer data can highlight the problem of overfitting, so in the second experiment, we randomly sample 30 points

TABLE 6. The results of comparison for different regularization methods on MNIST dataset. "None" means the network is trained without using any regularization methods.

Method	*Test Classification* Error %
None	0.83
L2	0.71
BN	0.69
Dropout	0.66
Dropout+L2	0.65
Dropout+BN	0.63
Dropout+L2+BN	0.58
SD	**0.64**
SD+L2	0.6
SD+BN	0.59
SD+L2+BN	**0.56**

data in the interval (−1,1) of linear space as another dataset. The same model architecture (30-300-300-1) with ReLUs is trained applying the *Adaptive moment estimation* (Adam) optimizer with Dropout and the proposed SD. The learning rate is fixed with 0.01 and training epoch is set to 500. Fig. 6 shows the test result after every 100 epochs of training. Fig. 6(f) is the finial result which demonstrates that the proposed SD block achieves the lowest test loss (0.1919 vs. 0.2660 and 0.3485), and the generalization performance of the green line (SD) is better than the blue line (Dropout) and the red line (overfitting).



(a) train and test dataset. (b) epoch 100. (c) epoch 200.

(d) epoch 300. (e) epoch 400. (f) epoch 500.

FIGURE 6. The fitting situation of dropout and SD on dataset during the training process. Fig. 6(a) shows the training and testing datasets. In Fig. 6(b), the red line called as "Overfitting" represents that the network is overfitting without using any regularization methods. It is not easy to distinguish the performance among SD, Dropout and overfitting at epoch 100. From Fig. 6(c) to Fig. 6(f), the red line gradually fits the train data perfectly, but it is not well done in test data. These four pictures shows the generalization performance of SD (green line) is better than Dropout (blue line).

VI. CONCLUSION

In this paper, the novel SD technique is proposed for solving the problem of overfitting. The proposed SD block maintains the most important channels by dropping out features according to the global information of each channel. The method is composed of three steps: channel compression calculation, channel weight calculation, and sorting and dropping calculation. The proposed SD is further embedded into state-of-the-art backbone CNN models such as VGGNet16, ResNet50, and ResNet152 for image classification tasks. Extensive experiments on five challenging image classification datasets demonstrate that the proposed SD block is effective and widely applicable.

In the future work, we will try to extend the proposed SD block, and apply it into high-level computer vision task such as tacking [48]. We will also focus on theoretical analysis. We believe it will contribute to further improvement and application of the algorithm.

REFERENCES

[1] X. Zhang, D. Wang, Z. Zhou, and Y. Ma, "Robust low-rank tensor recovery with rectification and alignment," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Jul. 16, 2019, doi: 10.1109/TPAMI.2019.2929043.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556

[4] X. Zhang, W. Hu, N. Xie, H. Bao, and S. Maybank, "A robust tracking system for low frame rate video," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 279–304, Dec. 2015.

[5] X. Zhang, W. Li, X. Ye, and S. Maybank, "Robust hand tracking via novel multi-cue integration," *Neurocomputing*, vol. 157, pp. 296–305, 2015.

[6] D. A. van Dyk and X.-L. Meng, "The art of data augmentation," *J. Comput. Graph. Statist.*, vol. 10, no. 1, pp. 1–50, Mar. 2001.

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

[8] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Stat. Soci. B (Methodol.)*, vol. 58, no. 1, pp. 267–288, Jan. 1996.

[9] A. Y. Ng, "Feature selection, L_1 vs. L_2 regularization, and rotational invariance," in *Proc. 21st Int. Conf. Mach. Learn.*, 2004, p. 78.

[10] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *J. Roy. Stat. Soc., B (Stat. Methodol.)*, vol. 67, no. 2, pp. 301–320, Apr. 2005.

[11] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Proc. Adv. Neural Inf. Process. Syst.*, 1992, pp. 950–957.

[12] X. Zhang, Q. Liu, D. Wang, L. Zhao, N. Gu, and S. Maybank, "Self-taught semisupervised dictionary learning with nonnegative constraint," *IEEE Trans. Ind. Informat.*, vol. 16, no. 1, pp. 532–543, Jan. 2020.

[13] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.

[14] X. Zhang, J. Zheng, D. Wang, and L. Zhao, "Exemplar-based denoising: A unified low-rank recovery framework," *IEEE Trans. Circuits Syst. Video Technol.*, early access, Jul. 9, 2019, doi: 10.1109/TCSVT.2019.2927603.

[15] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, no. 1, p. 60, Dec. 2019.

[16] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[17] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," 2012, *arXiv:1207.0580*. [Online]. Available: http://arxiv.org/abs/1207.0580

[18] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, *arXiv:1502.03167*. [Online]. Available: http://arxiv.org/abs/1502.03167

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[20] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7132–7141.

[21] M. Lin, Q. Chen, and S. Yan, "Network in network," 2013, *arXiv:1312.4400*. [Online]. Available: http://arxiv.org/abs/1312.4400

[22] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng, "Dual path networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4467–4475.

[23] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[24] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.*, pp. 740–755, 2014.

[25] S. Shao, Z. Li, T. Zhang, C. Peng, G. Yu, X. Zhang, J. Li, and J. Sun, "Objects365: A large-scale, high-quality dataset for object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 8430–8439.

[26] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," 2017, *arXiv:1708.04896*. [Online]. Available: http://arxiv.org/abs/1708.04896

[27] F. J. Moreno-Barea, F. Strazzera, J. M. Jerez, D. Urda, and L. Franco, "Forward noise adjustment scheme for data augmentation," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Nov. 2018, pp. 728–734.

[28] G. Kang, X. Dong, L. Zheng, and Y. Yang, "PatchShuffle regularization," 2017, *arXiv:1707.07103*. [Online]. Available: http://arxiv.org/abs/1707.07103

[29] H. Inoue, "Data augmentation by pairing samples for images classification," 2018, *arXiv:1801.02929*. [Online]. Available: http://arxiv.org/abs/1801.02929

[30] Y. Lecun, 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.

[31] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 807–814.

[32] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," 2013, *arXiv:1302.4389*. [Online]. Available: http://arxiv.org/abs/1302.4389

[33] M. D. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," 2013, *arXiv:1301.3557*. [Online]. Available: http://arxiv.org/abs/1301.3557

[34] E. Hoffer, I. Hubara, and D. Soudry, "Train longer, generalize better: Closing the generalization gap in large batch training of neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1731–1741.

[35] M. Sajjadi, M. Javanmardi, and T. Tasdizen, "Regularization with stochastic transformations and perturbations for deep semi-supervised learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1163–1171.

[36] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko, "Semi-supervised learning with ladder networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 3546–3554.

[37] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, Aug. 1996.

[38] H. Grabner and H. Bischof, "On-line boosting and vision," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 1, Jun. 2006, pp. 260–267.

[39] A. K. Seewald, "How to make stacking better and faster while also taking care of an unknown weakness," in *Proc. 9th Int. Conf. Mach. Learn.*, 2002, pp. 554–561.

[40] X. Bouthillier, K. Konda, P. Vincent, and R. Memisevic, "Dropout as data augmentation," 2015, *arXiv:1506.08700*. [Online]. Available: http://arxiv.org/abs/1506.08700

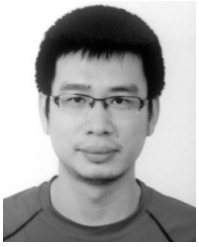
[41] P. Morerio, J. Cavazza, R. Volpi, R. Vidal, and V. Murino, "Curriculum dropout," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 3544–3552.

[42] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1058–1066.

[43] A. Karpathy, J. Johnson, and L. Fei-Fei, "Visualizing and understanding recurrent networks," 2015, *arXiv:1506.02078*. [Online]. Available: http://arxiv.org/abs/1506.02078
[44] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[45] L. Deng, "The MNIST database of handwritten digit images for machine learning research [Best of the Web]," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, Nov. 2012.
[46] A. Krizhevsky, V. Nair, and G. Hinton. (Jun. 2009). *Cifar-10 and Cifar-100 Datasets*. [Online]. Available: https://www.cs.toronto.edu/kriz/cifar.html
[47] B. Zhou, A. Khosla, A. Lapedriza, A. Torralba, and A. Oliva, "Places: An image database for deep scene understanding," 2016, *arXiv:1610.02055*. [Online]. Available: http://arxiv.org/abs/1610.02055
[48] X. Zhang, W. Hu, S. Chen, and S. Maybank, "Graph-Embedding-Based learning for robust object tracking," *IEEE Trans. Ind. Electron.*, vol. 61, no. 2, pp. 1072–1084, Feb. 2014.
[49] X. Zhang, X. Shi, W. Hu, X. Li, and S. Maybank, "Visual tracking via dynamic tensor analysis with mean update," *Neurocomputing*, vol. 74, no. 17, pp. 3277–3285, 2011.
[50] X. Zhang, W. Hu, W. Qu, and S. Maybank, "Multiple object tracking via species-based particle swarm optimization," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 11, pp. 1590–1602, 2010.

LEDAN QIAN was born in Yueqing, Zhejiang, China, in 1979. He received the B.E. degree in computer science and technology from the Zhejiang University of Technology, Hangzhou, China, in 2003, and the M.E. degree in control engineering from the Wuhan University of Technology, Wuhan, China, in 2008.

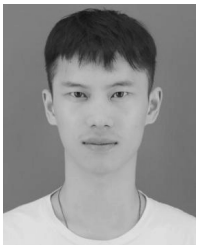
In 2015, he joined the City Key Laboratory, Intelligent Computing and Application of Visual Big Data. He is currently a Lecturer with the College of Mathematics and Physics, Wenzhou University, China, where he has strong teaching emphasis on computer networks. He has participated in a number of projects funded by the Zhejiang Education Department Fund and the Wenzhou Science Technology Fund. He has authored three research articles on international journal and seven articles in Chinese. His research interests include pattern recognition, deep learning, computer networks, and VR/AR.



LIBING HU was born in Wenzhou, Zhejiang, China, in 1979. He received the Bachelor of Management degree from Zhejiang University, in 2006, and the master's degree in software engineering from the Huazhong University of Science and Technology, in 2009. His research interests include big data analysis, image processing, and neural networks.

LI ZHAO received the B.Sc. degree in automation and the M.Eng. degree in control theory and control engineering from Central South University, China, in 2005 and 2008, respectively. She is currently an Assistant Researcher with Wenzhou University. Her research interests include pattern recognition, computer vision, and machine learning.

TAO WANG received the B.Sc. degree in information and computing science from Hainan Normal University, China, in 2018. He is currently pursuing the degree with the College of Computer Science and Artificial Intelligence, Wenzhou University, China. His research interests include several topics in computer vision and machine learning, such as object tracking, image/video quality restoration, adversarial learning, image-to-image translation, and reinforcement learning.



RUNHUA JIANG received the bachelor's degree from the Department of Information Science, Tianjin University of Finance and Economy, China. He is currently pursuing the degree in computer software and theory with the College of Computer Science and Artificial Intelligence, Wenzhou University, China. His research interests include image and video processing, pattern recognition, and machine learning.