Demystifying Dropout

Hongchang Gao [1 2] **Jian Pei** [3 4] **Heng Huang** [1 2]

Abstract

Dropout is a popular technique to train large-scale deep neural networks to alleviate the overfitting problem. To disclose the underlying reason for its gain, numerous works have tried to explain it from different perspectives. In this paper, unlike existing works, we explore it from a new perspective to provide new insight into this line of research. In detail, we disentangle the forward and backward pass of dropout. Then, we find that these two passes need different levels of noise to improve the generalization performance of deep neural networks. Based on this observation, we propose the augmented dropout, which employs different dropping strategies in the forward and backward pass, to improve the standard dropout. Experimental results have verified the effectiveness of our proposed method.

1. Introduction

In recent years, deep neural networks have demonstrated superior performance on a wide range of tasks, such as image classification (Krizhevsky et al., 2012; Simonyan & Zisserman, 2014; He et al., 2016), object detection (Girshick et al., 2014; Ren et al., 2015), image generation (Goodfellow et al., 2014; Brock et al., 2018; Miyato et al., 2018; Gao & Huang, 2018), and machine translation (Vaswani et al., 2017). However, deep neural networks face with the overfitting issue since they usually have an extremely large number of parameters. Then, dropout (Hinton et al., 2012) is proposed to alleviate this problem. Specifically, dropout randomly sets some neurons of the network to zero during the training phase to prevent feature co-adaption. This simple idea has become a common technique to alleviate overfitting issues and lift up the generalization performance of deep neural networks in various tasks.

[1]JD Finance America Corporation [2]Department of Electrical and Computer Engineering, University of Pittsburgh, Pittsburgh, USA [3]JD.com [4]School of Computing Science, Simon Fraser University, Canada. Correspondence to: Heng Huang <heng.huang@pitt.edu>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

Inspired by the good performance of dropout, a number of variants have been proposed, promoting the generalization performance of existing network architectures, such as DropConnect (Wan et al., 2013), maxout(Goodfellow et al., 2013), DropPath (Larsson et al., 2016), DropBlock (Ghiasi et al., 2018). Besides, to disclose the success of dropout, various works (Hinton et al., 2012; Wager et al., 2013; Bouthillier et al., 2015; Helmbold & Long, 2017; Jain et al., 2015) have attempted to analyze it from different perspectives. In the seminal paper (Hinton et al., 2012), dropout is described as an ensemble technique which ensembles many sub-networks with sharing parameters. (Wager et al., 2013) treats dropout as an adaptive ℓ_2 regularization and establishes the connection between dropout and the adaptive optimization algorithm AdaGrad (Duchi et al., 2011). While, in (Helmbold & Long, 2017), the difference between weight decay regularization and dropout is disclosed. Specifically, unlike the traditional weight decay regularization, dropout is insensitive to the rescaling of input features and the outputs. (Gal & Ghahramani, 2016) discusses dropout from the Bayesian perspective, treating it as an approximation to the deep Gaussian process. On the other hand, some works attempt to theoretically study the generalization performance for the deep neural network with dropout. For instance, (Gao & Zhou, 2016) shows that dropout can help to reduce the Rademacher complexity of deep neural networks. In particular, dropout pushes the polynomial reduction for shallow neural networks and the exponential reduction for deep neural networks. Later, Mou (Mou et al., 2018) studies the generalization performance of dropout under the distribution-free setting.

By examining the works above which try to explain dropout, we find that most of them focus on the forward pass to attribute the gain of dropout to the model combination or feature augmentation. For instance, in the original paper (Hinton et al., 2012), dropout is described to ensemble a number of sub-networks which are trained on different subsets of features (hidden units). As a result, the authors argue that dropout can prevent the network from relying on single features. Additionally, (Bouthillier et al., 2015) views dropout as a kind of data augmentation. However, as we know, deep neural networks include the forward pass and the backward pass and they share the same neurons. When zeroing out some neurons of the deep neural network, the backward pass will also be affected. As a result, the gradient will be disturbed by dropout either when using backpropagation to optimize deep neural networks. As discussed earlier, most of existing works ignore the effect of dropout on the backward pass. Thus, it is necessary to explore how dropout affects the backward pass. On the other hand, the generalization performance of deep neural networks is also affected by optimization algorithms. For instance, (Keskar et al., 2016) shows that a large batch size for SGD can lead to degradation in generalization performance. One argument for this phenomenon is that the noise introduced by SGD of a small batch size can help to escape from sharp minima. Besides, a number of works (Ge et al., 2015; Jin et al., 2017; Zhang et al., 2017; Daneshmand et al., 2018) have theoretically justified that noise does be helpful for SGD to escape from saddle points. Furthermore, (Neelakantan et al., 2015) shows that adding noise to the gradient can improve the performance of deep neural networks practically.

Based on the above discussion, a natural question follows: Which pass accounts for the gain from dropout? Is it due to the feature augmentation in the forward pass or the noisy gradient in the backward pass? To answer this question, in this paper, we separate the forward pass and backward pass to disclose the underlying reason for dropout's success. Specifically, we propose the *forward dropout* which conducts dropout only on the forward pass and the *backward dropout* which performs dropout only on the backward pass. In this way, the forward dropout will account for the feature augmentation while the backward dropout will account for the noisy gradient. Our empirical result shows that both the feature augmentation in the forward pass and the noisy gradient in the backward pass are necessary to the success of dropout. Either one of them cannot beat the standard dropout. However, we also find that the dropout ratio has a different effect on these two passes. Specifically, the forward and backward dropout achieve their best performance with different dropout ratios. Based on these observations, we propose a novel *augmented dropout*, which employs different dropout ratios for the forward and backward pass. In this way, the augmented dropout can fully utilize the benefit of these two passes. At last, extensive experimental results have verified the effectiveness of our proposed methods.

2. Related Works

In this section, we will review related works about dropout to give the motivation for this paper.

Dropout is first proposed in the seminal work (Hinton et al., 2012) to alleviate the overfitting problem of deep neural networks. The basic idea is to randomly set some neurons of the neural network to zero in the training phase. With this simple yet effective technique, the performance of numerous deep neural networks is improved to new state-of-the-art levels. Recently, many variants have been proposed. For instance, DropConnect (Wan et al., 2013) proposes to randomly drop the connection between the neurons in two consecutive layers. DropPath (Larsson et al., 2016) randomly drops layers of the neural network. SpatialDropout (Tompson et al., 2015) zeros out channels from the feature map. DropBlock (Ghiasi et al., 2018) drops continuous regions of the feature map. All these variants have achieved some improvement in practical applications.

The good performance of dropout has attracted researchers to analyze its underlying reasons. In the original paper (Hinton et al., 2012), the authors attribute the gain from dropout to the ensemble of many sub-networks. In particular, the author argues that dropping neurons during training is actually to train an exponential number of “thinned” networks and then average the results of these thinned networks in the testing phase. (Bouthillier et al., 2015) interprets dropout as a kind of data augmentation approach which augments the training data to cover more regions of the true data distribution. In this way, the overfitting problem can be alleviated by the augmented data. (Wager et al., 2013) demonstrates that dropout is a kind of adaptive regularization approach for generalized linear models (GLMs). With this regularization, model parameters will be prevented to overfit data. In (Helmbold & Long, 2017), dropout is also interpreted as a regularization method to avoid overfitting. Obviously, these explanations for dropout, including model combination, data augmentation, and regularization, only consider the effect of dropout on the forward pass, ignoring how it affects the optimization procedure in the backward pass.

As we know, deep neural networks include the forward pass and the backward pass and they share the same neurons. When dropping some neurons of deep neural networks, the gradient from the backpropagation will also be disturbed. In other words, dropout leads to the noisy gradient. In recent years, numerous works (Ge et al., 2015; Jin et al., 2017; Zhang et al., 2017; Daneshmand et al., 2018) have shown that noisy SGD can help to escape from saddle points. For instance, (Ge et al., 2015) proves adding noise to SGD can escape strictly saddle points while (Jin et al., 2017) justifies that injecting noise to model parameters, leading to the noisy gradient, can also escape from strictly saddle points. In practice, (Neelakantan et al., 2015) demonstrates that injecting noise to the stochastic gradient can improve the performance of deep neural networks. Furthermore, SmoothOut (Wen et al., 2018), which injects noise to model parameters of deep neural networks can improve the generalization performance either. Thus, to fully understand the mechanism of dropout, it is necessary and important to explore how the noisy gradient introduced by dropout affect the generalization performance of deep neural networks.

3. Preliminary Knowledge

In this section, we will present some preliminary knowledge about dropout.

Throughout this paper, we use $H = \{h^{(l)}\}_{l=0}^{L-1}$ to denote the deep neural network with L layers where $h^{(l)}$ denotes the l-th hidden layer. In particular, $h^{(0)}$ denotes the input layer while $h^{(L-1)}$ denotes the output layer. In addition, $W = \{W_l\}_{l=0}^{L-1}$ denotes model parameters of the deep neural network. Specifically, W_l denotes the model parameter in the l-th layer. Furthermore, $\{\epsilon^{(l)}\}$ denotes the random variable of dropout. In detail, the entries of $\epsilon_i^{(l)}$ are independently drawn from the following distribution:

$$\begin{cases} P(\epsilon_i^{(l)} = 0) = p^{(l)} \\ P(\epsilon_i^{(l)} = \frac{1}{1-p^{(l)}}) = 1 - p^{(l)} \end{cases} , \tag{1}$$

where $p^{(l)} \in (0, 1)$ denotes the dropout ratio in the l-th layer. Based on these terminologies, the deep neural network with dropout is represented as follows:

$$\begin{aligned} z^{(l+1)} &= g_l(W_l; h^{(l)} \odot \epsilon^{(l)}) , \\ h^{(l+1)} &= f_l(z^{(l+1)}) . \end{aligned} \tag{2}$$

Here, $\odot$ denotes the element-wise multiplication, g_l represents the transformation in the l-th layer. In particular, for the fully connected layer, g_l denotes the linear transformation. For the convolutional layer, g_l denotes the convolution transformation. $z^{(l+1)}$ denotes the value of the hidden layer before conducting the non-linear activation. $f_l(\cdot)$ denotes the activation function.

Given a dataset $\{(x_i, y_i)\}_{i=1}^n$, the goal of dropout training is to optimize the following objective function:

$$\min_{W,b} J(W, b) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(W, b; x_i, y_i) , \tag{3}$$

where $\mathcal{L}$ denotes the loss function to penalize the wrong prediction. This objective function can be optimized by stochastic gradient descent (SGD) method, where both data samples x_i and hidden features $h^{(l)}$ are sampled randomly.

4. Demystify Dropout

Unlike existing works which focus on the forward pass, such as model combination and data augmentation, ignoring how dropout affects the backward pass, in this section, we will disclose how dropout affects both the forward and backward pass of deep neural networks respectively.

In this section, we assume the transformation g_l in each layer is a linear operation. Then, each layer is defined as follows:

$$\begin{aligned} z^{(l+1)} &= W_l(h^{(l)} \odot \epsilon^{(l)}) + b^{(l)} , \\ h^{(l+1)} &= f_l(z^{(l+1)}) . \end{aligned} \tag{4}$$

4.1. Back Propagation with Dropout

To employ SGD to train deep neural networks, at first, we need to compute the gradient in the backward pass. Here, we only focus on the gradient of hidden layers since that of the output layer is easy to obtain.

For the fully connected layer defined in Eq. (4), the gradient with respect to model parameters in the l-th layer is shown as follows:

$$\frac{\partial J}{\partial W^{(l)}} = \delta^{(l+1)}(h^{(l)} \odot \epsilon^{(l)})^T , \tag{5}$$

where

$$\delta^{(l)} \triangleq \frac{\partial J}{\partial z^{(l)}} = ((W^{(l)T}\delta^{(l+1)}) \odot f_l'(z^{(l)})) \odot \epsilon^{(l)} . \tag{6}$$

Note that we ignore the gradient of the bias in that it is easy to get. From Eq. (5) and Eq. (6), we can find that dropout happens not only in the forward pass but also in the backward pass. Specifically, in the backward pass, dropout disturbs the gradient by dropping features as Eq. (5) and the backpropagated error as Eq. (6). As a result, some entries of the gradient will be zeroed out, leading to the noisy gradient. Existing literature shows that noisy gradient is helpful to escape saddle points to converge to the solution which has good generalization performance. Hence, a natural question follows: Does the gain of dropout come from the forward pass, such as model combination and data augmentation, or the noisy gradient in the backward pass? In the following subsection, we will propose a novel method to investigate it.

4.2. Forward and Backward Dropout

To disclose the effect of dropout on the forward pass and the backward pass, we disentangle dropout: proposing the forward dropout and backward dropout as follows.

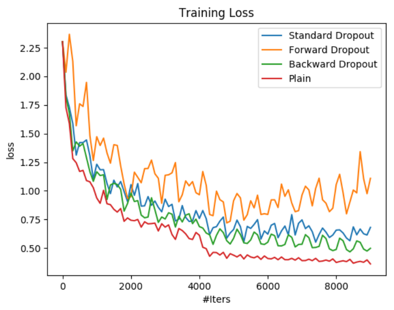
Definition 1. *(Forward Dropout) As the standard dropout, forward dropout randomly drops features in the forward pass but it does not drop features and backpropagated errors in the backward pass.*

Definition 2. *(Backward Dropout) Backward dropout keeps all features in the forward pass while drops features and back propagated errors as the standard dropout in the backward pass.*

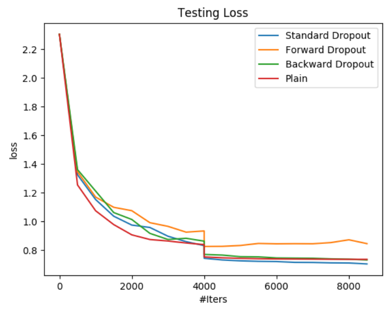
From the definition, we can know that the forward dropout only disturbs the forward pass while the backward dropout only disturbs the backward pass. Therefore, with them, we can investigate the effect of dropout on the forward pass by the forward dropout and that on the backward pass by the backward dropout. To this end, we conduct experiments on two networks, including ConvNet-Quick and ResNet-20, for CIFAR10 to demonstrate the effect of these two dropout methods.

Table 1. The test accuracy of ConvNet-Quick for CIFAR10

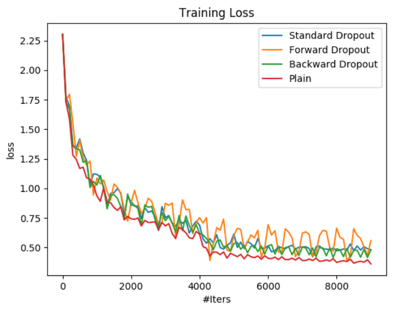
Dropout Ratio	0.3	0.2	0.1	0.05	0.01	0.005
Plain	0.7579					
Standard Dropout	0.7523	0.7617	0.7657	0.7647	0.7626	0.7608
Forward Dropout	0.6908	0.7211	0.7482	0.7627	0.7612	0.7578
Backward Dropout	0.7433	0.7557	0.7585	0.7593	0.7583	0.7599



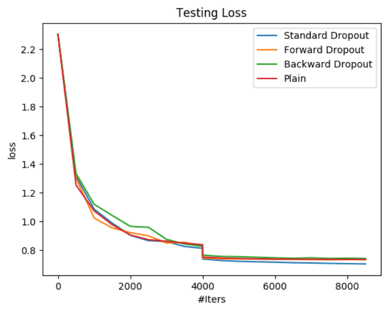
(a) p=0.2



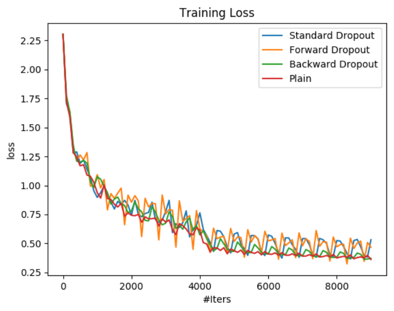
(b) p=0.2



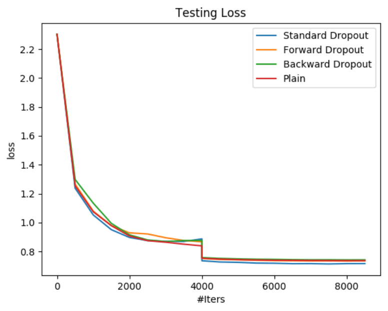
(c) p=0.05



(d) p=0.05



(e) p=0.005



(f) p=0.005

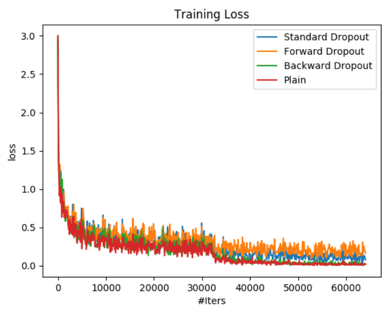
Figure 1. The training and testing loss of ConvNet-Quick for CIFAR10

Table 2. The test accuracy of ResNet-20 for CIFAR10

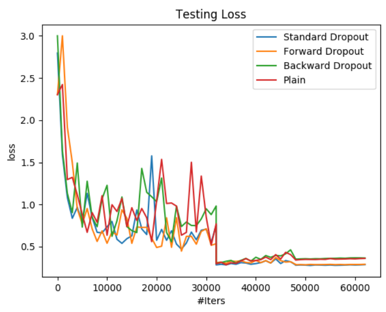
Dropout Ratio	0.3	0.2	0.1	0.05	0.01	0.005
Plain	0.9143					
Standard Dropout	0.9163	0.9176	0.9193	0.9174	0.9141	0.9154
Forward Dropout	0.9007	0.9093	0.9169	0.9171	0.9141	0.9142
Backward Dropout	0.9109	0.9130	0.9140	0.9142	0.9147	0.9146

ConvNet-Quick ConvNet is the network that employed in the original dropout paper (Hinton et al., 2012). It has three convolutional layers and two fully connected layers. We use ConvNet-Quick that is the implementation of ConvNet provided by the official Caffe toolbox (Jia et al., 2014). In this experiment, we conduct dropout on the two fully connected layers. The batch size is set to 100. We employ SGD with the momentum of 0.9 as the optimizer. The training procedure starts with a learning rate of 0.001 which is divided by 10 at 4,000 iterations. We terminate the training procedure at 9,000 iterations. Here, we did not perform any data augmentation. The only preprocessing for input images is to subtract the mean value.

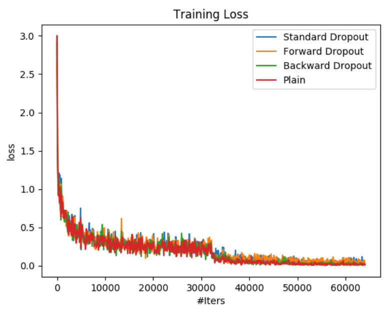
ResNet-20 ResNet-20 (He et al., 2016) is a widely used state-of-the-art method for CIFAR10 dataset. There are three building groups, each of which consists of a few convolution blocks Conv-BN-ReLU. The original ResNet-20 has no dropout. Here, following Wide-ResNet (Zagoruyko & Komodakis, 2016), we insert dropout between two convo-



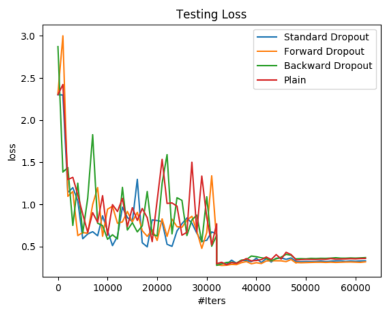
(a) p=0.2



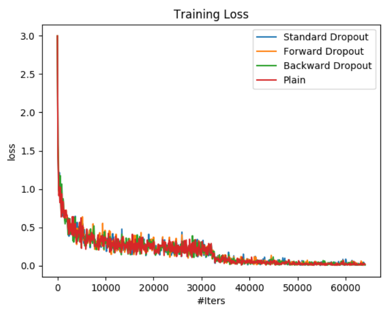
(b) p=0.2



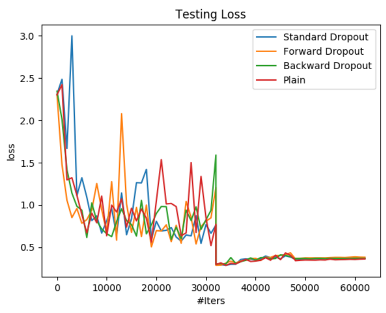
(c) p=0.05



(d) p=0.05



(e) p=0.005



(f) p=0.005

Figure 2. The training and testing loss of ResNet-20 for CIFAR10

lution blocks as Conv-BN-ReLU-Dropout-Conv-BN-ReLU. In our experiment, the batch size is set to 128. The number of iterations is set to 640,000. In addition, we employ the Nesterov with the momentum of 0.9 as the optimizer. The initial learning rate is set to 0.1 and multiplied by 0.1 at 320,000 and 480,000 iterations. Furthermore, a simple data augmentation is used in this experiment. Specifically, each image is padded by 4 pixels on each side. Then, a 32×32 patch is randomly cropped from the augmented image or its corresponding horizontal flip. At last, the weight decay parameter is set to 0.001.

In Table 1 and Table 2, we report the classification accuracy which is the mean of four runs. Here, the network without dropout is denoted by Plain. In addition, to comprehensively evaluate the performance of dropout, we employ different dropout ratios. From these two tables, we have the following observations:

- Comparing with the plain network, the standard dropout does improve the performance of deep neural networks for almost all cases, which verifies the effectiveness of dropout.
- As for the forward dropout, its performance is heavily affected by the dropout ratio. When the dropout ratio is large, the performance of the forward dropout is degraded significantly.
- Comparing with the forward dropout, the backward one is less affected by the dropout ratio. When decreasing the dropout ratio, the generalization performance of the deep neural network is improved gradually.
- When the dropout ratio is large, such as $p = 0.3$, both the forward and backward dropout degrade the performance of the plain deep neural networks. When decreasing the dropout ratio to a moderately small value, such as $p = 0.05$, both of them show some improvement. When the dropout ratio is very small, such as $p = 0.005$, the forward dropout has very few effects on the performance of deep neural networks, but the backward dropout can improve the plain neural network.
- Both the forward and backward dropout are inferior to the standard dropout for almost all cases.

To further disclose the underlying reason for these observations, we demonstrate the loss function value at different iterations in Figure 1 and Figure 2. Combined with these figures, we have the following conclusions:

- When the dropout ratio is large ($p = 0.2$), the training loss of the forward dropout are much larger than those of other methods. In other words, when p is large, due to interrupting features heavily, the forward dropout can increase the model bias and cause underfitting. As a result, it degrades the performance of the plain network. As for the standard dropout, although it employs the same dropout ratio with the forward dropout, yet its loss and accuracy are better than those of the forward dropout. The possible reason is the implicit regularization of the noisy gradient in the backward pass, which is helpful to escape local minima. When it comes to the backward dropout, although it can arrive at a smaller training loss compared with the standard dropout, it cannot outperform the standard one. The possible reason is that there is no data augmentation in the forward pass as the standard dropout. Hence, its generalization performance is worse than the standard dropout.

- When the dropout ratio becomes moderately small ($p = 0.05$), the training loss of the forward and backward dropout approach to that of the standard dropout, and their accuracy is a little better than that of the plain network. Thus, the data augmentation in the forward and the noisy gradient in the backward pass caused by the mild noise are helpful to improve the generalization performance. Additionally, the forward dropout outperforms the backward one. In other words, mild noise is more helpful in the forward pass.
- When the dropout ratio is very small ($p = 0.005$), it is intuitive that the forward dropout has little effect on the performance of the plain deep neural networks. Surprisingly, the backward dropout has better performance than the plain network, which means that the noisy gradient caused by the small noise in the backward pass contributes to improving the generalization performance.

In summary, the moderately small noise in the forward pass can augment the data to improve the generalization performance. In other words, to augment data solely, we should use moderately small noise. As for the noise in the backward pass, it can regularize the bias introduced from the forward pass. In addition, the very small noise in the backward pass is helpful to the optimization of deep neural networks, which consequently improves the generalization performance.

5. Augmented Dropout

In this section, we will propose the augmented dropout based on the observations in the last section.

The observations in the last section can be summarized as follows:

- Both the forward and backward dropout are inferior to the standard dropout when they act on the plain network solely.
- The forward and backward dropout achieve their best performance with different dropout ratio.

Based on these observations, we propose the *augmented dropout*. In detail, due to the first observation, we keep both the forward and backward dropout in our augmented dropout. Inspired by the second observation, we conduct dropout on the forward pass and backward pass separately with their own dropout ratio. Specifically, in the forward pass, we have the following procedure:

$$\begin{aligned} \hat{h}^{(l)}_{forward} &= h^{(l)} \odot \epsilon^{(l)}_{foward} , \\ z^{(l+1)} &= W_l \hat{h}^{(l)}_{forward} + b^{(l)} , \\ h^{(l+1)} &= f_l(z^{(l+1)}) , \end{aligned} \tag{7}$$

where $\epsilon^{(l)}_{foward}$ is drawn from the following distribution in terms of the forward dropout ratio $p^{(l)}_{forward}$:

$$\begin{cases} P(\epsilon^{(l)}_i = 0) = p^{(l)}_{forward} \\ P(\epsilon^{(l)}_i = \frac{1}{1-p^{(l)}_{forward}}) = 1 - p^{(l)}_{forward} \end{cases} . \tag{8}$$

As for the backward pass, we have:

$$\begin{aligned} &\hat{h}^{(l)}_{backward} = h^{(l)} \odot \epsilon^{(l)}_{backward} , \\ &\frac{\partial J}{\partial W^{(l)}} = \delta^{(l+1)} \hat{h}^{(l)T}_{backward} , \\ &\delta^{(l)} \triangleq \frac{\partial J}{\partial z^{(l)}} = (W^{(l)T} \delta^{(l+1)}) \odot (f'_l(z^{(l)}) \odot \epsilon^{(l)}_{backward}) , \end{aligned} \tag{9}$$

where $\epsilon^{(l)}_{backward}$ is drawn from the following distribution in terms of the backward dropout ratio $p^{(l)}_{backward}$:

$$\begin{cases} P(\epsilon^{(l)}_i = 0) = p^{(l)}_{backward} \\ P(\epsilon^{(l)}_i = \frac{1}{1-p^{(l)}_{backward}}) = 1 - p^{(l)}_{backward} \end{cases} , \tag{10}$$

Compared with the standard dropout, we employ different dropout ratio in the forward and backward pass. In this way, the augmented dropout can extensively utilize the benefit of the data augmentation from the forward pass and the noisy gradient in the backward pass. At last, we summarize our proposed augmented dropout in Algorithm 1. In our implementation, the backward random variable is obtained from the forward one as $\epsilon^{(l)}_{backward} = \epsilon^{(l)}_{forward} \odot \epsilon^{(l)}$ where $\epsilon^{(l)}$ is another random variable drawn according to a small dropout ratio $p^{(l)}$. In the following experiments, we will call $p^{(l)}_{forward}$ as the forward dropout ratio and $p^{(l)}$ as the backward dropout ratio.

Algorithm 1 Augmented Dropout

```
Forward Pass:
    Get input features $h^{(l)}$ from the $l$ layer
    Draw forward random variables to corrupt features:
        $\hat{h}^{(l)}_{forward} = h^{(l)} \odot \epsilon^{(l)}_{forward}$
    Compute output features:
        $h^{(l+1)} = f_l(W_l \hat{h}^{(l)}_{forward} + b^{(l)})$
Backward Pass:
    Get the backpropagated error $\delta^{(l+1)}$ from the $l+1$ layer
    Draw backward random variables to corrupt hidden features:
        $\hat{h}^{(l)}_{backward} = h^{(l)} \odot \epsilon^{(l)}_{backward}$
    Compute the gradient:
        $\frac{\partial J}{\partial W^{(l)}} = \delta^{(l+1)} \hat{h}^{(l)T}_{backward}$
    Compute the backpropagated error:
        $\delta^{(l)} = (W^{(l)T} \delta^{(l+1)}) \odot (f_l'(z^{(l)}) \odot \epsilon^{(l)}_{backward})$
```

Table 3. The test accuracy of LeNet for MNIST

Standard		Augmented	
Dropout Ratio	Acc	Acc	Dropout Ratio
0.1	0.9916	0.9918	0.1/0.0001
0.2	0.9922	0.9923	0.2/0.0001
0.3	0.9918	0.9925	0.3/0.0001
0.4	0.9917	0.9920	0.4/0.0001
0.5	0.9922	0.9925	0.5/0.0001

6. Experiments

In this section, we will conduct extensive experiments to verify the performance of our proposed augmented dropout.

6.1. Datasets

Throughout our experiments, we employ four widely used datasets: MNIST (LeCun et al., 1998), SVHN (Netzer et al., 2011), CIFAR10 (Krizhevsky, 2009), and CIFAR100 (Krizhevsky, 2009). In particular, MNIST contains 70,000 handwritten digits images of size 28×28. 60,000 images are used for the training set and the rest 10,000 images are used for the testing set. SVHN is also a digits images dataset. Its training set has 73,257 images and the testing set contains 26,032 images. CIFAR10 has 60,000 color images of size 32×32 from 10 classes. The training set contains 50,000 images while the testing set includes the rest 10,000 images. CIFAR100 also has 60,000 color images of size 32×32. But the number of classes is 100. The size of the training and testing set is same as CIFAR10.

6.2. Results

To verify the performance of the augmented dropout, we evaluate it on different neural networks, including LeNet (LeCun et al., 1998), ConvNet-Quick (Hinton et al., 2012), ALL-Conv (Springenberg et al., 2014), and Network in Network (NiN) (Lin et al., 2013). In addition, we use Caffe (Jia et al., 2014) to implement these networks and all the classification accuracy is the mean value of four runs.

LeNet: As for MNIST dataset, we employ the widely used 5-layer LeNet (LeCun et al., 1998). Here, dropout is performed on the input of the last fully connected layer. We adopt the implementation from the Caffe library[1]. In addition, we train it with the stochastic gradient descent method with the momentum of 0.9 and batch size of 100. The weight decay hyperparameter is set to 0.0005. The learning rate starts from 0.01 and is divided by 10 at 6,000 iterations. The optimization procedure is terminated at 12,000 iterations. The classification result with different dropout ratio is reported in Table 3, which is the mean value of four runs. Here, we set the forward dropout ratio as the standard one while the backward dropout ratio is fixed to 0.0001. It can be seen that the augmented dropout outperforms the standard dropout consistently for all cases, which verifies the effectiveness of our proposed method.

ConvNet-Quick[2]: This is the Caffe implementation of ConvNet (Hinton et al., 2012). Here, dropout is conducted on the input of two fully connected layers. For this simple network, we use two datasets: SVHN and CIFAR10. In this experiment, we did not use the data augmentation technique. The only preprocessing step is to subtract the mean value. The training setting is exactly same as that in Section 4. In Table 4, we report the classification result of ConvNet-Quick running on SVHN and CIFAR10 respectively. Here, different dropout ratios are used to extensively evaluate the performance of our proposed method. Same with LeNet, we set the forward dropout ratio as the standard one and vary the backward dropout ratio to improve the performance of the standard dropout. From this table, we can also find that our proposed method outperforms the standard dropout method consistently, which verifies the effectiveness of our proposed method.

ResNet-20 (He et al., 2016): To further verify the performance of the proposed method, we evaluate it on ResNet-20.

[1] https://github.com/BVLC/caffe/blob/master/examples/mnist/lenet_train_test.prototxt

[2] https://github.com/BVLC/caffe/blob/master/examples/cifar10/cifar10_quick_train_test.prototxt

Table 4. The test accuracy of ConvNet-Quick

Datasets	Standard		Augmented	
	Dropout Ratio	Acc	Acc	Dropout Ratio
SVHN	0.1	0.9240	0.9248	0.1/0.0002
	0.2	0.9231	0.9258	0.2/0.0002
	0.3	0.9249	0.9250	0.3/0.0002
CIFAR10	0.1	0.7657	0.7674	0.1/0.0002
	0.2	0.7617	0.7655	0.2/0.0002
	0.3	0.7523	0.7606	0.3/0.0002

Table 5. The test accuracy of ResNet-20

Datasets	Standard		Augmented	
	Dropout Ratio	Acc	Acc	Dropout Ratio
SVHN	0.1	0.9618	0.9627	0.1/0.0002
	0.2	0.9626	0.9648	0.2/0.0002
	0.3	0.9655	0.9667	0.3/0.0002
CIFAR10	0.1	0.9193	0.9196	0.1/0.0001
	0.2	0.9176	0.9195	0.2/0.0001
	0.3	0.9163	0.9177	0.3/0.0001
CIFAR100	0.1	0.6762	0.6786	0.1/0.0001
	0.2	0.6748	0.6770	0.2/0.0001
	0.3	0.6686	0.6688	0.3/0.0001

Specifically, the dropout is performed between two convolutional blocks as Conv-BN-ReLU-Dropout-Conv-BN-ReLU. For this complicated network, we evaluate it with three datasets: SVHN, CIFAR10, and CIFAR100. In this experiment, we did not use the data augmentation technique for SVHN. For the other two datasets, we employ the simple data augmentation as Section 4. The training setting is also exactly same as that in Section 4. The classification result is reported in Table 5. It can be seen that the proposed augmented dropout method has better performance than the standard dropout method. Thus, we can conclude that the augmentation from the backward pass does help to further improve the performance of the standard dropout.

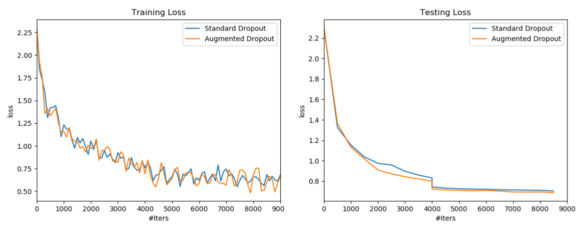
ALL-Conv (Springenberg et al., 2014) & **NiN** (Lin et al., 2013): We compare the proposed augmented dropout method with three baseline methods: Monte Carlo dropout (Bulò et al., 2016), distillation dropout (Bulò et al., 2016), and expectation-linear (EL) dropout (Ma et al., 2016). For these two networks, we evaluate them on CIFAR10. Here, the global contrast normalization and ZCA are employed to preprocess the input image. In Table 6, we report the test error of different methods for ALL-Conv and NiN. It can be seen that the proposed augmented dropout shows superior performance over these baseline methods, which further verifies the effectiveness of our proposed method.

At last, to further demonstrate the performance of the proposed method, we show the training and testing loss of the standard dropout and our proposed augmented dropout. The result of ConvNet-Quick is shown in Figure 3. Here, we report the result of standard dropout with dropout ratio being

Table 6. The test error of different dropout methods. † denotes the result is extracted from (Bulò et al., 2016). ‡ denotes the result is extracted from (Ma et al., 2016).

Method	ALL-Conv	NiN
Monte Carlo	10.60†	11.04†
Distillation	10.80†	11.14†
EL	10.86‡	-
Augmented	10.04	10.28

0.2 and augmented result with forward dropout ratio being 0.2 and backward dropout ratio being 0.0002. It can be seen that these two methods have almost the same training loss. But the testing loss of our augmented dropout is a little better than that of the standard one, which verifies the noisy gradient in the backward pass of the augmented dropout does help to converge to a better minimum. Thus, it is beneficial to use a different dropout ratio in the forward and backward pass to improve the generalization performance of deep neural networks.



(a) Training Loss (b) Testing Loss

Figure 3. The loss function of standard dropout and augmented dropout for ConvNet-Quick.

7. Conclusion

In this paper, to understand the mechanism of dropout, we propose the novel forward and backward dropout methods and show that dropout has a different effect on the forward and backward pass. Meanwhile, the forward and backward pass achieves their best performance at different dropout ratios. Based on these observations, we propose the augmented dropout which employs different dropout ratio for the two passes to fully utilize their benefit. Extensive experimental results confirmed the effectiveness of our method.

Acknowledgement

H.G. and H.H. were partially supported by U.S. NSF IIS 1836945, IIS 1836938, DBI 1836866, IIS 1845666, IIS 1852606, IIS 1838627, IIS 1837956.

References

Bouthillier, X., Konda, K., Vincent, P., and Memisevic, R. Dropout as data augmentation. *arXiv preprint arXiv:1506.08700*, 2015.

Brock, A., Donahue, J., and Simonyan, K. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Bulò, S. R., Porzi, L., and Kontschieder, P. Dropout distillation. In *International Conference on Machine Learning*, pp. 99–107, 2016.

Daneshmand, H., Kohler, J., Lucchi, A., and Hofmann, T. Escaping saddles with stochastic gradients. *arXiv preprint arXiv:1803.05999*, 2018.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059, 2016.

Gao, H. and Huang, H. Joint generative moment-matching network for learning structural latent code. In *IJCAI*, 2018.

Gao, W. and Zhou, Z.-H. Dropout rademacher complexity of deep neural networks. *Science China Information Sciences*, 59(7):072104, 2016.

Ge, R., Huang, F., Jin, C., and Yuan, Y. Escaping from saddle pointsonline stochastic gradient for tensor decomposition. In *Conference on Learning Theory*, pp. 797–842, 2015.

Ghiasi, G., Lin, T.-Y., and Le, Q. V. Dropblock: A regularization method for convolutional networks. In *Advances in Neural Information Processing Systems*, pp. 10750–10760, 2018.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Helmbold, D. P. and Long, P. M. Surprising properties of dropout in deep networks. *The Journal of Machine Learning Research*, 18(1):7284–7311, 2017.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Jain, P., Kulkarni, V., Thakurta, A., and Williams, O. To drop or not to drop: Robustness, consistency and differential privacy properties of dropout. *arXiv preprint arXiv:1503.02031*, 2015.

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

Jin, C., Ge, R., Netrapalli, P., Kakade, S. M., and Jordan, M. I. How to escape saddle points efficiently. *arXiv preprint arXiv:1703.00887*, 2017.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Larsson, G., Maire, M., and Shakhnarovich, G. Fractalnet: Ultra-deep neural networks without residuals. *arXiv preprint arXiv:1605.07648*, 2016.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lin, M., Chen, Q., and Yan, S. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

Ma, X., Gao, Y., Hu, Z., Yu, Y., Deng, Y., and Hovy, E. Dropout with expectation-linear regularization. *arXiv preprint arXiv:1609.08017*, 2016.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Mou, W., Zhou, Y., Gao, J., and Wang, L. Dropout training, data-dependent regularization, and generalization bounds. In *International Conference on Machine Learning*, pp. 3642–3650, 2018.

Neelakantan, A., Vilnis, L., Le, Q. V., Sutskever, I., Kaiser, L., Kurach, K., and Martens, J. Adding gradient noise improves learning for very deep networks. *arXiv preprint arXiv:1511.06807*, 2015.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. 2011.

Ren, S., He, K., Girshick, R., and Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pp. 91–99, 2015.

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

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

Tompson, J., Goroshin, R., Jain, A., LeCun, Y., and Bregler, C. Efficient object localization using convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 648–656, 2015.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

Wager, S., Wang, S., and Liang, P. S. Dropout training as adaptive regularization. In *Advances in neural information processing systems*, pp. 351–359, 2013.

Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., and Fergus, R. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*, pp. 1058–1066, 2013.

Wen, W., Wang, Y., Yan, F., Xu, C., Chen, Y., and Li, H. Smoothout: Smoothing out sharp minima for generalization in large-batch deep learning. *arXiv preprint arXiv:1805.07898*, 2018.

Zagoruyko, S. and Komodakis, N. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Zhang, Y., Liang, P., and Charikar, M. A hitting time analysis of stochastic gradient langevin dynamics. *arXiv preprint arXiv:1702.05575*, 2017.