Random Shifting for CNN: a Solution to Reduce Information Loss in Down-Sampling Layers

Gangming Zhao[1,4], **Jingdong Wang**[5], **Zhaoxiang Zhang**[1,2,3,4*]
[1]Research Center for Brain-inspired Intelligence, CASIA
[2]National Laboratory of Pattern Recognition, CASIA
[3]CAS Center for Excellence in Brain Science and Intelligence Technology
[4]University of Chinese Academy of Sciences
[5]Microsoft Research

Abstract

Down-sampling is widely adopted in deep convolutional neural networks (DCNN) for reducing the number of network parameters while preserving the transformation invariance. However, it cannot utilize information effectively because it only adopts a fixed stride strategy, which may result in poor generalization ability and information loss. In this paper, we propose a novel random strategy to alleviate these problems by embedding random shifting in the down-sampling layers during the training process. Random shifting can be universally applied to diverse DCNN models to dynamically adjust receptive fields by shifting kernel centers on feature maps in different directions. Thus, it can generate more robust features in networks and further enhance the transformation invariance of down-sampling operators. In addition, random shifting cannot only be integrated in all down-sampling layers including strided convolutional layers and pooling layers, but also improve performance of DCNN with negligible additional computational cost. We evaluate our method in different tasks (e.g., image classification and segmentation) with various network architectures (i.e., AlexNet, FCN and DFN-MR). Experimental results demonstrate the effectiveness of our proposed method.

1 Introduction

Deep convolutional neural networks (DCNN) have been widely studied in computer vision and pattern recognition. They achieve impressive performances in many vision applications such as image classification, object detection [Girshick *et al.*, 2014], image segmentation [Long *et al.*, 2015] and edge detection [Xie and Tu, 2015]. The classic convolutional neural networks consist of alternative stacked convolutional layers and spatial pooling layers. Pooling can be regarded as a form of non-linear down-sampling, which reduces the dimensionality of intermediate representations and provides small transformation invariance. In addition, many researchers utilize strided convolution to substitute pooling as another down-sampling operator. In traditional DCNN models, down-sampling operators can reduce the computational cost and improve the performance of networks. However, we argue that the traditional fixed strategy cannot mine information effectively, which may lead to poor generalization ability and information loss.

We propose random shifting to alleviate these problems by exploring information compared with traditional down-sampling layers during the training of DCNN. It can be universally embedded into diverse DCNN models to improve performance by shifting kernel centers on feature maps in different directions. The offsets of kernel center shifting are ran-

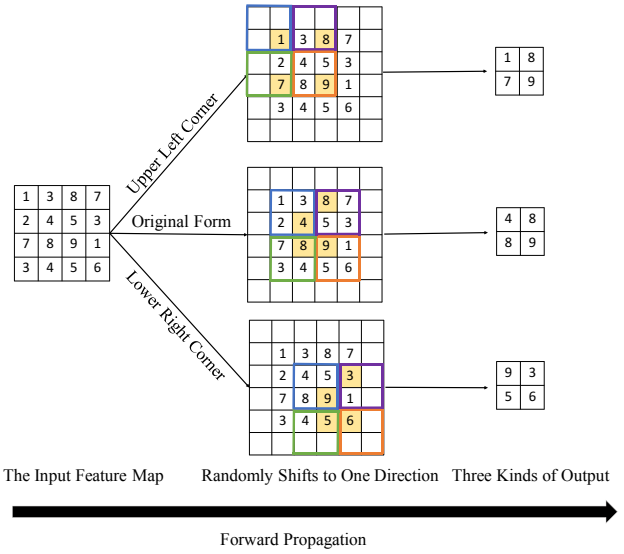

Figure 1: An example of random shifting pooling operator. The outputs are marked yellow. The dimension of the input feature map is (4, 4); the kernel size is (2, 2) and the stride is (2, 2). After pooling, the size of the feature map is reduced to (2, 2). If we add two rows and two columns offset variables, feature map can be shifted to upper left corner, lower right corner or keep original form. At each iteration, random shifting utilizes different information by shifting feature map to the three directions. It can be observed that the random shifting pooling explores more information compared with the traditional pooling.

*Corresponding author. (zhaoxiang.zhang@ia.ac.cn)

domly generated respectively for each iteration during training. DCNN with random shifting can produce more robust convolution filters in networks and further enhance the transformation invariance of down-sampling operators. We can apply random shifting in all down-sampling layers including strided convolutional layers and pooling layers at a low computational cost.

Fig. 1 illustrates a random shifting pooling operator, where the dimension of the input feature map is (4, 4), the kernel size is (2, 2) and the stride is (2, 2). After pooling, the size of the feature map is reduced to (2, 2). As we can see in Fig. 1, the pooling operator with random shifting can utilize the information ignored compared with the traditional pooling operator. In addition, for any down-sampling layer, random shifting can further improve the performance of DCNN by utilizing more information.

2 Related Work

Many efforts have been made to improve the DCNN learning algorithm, such as xavier [Glorot and Bengio, 2010] and msra [He *et al.*, 2015] for initialization, FitNets [Romero *et al.*, 2014] and Net2Net [Chen *et al.*, 2015] for knowledge transfer and batch normalization [Ioffe and Szegedy, 2015] for both performance and efficiency improvement. In addition, GoogLeNet [Szegedy *et al.*, 2015], ResNet [He *et al.*, 2016] and Deep Fusion Network (DFN) [Zhao *et al.*, 2016] are shown to be able to effectively train a very deep network. Furthermore, Swapout ensembles the advantages of dropout [Srivastava *et al.*, 2014], stochastic depth [Huang *et al.*, 2016] and residual architectures [He *et al.*, 2016] to further improve the performance of DCNN.

In deep convolutional neural networks, traditional down-sampling is widely adopted to reduce the number of network parameters and preserve transformation invariance. However, it inevitably results in information loss. This problem is barely studied. To our best knowledge, the work by Springenberg *et al.* [Springenberg *et al.*, 2014] is the only paper which dealt with this problem. They found that max-pooling can simply be replaced by a strided convolutional layer without loss in accuracy on several image recognition benchmarks. Different from their work [Springenberg *et al.*, 2014], our proposed method mainly explores how to reduce information loss by randomly shifting kernel centers on feature maps during training as illustrated in Fig. 1. In addition, random shifting can be applied in all down-sampling layers including strided convolutional layers and pooling layers. Since randomness has been widely used to improve the performance in previous research (e.g., dropout [Srivastava *et al.*, 2014] and drop-connect [Wan *et al.*, 2013].), it is the first time to introduce randomness into the down-sampling process. Random shifting as another random strategy has dynamic larger receptive field by randomly shifting kernel centers on feature maps during training. The trained network with random shifting cannot only generate more robust features, but also can fit the distribution of the test dataset better.

Considering that random image cropping is widely used data augmentation technique, random shifting can be treated as a feature augmentation method. There is a main difference that random shifting focuses on feature maps while random cropping takes advantage of the input image.

However, cropping as a data augmentation method has a non-negligible disadvantage. There are many overlapping crops fed into the network for each image which leads to repetitive computations in the forward and backward processes. Fortunately, Fully Convolutional Networks (FCN) [Long *et al.*, 2015] [Chen *et al.*, 2014] can avoid cropping image into patches. These works have been proposed to use fully convolutional network in image parsing. As an alternative to solve the similar problem in image recognition, some researchers [Simonyan and Zisserman, 2014] [Oquab *et al.*, 2014] use fully convolution to extend the output of the network form $1 \times 1 \times c$ to $n \times m \times c$, where c, n and m denote the number of class, the width and height of output, respectively. denote this method as FCN-style. It performs slightly better to use multiple crops, which is described in [Simonyan and Zisserman, 2014]. To improve the performance of FCN-style and avoid the repetitive computations of random cropping, we combine FCN-style with random shifting which dynamic adjusts the receptive field and utilizes more contextual information.

3 Model Description

In this section, we first describe strided convolutional operators. Then we introduce the process of random shifting applied to strided convolutional layers in details.

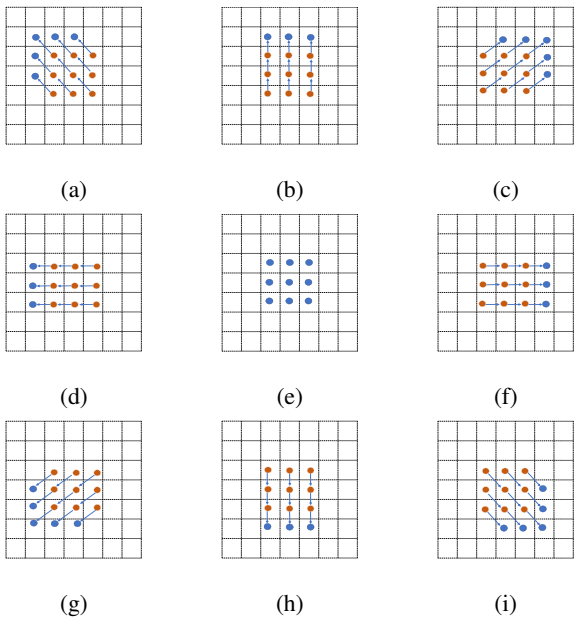

Figure 2: Illustration of sampling grids in 3×3 regular and random shifting convolutions. (a), (b), (c), (d), (f), (g), (h), and (i): random shifts kernel centers in different directions with augmented offsets (blue arrows) in a random shifting convolution. (e): a traditional convolution.

Algorithm 1 Random Shifting

Input: $C_s^k(x)$

1: Calculate the maximum available value r of random shifting. Practically, to keep all the kernel centers not fall out of feature maps, we consider three factors: kernel stride, kernel size and padding. We denote stride, kernel size and padding in a down-sampling layer as s, k and p, respectively. We calculate r as:

$$r = min(s-1, (k-1)/2 - p).$$

2: Randomly select a value O as the offset from the available domain of random shifting

$$\{-r, -r+1, ..., 0, ..., r-1, r\}$$

at each iteration. Here 0 means the kernel center is kept just as the conventional DCNN, negative numbers mean we shift the kernel center to left of the conventional kernel center and positive ones mean a shift to the right.

3: Random shifting is defined by the function $Y(f)$, where f is the down-sampling layer.

$$Y(C_s^k) = \sum_{\substack{i' \in [(i-1)s+1+O, (i-1)s+k+O] \\ j' \in [(j-1)s+1+O, (j-1)s+k+O]}} w_{m,i',j'} * x_{m,i',j'}$$

Output: Y

3.1 Down-Sampling Operators

Recently, many researches adopt strided convolutional layer as an alternative to pooling in DCNN. We use a strided convolutional layer for illustrating how to apply random shifting. Let x $\in R^{c*h*w}$ be the input feature map before a strided convolutional layer. This strided convolutional layer with kernel size of k $\times$ k and stride of s $\times$ s, which is defined by the function y= $C_s^k(x)$, y $\in R^{c*\lceil \frac{h-k}{s}+1 \rceil * \lceil \frac{w-k}{s}+1 \rceil}$, we extend the size of convolutional kernel w to h $\times$ w, and

$$y_{m,i,j} = \sum_{\substack{i' \in [(i-1)s+1, (i-1)s+k], i' \leq h \\ j' \in [(j-1)s+1, (j-1)s+k], j' \leq w}} w_{m,i',j'} * x_{m,i',j'} \quad (1)$$
$$m \in [1, c], i \in [1, \frac{h}{s}], j \in [1, \frac{w}{s}].$$

Max pooling and strided convolution reduce the dimensionality of feature maps through down-sampling the input representation. These two operations prevent over-fitting by providing an abstracted form of the representation. In addition, down-sampling operators reduce the computational cost and preserve small transformation invariance to the internal representation. We propose random shifting to enhance the effect of transformation invariance and reduce the information loss during down-sampling as illustrated in Fig. 1.

3.2 Random Shifting

Random shifting can be applied in any down-sampling layer to reduce information loss. As long as the kernel stride on feature map is greater than 1, the available domain of random shifting is not empty. At each iteration of training, we randomly choose one from the available domain of random shifting as the offset. Algorithm 1 describes the process of random shifting in details. Random shifting can further enhance the transformation invariance of max pooling and strided convolutional layers because the random strategy expands the receptive field by shifting kernel centers on feature maps. Down-sampling layer with random shifting can generate more robust features. In addition, random shifting requires nearly no time for computation. The implementation is straightforward. Fig. 2 shows how to effectively increase receptive field by utilizing different information in a random shifting convolution during forward process.

4 Experiments

To demonstrate the effectiveness of our proposed method, we design four experiments to:

1) testify different kinds of tactics for adding random shifting.

2) explore the relation between random shifting and data augmentation methods such as image cropping and random flipping.

3) exploit both full convolution and random shifting for object recognition.

4) further explore better performance and verify the applicability of random shifting.

4.1 Datasets

In our experiments, we use three datasets that are commonly used for object recognition: CIFAR-10 [Krizhevsky and Hinton, 2009], CIFAR-100 [Krizhevsky and Hinton, 2009] and ImageNet [Berg *et al.*, 2010]. And SIFT Flow [Liu *et al.*, 2016] is used for segmentation. All the four datasets are described as follows and the network settings are introduced in the corresponding experimental section.

CIFAR-10 The CIFAR-10 dataset has 50, 000 images for training and 10, 000 for testing. All images are tiny RGB images with a size of 32 $\times$ 32 $\times$ 3, falling into 10 categories with 6, 000 images per class. The dataset is preprocessed by global contrast normalization and ZCA whitening.

CIFAR-100 The CIFAR-100 dataset is similar to the CIFAR-10, except that it has 100 classes. There are 600 images per class, where we use 500 images for training and 100 images for testing. We use the same network settings as those in CIFAR-10 except that the last layer outputs 100 feature maps instead of 10.

ImageNet The ImageNet 2012 dataset consists of 1.2 million training, 50,000 validation, and 100,000 testing images, which are categorized in 1,000 classes.

SIFT Flow SIFT Flow is a dataset of 2,688 images with pixel labels for 33 semantic categories (bridge, mountain, sun), as well as three geometric categories (horizontal, vertical, and sky).

4.2 Evaluation Metrics and Notations

For the experiments on the evaluation of random shifting in object recognition, we use the recognition accuracy to verify its effectiveness. For the experiments on the comparison between conventional convolution network and FCN-

style with/without random shifting, we consider two aspects: recognition accuracy and the training speed.

For notations, *Normal* denotes normal training without any kinds of data augmentation. *Aug* denotes the network is trained with random image cropping and image flipping. If random shifting is adopted, we will denote the specific layer or layers where it is added to. For example, *SPool1* means random shifting is added to the pool1 layer. *Crop, Flip and SConv1, SPool1, 2, 5* means image cropping, image flipping and random are all used in conv1, pool1, pool2 and pool5 layer. But sometimes random shifting is briefly denoted as rand for simplicity. In addition, *FCN + 128 and SPool1* means that we use the FCN-style in which the batch size is 128, the size of the output classification map is 2×2 and the random strategy is applied in pool1.

4.3 How to Use Random Shifting

In this section, to provide a guidance on how to use our method in practice, we explore two questions: 1) how to set learning rate during training when random shifting is used. 2) how to decide which layers random shifting should be added.

We adopt AlexNet [Krizhevsky *et al.*, 2012] to explore the influence of random shifting on ImageNet. AlexNet has 5 convolutional layers and 3 fully-connected layers. The first and second convolutional layers are followed by a pooling layer and a LRN layer respectively. Random shifting can be applied in the first convolutional layer and all the pooling layers, considering that their strides are not 1×1. Because we only aim at providing an illustrative comparison of the relative benefits of combining FCN with random shifting on this dataset, we do not adopt the state-of-the-art networks such as VGG [4], GoogleNet [5], which cost much more time for training.

For CIFAR, we adopt NIN [Lin *et al.*, 2014] for experiments. The network contains 3 convolutional layers with 192 filter channels. We also use the mlpconv layer after each convolutional layer, and a global averaged pooling scheme with kernel size 8 for the output prediction. ReLU neuron with 0.5 dropout rate and 3×3 max pooling with stride 2 are used after the first two convolutional layers. Since there are no strided convolutional layers in the network, we only apply random shifting into the two pooling layers.

And to perform image cropping, for CIFAR-10, we augment the datasets by zero-padding 4 pixels on each side, resulting in 40×40 images. For ImageNet, we follow the conventional image preprocessing method which resizes all the images to 255×255 and subtracts them with the mean image.

Method	CIFAR-10
Normal(baseline)	89.59%
SPool1	**90.39%**
SPool2	89.61%
SPool1, SPool2	90.05%

Table 1: Comparison of performing random shifting in different layers on CIFAR-10. Network in Network is used as baseline.

Method	ImageNet
Aug(baseline)	80.19%
SConv1	80.25%
SPool1	80.23%
SPool2	80.22%
SPool5	80.63%
SConv1 and SPool1	80.20%
SPool1 and SPool2	**80.60%**
SPool2 and SPool5	**80.80%**

Table 2: Comparison of performing random shifting within different layers on ImageNet. AlexNet is used as baseline.

Learning Rate

When random shifting is used, we experimentally find learning rate is better set to be lower than the original one for making the model converge better. For example, on CIFAR-10, the initial learning rate for training with/without data augmentation is 0.1, we reduce it to 0.07 when random strategy is used.

Where to Add

Table 1 shows the obtained results by adding random shifting to different layers on CIFAR-10. If the size of the input image is small, adding random shifting in lower layers is a good choice. As shown in Table 1, random shifting can be added to both pool1 and pool2, separately, but when adding random strategy to pool2 or to pool1 and pool2 simultaneously, the performance is not as good. The small input images limit the size of the feature map. Therefore, changing the kernel centers by few pixels in higher layers means shifting a lot on the input image, which results in an unstable network.

Table 2 shows the results on ImageNet. *Aug* is the baseline, where we achieve a top-5 accuracy of 80.19% on the validation set, using just the center crop in testing. It's quite near to 80.2% as reported in [Jia *et al.*, 2014]. As for large input images, Table 2 shows that when random strategy is added to lower layers such as conv1, pool1 or pool2, no obvious performance gain is achieved. The feature maps in low layers usually have a large size, and changing the kernel centers by a few pixels on these feature maps is trifling. Therefore, in this situation, it's better to add random shifting in higher layer or multiple layers.

Method	CIFAR-10	CIFAR-100
Normal$^{Network\ in\ Network}$	89.59%	64.32%
Crop	90.19%	65.18%
Flip	90.62%	66.19%
SPool1	90.43%	65.20%
Crop and SPool1	90.21%	65.19%
Flip and Crop	91.74%	67.56%
Flip and SPool1	91.72%	67.76%

Table 3: The accuracy of normal, cropping, flipping and random shifting on CIFAR-10, and CIFAR-100.

Method	ImageNet
Normal^{AlexNet}	78.15%
Crop	78.30%
Flip	78.62%
Crop and SPool2, SPool5	**80.08%**
Flip and Crop	80.19%
Flip and SPool2, SPool5	**80.76%**

Table 4: The accuracy of normal, cropping, flipping and random shifting on ImageNet.

4.4 Random Shifting vs. Data Augmentation

Random shifting can be considered as a data augmentation method such as cropping. But unlike random image cropping which only acts on the original image, our random strategy acts on the feature maps in different down-sampling layers. They are not equivalent with each other because the random strategy goes deeper into the network. To quantitatively illustrate the relations between the proposed random shifting and existing data augmentation methods such as image cropping and image flipping, we perform comparison experiments on two datasets with small input images (CIFAR-10 and CIFAR-100) and a dataset with big input images (ImageNet).

The results on CIFAR-10 and CIFAR-100 are shown in Table 3. From these results, we can draw such a conclusion: for small input images, the improvements of performance with random shifting or random cropping are similar. There is a slight performance gain when using image flipping and random shifting simultaneously compared with using image flipping alone. Table 4 shows the results on ImageNet. When the random strategy is added to the proper layers, obvious accuracy gain can also be obtained, especially when we add random strategy to pool2 and pool5 simultaneously.

Analysis The results presented above demonstrate that for small input images, the proposed random shifting has similar effect as image cropping. For small input images, image cropping has already enriched the image information to the best extent, so adding random shifting cannot achieve better performance. In this situation, random shifting can be regarded as an alternative to image cropping, which achieves similar performance but is more efficient. When it comes to more complex networks and larger input images, random shifting can be combined with data augmentation methods such as image cropping and image flipping to further improve the performance.

4.5 Object Recognition

In this section, we combine the proposed random shifting with FCN-style for image recognition on the three datasets. Table 5 shows the detailed network settings. For each dataset, the first two rows correspond to the FCN-style network setting and the third row corresponds to the baseline setting. In Tables 7 and 8 baseline networks are all trained with image cropping and random flipping on the fly.

CIFAR To get a dense classification map as supervision information, we resize each input image in CIFAR-10 to 36×36, and thus acquire a 2×2 output in FCN-style network. We perform image cropping/random shifting and random flipping simultaneously during conventional training, and random flipping with/without random shifting during FCN-style training. By utilizing FCN-style and random shifting, the whole training time is reduced by 36% as shown in Table 6. Besides, the proposed method achieves a recognition rate of 91.52% and 91.72% with/without FCN, both of which outperform original network as shown in Table 7. Tables 6 and 7 give the results of the final accuracy on test data and the training time, respectively. The consistent better performance again demonstrates the advantage of our proposed method.

	BTCH-S	LEARN-R	TIM-I	TIM-P-I
C10/100	32	0.065	24w	1
	64	0.08	12w	2
	128	0.1	12w	3.12
ImageNet	64	0.001	90w	1
	96	0.001	60w	1.5
	256	0.01	45w	2.8

Table 5: The network settings and information for three datasets under different batch sizes. BTCH-S=Batch Size, LEARN-R=Learning Rate, TIM-I=Time Iterations, TIM-P-R=Time Per Iterations, C10=CIFAR-10, C100=CIFAR-100.

Datasets	C10/100	ImageNet
Baseline	1.56	1.40
FCN, Flip and Rand	1.00	1.00

Table 6: The relative training time comparison on CIFAR-10, CIFAR-100 and ImageNet. C10=CIFAR-10, C100=CIFAR-100.

ImageNet To further illustrate the effectiveness of random shifting on larger scale training datasets, we test both original AlexNet and FCN-style AlexNet on ImageNet 2012 dataset. It should be noted that our goal is not to directly compete with the best performing result in the challenge, but to provide an illustrative comparison of the relative benefits of combining FCN with random shifting on this dataset. We follow the conventional image preprocessing procedure that resizes all the images to 259×259 and subtracts them with the mean image. Tables 6 and 8 give the results of the final recognition accuracy and the training time on ImageNet, respectively. Our proposed method reduces the training time by 29% and improves the recognition rate by 0.67% and 0.56% with/without FCN compared with original AlexNet.

Analysis Experimental results on three datasets in Tables 7 and 8 show that random shifting can obviously improve the

Method	CIFAR-10	CIFAR-100
Network in Network(baseline)	91.19%	66.74%
Flip and Rand	**91.72%**	**67.76%**
FCN+32+Flip	90.01%	65.12%
FCN+64+Flip	91.11%	66.38%
FCN+64+Flip, SPool1	**91.52%**	**67.48%**

Table 7: FCN-style with different mini-batches vs. normal training with augmentation on CIFAR-10 and CIFAR-100 test sets.

Method	ImageNet
AlexNet(baseline)	80.20%
Flip and Rand	**80.76%**
FCN+64+Flip	78.89%
FCN+96+Flip	80.47%
FCN+96+Flip and SPool2, SPool5	80.55%
FCN+96+Flip and SConv1, SPool1,2,5	**80.87%**

Table 8: FCN-style with different mini-batches vs. normal training with augmentation on ImageNet test sets.

Method	P.ACC	M.ACC	M.IU	F.W.IU
FCN-32s	84.3%	44.8%	34.0%	74.6%
FCN-32s, SPool1	84.4%	**45.9%**	34.2%	74.8%
FCN-16s	85.2%	51.7%	39.5%	76.1%
FCN-16s, SPool1	85.2%	**52.4%**	39.4%	76.2%

Table 10: Empirical comparison of random shifting with FCN, in terms of different evaluation indexes on SIFT Flow with class segmentation. P.ACC: Pixel Accuracy, M.ACC: Mean Accuracy, M.IU: Mean IU, F.W.IU: Frequency Weighted IU.

performance no matter whether it is applied to the conventional algorithm or combined with FCN-style, which suggests that it is a useful type of data augmentation. Secondly, FCN-style training contributes most for network training acceleration. However, it cannot be ignored for datasets with large input images, FCN-style can also preserve contextual information which improves the performance as shown in Table 8 between baseline and FCN and Flip. In addition, for small images such as CIFAR-10 and CIFAR-100, The boundary problem in FCN sometimes may have a negative influence on the performance as shown in Table 7. The boundary problem has been well demonstrated in inference phase of [Simonyan and Zisserman, 2014]. Finally, in FCN-style network batch size can be reduced by 4 times theoretically because the input image in FCN is equivalent to 4 crops in the conventional network in our setting. However, the variability in each batch is accordingly reduced, so properly increasing the batch size can improve the recognition accuracy as shown in Tables7, 8.

4.6 Comparison with DFN and FCN Baseline

To explore better performance and verify the applicability of random shifting, in this section, we conduct experiments with DFN on two classification datasets including CIFAR-10, CIFAR-100 and FCN on SIFT Flow. DFN was proposed by Zhao, Liming *et al.* They present a merge-and-run fusion scheme to combine the two networks, and find Deeply Merge-and-Run Fused Network (DFN-MR) performs better than to ResNet. DFN consists of two parts of down-sampling layers. Per part down-sampling layer includes two convolutional layers with kernel size 3×3 stride 2×2 and two convolutional layers with kernel size 1×1 and stride 2×2. We add random shifting in two convolutional layers with 3×3 kernel size of the two down-sampling parts. Table 9 shows the evaluation results on the two benchmark datasets, respectively, in terms of accuracy. For the baseline DFN models, we list the test results from the original paper [Zhao *et al.*, 2016]. We find that random shifting improves the performance of all three models. But adding random shifting in DFN-MR3 model hasn't yielded the same improvement as other models since the baseline model has achieved excellent results.

Method	CIFAR-10	CIFAR-100
DFN-MR1	95.06%	75.54%
DFN-MR1+Rand	**95.55%**	**75.87%**
DFN-MR2	96.06%	80.75%
DFN-MR2+Rand	**96.31%**	**81.00%**
DFN-MR3	96.43%	81.00%
DFN-MR3+Rand	**96.50%**	**81.13%**

Table 9: The classification accuracy comparison between deep fusion network and random shifting on CIFAR-10 and CIFAR-100 test set.

Also, we embed random shifting into FCN for semantic segmentation. Firstly, we add random shifting in pool1 layer of FCN-32s and use VGG-16 to fine-tune the model which achieves higher mean accuracy compared with original FCN-32s. Then we use acquired model in the first step to fine-tune FCN-16s, which shows that random shifting also improves the mean accuracy of FCN-16s. Table 10 gives the results of our method.

5 Conclusion

Our work focus on the information loss problem in convolutional neural networks. Specifically, we develop a novel and general technique to reduce the information loss in down-sampling layers by applying the proposed random shifting into the DCNN. Compared with the fixed stride strategy, we adopt the random strategy to improve the robustness of network architectures and it is the first time that randomness is introduced into the down-sampling process. Our method effectively suppress the contextual loss and improve the performance of the standard networks. Experimental results with our models on three benchmark datasets including CIFAR-10, CIFAR-100, and ImageNet demonstrate that DCNN models trained with random shifting achieve better performance compared to the corresponding baseline models. For semantic segmentation, we also achieve a solid improvement which further illustrate the superiority of random shifting. Additionally, we embed random shifting into FCN-style for training the classifiers. It not only reduces the training time but also improves the performance of the network. Finally, there are many interesting extensions of the present method. For example, more offset directions can be added to further enhance the randomness.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant 61375036, 61511130079, and 61602481. Zhaoxiang Zhang is the corresponding author of this paper.

References

[Berg *et al.*, 2010] Alex Berg, Jia Deng, and L Fei-Fei. Large scale visual recognition challenge 2010, 2010.

[Chen *et al.*, 2014] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.

[Chen *et al.*, 2015] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015.

[Girshick *et al.*, 2014] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[Glorot and Bengio, 2010] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256, 2010.

[He *et al.*, 2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[Huang *et al.*, 2016] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European Conference on Computer Vision*, pages 646–661. Springer, 2016.

[Ioffe and Szegedy, 2015] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[Jia *et al.*, 2014] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[Krizhevsky and Hinton, 2009] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[Lin *et al.*, 2014] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *Computer Science*, 2014.

[Liu *et al.*, 2016] Ce Liu, Jenny Yuen, and Antonio Torralba. Sift flow: Dense correspondence across scenes and its applications. In *Dense Image Correspondences for Computer Vision*, pages 15–49. Springer, 2016.

[Long *et al.*, 2015] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[Oquab *et al.*, 2014] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724, 2014.

[Romero *et al.*, 2014] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[Simonyan and Zisserman, 2014] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[Springenberg *et al.*, 2014] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[Srivastava *et al.*, 2014] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[Szegedy *et al.*, 2015] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[Wan *et al.*, 2013] Li Wan, Matthew Zeiler, Sixin Zhang, Yann L Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1058–1066, 2013.

[Xie and Tu, 2015] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1395–1403, 2015.

[Zhao *et al.*, 2016] Liming Zhao, Jingdong Wang, Xi Li, Zhuowen Tu, and Wenjun Zeng. On the connection of deep fusion to ensembling. 2016.