Received August 17, 2018, accepted September 19, 2018, date of publication October 1, 2018, date of current version October 31, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2872698

Understanding Mixup Training Methods

DAOJUN LIANG[1], **FENG YANG**[1], **TIAN ZHANG**[1], **AND PETER YANG**[2]
[1]School of Information Science and Engineering, Shandong Normal University, Jinan 250014, China
[2]Amazon, Seattle, WA 98101, USA

Corresponding author: Feng Yang (yangfeng@sdnu.edu.cn)

This work was supported by the Natural Science Foundation of China under Grant 61801277 and Grant 61373081.

ABSTRACT Mixup is a neural network training method that generates new samples by linear interpolation of multiple samples and their labels. The mixup training method has better generalization ability than the traditional empirical risk minimization method (ERM). But there is a lack of a more intuitive understanding of why mixup will perform better. In this paper, several different sample mixing methods are used to test how neural networks learn and infer from mixed samples to illustrate how mixups work as a data augmentation method and how it regularizes neural networks. Then, a method of weighting noise perturbation was designed to visualize the loss functions of mixup and ERM training methods to analyze the properties of their high-dimensional decision surfaces. Finally, by analyzing the mixture of samples and their labels, a spatial mixup approach was proposed that achieved the state-of-the-art performance on the CIFAR and ImageNet data sets. This method also enables the generative adversarial nets to have more stable training process and more diverse sample generation ability.

INDEX TERMS Machine learning, computer vision, image processing.

I. INTRODUCTION

Deep Neural Networks have made breakthroughs progress in many fields [1]–[3]. However, neural networks tend to have more parameters than the training data, which allows the neural network to overfit any training data [4]. There are many ways to avoid the overfitting of datasets with such huge parameters [1], [5]–[11]. These methods can be roughly divided into two categories: data augmentation methods and regularization methods. The data augmentation methods allow the neural network to train on more samples to avoid it remembering certain samples. The regularization methods can reduce the complexity of the network model by limiting or adjusting the model parameters. These methods are widely used in deep neural networks because they do not increase the model parameters and are easy to implement.

Recently, in DisturbLabel [12], a small number of sample labels were randomly replaced with other labels. This is a regularization method that is used at the loss layer to allow the neural network to avoid overfitting and enhance its generalization performance. In SamplePairing [13] and mixup [14], a method of training a neural network using two samples simultaneously is proposed. SamplePairing randomly picks one sample in the training set to add to the original sample and uses the original sample's label to train the network. The mixup uses a random value to weight the two samples and their corresponding labels. All of the above methods have some effect of data augmentation and regularization, and they can achieve better generalization performance than Empirical Risk Minimization (ERM) [15]. However, the reason why these methods can achieve regularization and data augmentation has not been well understood.

There are two ways to understand mixup training, one is to use the synthetic sample as a new sample and the other is to use the synthetic sample as a linear combination of the original sample. The former view shows that mixup uses a linear interpolation of different classes of samples to generate new samples. Different linear interpolation will produce different new samples, which makes the neural network have more opportunities to sample new data to avoid overfitting. This shows that mixup is a data augmentation method. We use a variable ratio to generate new samples and different ranges of sample interpolation to investigate whether data augmentation enhances generalization performance. This experiment shows that mixup, as a data augmentation method, produces far fewer new samples than its linear interpolation capability, but much larger than the number of combinations between multiple classes. The latter view shows that mixup can learn multiple samples at the same time and can avoid confusion between multiple samples so that two different categories can be easily separated. This shows that mixup plays a regularized role to a great extent. We visualize the loss function of the neural network trained with mixup and find that it is flatter than the neural network trained by ERM and the curl of the decision surface between several classes is relatively low, making neural network more easily predict the linear interpolation between samples.

Our experiment shows that there is a positive correlation between the input information of neural network and the output information of its loss layer. Based on this, the spatial images are stitched in different proportions by different combinations, and its label information is proportional to its spatial information. This method can achieve the same effect as the original mixup. Different from the way that the original mixup linearly interpolates the channels of the image, the stitching of the spatial image is to interpolate the natural distribution of the samples. This makes it two orthogonal data augmentation methods with the original mixup. Combining all kinds of mixup methods can make the training data more diverse and the regularization effect more powerful, so as to further improve the generalization ability of the network. Our method have achieved the state of the art on CIFAR [16] and ImageNet [17] datasets, and the code implemented by Pytorch[1] is available on GitHub.[2]

Generative adversarial nets (GAN) [18] has achieved good results in image generation tasks. The difficulties in this task are the instability of training [19], [20] and the problem of mode collapse [21]. Instability will lead to the network difficult to train, mode collapse will cause the network to generate a single image mode. In order to make GAN have a more stable training process and generate more diverse images, various mixup methods are used in image editing tasks to take full advantage of their data augmentation and regularization. The GAN trained using the mixup method is called MixGAN. We used CycleGAN [22] to experiment with the task of image-to-image transformation [23]–[26]. The experimental results show that the MixGAN can transform the style of the mixture of spatial and maintain the original mixing ratio. Compared with the original training method requires less training time, and the generated images have more variety of details.

II. RELATED WORK

The neural network has a lot of hyper-parameters and millions of training parameters, so that it has strong fitting ability and even can fit random noise [4]. Therefore, it is a challenging job to reduce the overfitting of neural networks.

Extending the dataset and regularizing the model can reduce the overfitting of the model from different aspects. Some data augmentation techniques directly distort data space, such as, translation, rotation, flipping, cropping, adding noises, etc. Other methods use slightly more sophisticated methods such as PatchShuffle [28] which divides the sample data into several different patches, sorting the pixels in each patch according to a certain rule, while keeping the overall structure of the image intact. [29] randomly select a rectangular area in the image, and fill the area with Gaussian noise, making the original information of the area is blocked, forcing the neural network to infer about the part of the information. SamplePairing [13] randomly picks a sample in the training set and adds it to the original sample to produce a new sample. These methods make the processed image slightly different from the original image, although the enhanced data may not be in line with the original sample distribution nor independent of the original sample. However, this can make the neural network learn other models related to the original sample, which can make the neural network have a more diversified representation ability and thus have better generalization performance.

There's a lot of work related to regularization. Dropout [5] and DropConnect [9] make each neuron more capable of representation by discarding a certain percentage of neurons or connecting paths during training. Stochastic Depth [30] averages architectures with various depths through randomly skipping layers. Swapout [31] samples from abundant set of architectures with dropout and stochastic depth as its special case. These methods serve as a model integration in depth, and [32] attempts to demonstrate that dropout is equivalent to data augmentation to some extent. In batch normalization (BN) [10], the features of each layer in the network will be given a normal distribution again, which makes the neural network easier to learn. Recently, there have been some works to regularize the model at the loss layer. DisturbLabel [12] use an extremely simple algorithm to randomly replaces a part of labels as incorrect values in each iteration. In [14], mixup trains a neural network on convex combinations of pairs of examples and their labels, which regularizes the neural network to favor simple linear behavior in-between training examples.

Neural networks have long been considered black boxes because there is no good theory to explain their generalization effect. Much of the current interpretation is based on very strong assumptions, which are quite different from the actual situation and not very practical in practice. By visualizing the loss function of neural network, the influence of different training methods on the loss function can be more intuitively understood, and the differences between different training methods can be compared better. Several works have attempted to study the loss surfaces of deep nonlinear neural networks [33]–[35]. Reference [33] attempted to understand the loss function of neural networks through studying the random Gaussian error functions of Ising models. [35] studied the loss surface along the line between a random guess, and a nearby minimizer obtained by stochastic gradient descent. We propose a method of weighting noise disturbances that can alter all network parameters to visualize the loss function.

Image-to-image translation is a class of vision and graphics problems where the goal is to learn the mapping between an input image and an output image using a training set of aligned image pairs. Some work has proposed bidirectional mapping methods when pairs of images are not available, such as [22]. The approach builds on the "pix2pix"

[1]https://pytorch.org

[2]https://github.com/liangdaojun/spatial-mixup

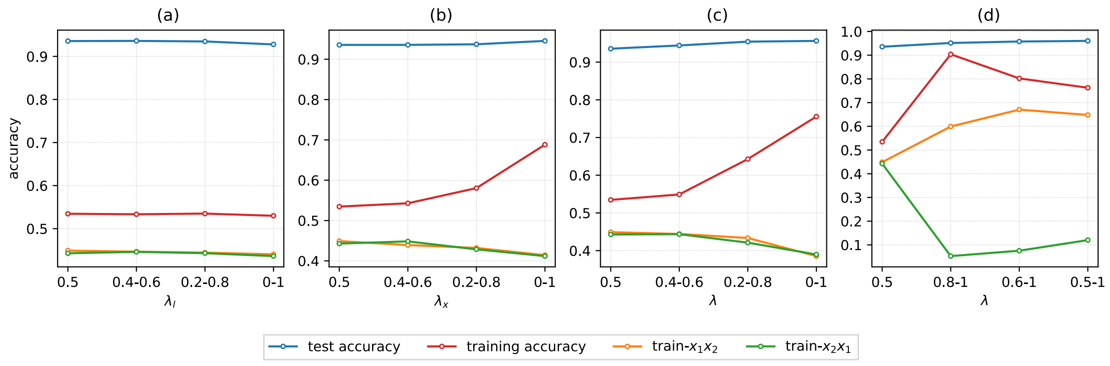


FIGURE 1. **The effect of different proportions of mixed samples and labels on network training or test accuracy. The x-axis mean the range of values for λ. The train-x_1x_2 and train-$x2x1$ means that during training or testing, sample x_1 and sample x_2 are mixed in different proportions and then input to the network, and the two maximum probabilities of the network output are taken as the categories corresponding to the two samples respectively. All experiments were done using PreAct-Resnet [27] on CIFAR-10 [16] dataset. (a) $\lambda_X = 0.5$. (b) $\lambda_l = 0.5$. (c) $\lambda_X = \lambda_l$. (d) $\lambda_X \neq \lambda_l$.**

framework of Isola *et al.* [36], which uses a conditional generative adversarial network [18] to learn a mapping from input to output images. In this paper, we combine the spatial mixup with the GAN to implement the image editing task.

III. METHOD

In this section, the generalization performance of mixups in different mixing ranges is first explored in Section III-A. In Section III-B, the ability of mixup and ERM to predict multiple samples simultaneously was tested. Section III-C uses the weighted noise disturbance method to visualize the loss function of the two training methods.

A. GENERAL MIXUP

The mixup in the original paper is implemented by using a random value to weight the samples and their labels.

$$\tilde{x} = \lambda x_i + (1-\lambda)x_j$$
$$\tilde{y} = \lambda y_i + (1-\lambda)y_j \tag{1}$$

where (x_i, y_i) and (x_j, y_j) are two examples drawn at random from our training data, and $\lambda \sim \text{Beta}(\alpha, \alpha)$ for $\lambda \in [0, 1]$, $\alpha \in (0, \text{inf})$. In the mixup, training samples and their labels simultaneously reduce the λ or $1-\lambda$ times, implying this linear relationship. The signal strength scaled by λ or $1-\lambda$ times, the corresponding label signal also scaled by λ or $1-\lambda$ times. Our experiments show that training samples and their labels do not simply have a linear relationship. We can use the different distributions of random values to separately weighted average the training samples and their labels, and can achieve the same effect as the mixup.

In order to explore the impact of the mixing of images and their labels on the performance of the network, we either interpolate the images or interpolate their labels, or interpolate them at the same time. The Uniform distribution can better control the range of λ compared to the Beta distribution, and has the same effect on some datasets. Therefore, the Uniform distribution is adopted to control the sampling of λ. We use λ_x to represent the mixing ratio of two samples x for $\lambda \in \text{Uniform}(\lambda_1, \lambda_2)$, and R_l to represent the mixing ratio of two labels y for $\lambda \in \text{Uniform}(\lambda_1, \lambda_2)$, where $0 \leq \lambda_1 \leq \lambda \leq \lambda_2 \leq 1$. When $\lambda_x = \lambda_l$, we denote them as λ.

Figure 1 shows the difference between λ_x and λ_l on network training accuracy and test accuracy. In Figure 1.a, $\lambda_x = 0.5$, so that λ_l is mixed in different ratios, which means that the same sample will correspond to a different label. It can be observed that network performance decreases with increasing range of λ_l. Similarly, in Figure 1.b, making $\lambda_l = 0.5$ and changing λ_x, which means that the linearly interpolated samples correspond to the same label, we can see that the network performance increases with the increase of the λ_x range. In contrast to the change λ_l, the network performance of the varying λ_x will be consistently higher than the change λ_l. Figure 1.a shows that using different label signals for the same sample will result in a strong regularization of the network, leading to instable learning of the training samples, resulting in underfitted networks. In Figure 1.b, λ_x was changed for all other experiments compared to $\lambda_x = 0.5$ and $\lambda_l = 0.5$, and the label was not altered. This shows that linear interpolation of samples can indeed achieve data augmentation.

In Figure 1.c, $\lambda_x = \lambda_l$, that is, the mixing ratio between the sample and their label is the same. It is noteworthy that when λ's range of variation in [0,1], it is the original mixup. It can be seen from the figure that the performance of the network increases as the range of λ changes. The same result can also be observed in Figure 1.d, but in Figure 1.d, λ_x and λ_l are limited to the asymmetrical range of variation with a mixing ratio above 0.5. Although in both graphs the performance of the network is higher than that of the only one, and the network performance is almost the same when λ is in the range of [0.5,1] and [0,1]. This shows that neural networks optimize each category alternately when training multiple samples and labels simultaneously. That is, in a forward process, when the network parameter is adjusted to a category, the other category plays a regularization role.

Predicting both categories helps to analyze the level of confusion that neural networks have over different categories. The train-x_1x_2 in the Figure 1 represents the prediction accuracy of both categories at the same time, and the train-x_2x_1 represents the precision of the neural network, which classifies two samples into each other. For instance, when the input is $\lambda x_1 + (1 - \lambda)x_2$ or $\lambda x_2 + (1 - \lambda)x_1$, the corresponding train-x_1x_2 and train-x_2x_1 accuracy measures the proportion of the outputs that ranks the top-2 prediction as (y_1, y_2). In Figure 1.a-c, the train-x_1x_2 and train-x_2x_1 are almost equal, which shows that the network can not distinguish the two samples into different categories at the same time. As the mixing range λ increases, the training accuracy and testing accuracy of the network increase, but the train-x_1x_2 and train-x_2x_1 gradually decreases. This is because as the combination of samples increases, the network needs to fit a wider distribution of loss layers during training, which will increase the probability of misclassification of samples with less information. In Figure 1.d, the difference in precision between the train-x_1x_2 and train-x_2x_1 is widened, demonstrating that their network will be less obfuscated for both samples. Because this training method uses an asymmetric combination of samples, one of the categories will result in a high prediction accuracy.

B. MIXUP AND ERM'S MULTI-CATEGORY TEST

To further compare the generalization ability of mixups and ERMs for mixed samples, we use mixup and ERM methods to test the trained networks with different mixing ratios respectively. In this experiment, λ takes values from 0 to 1 in steps of 0.01. The experiment was run on CIFAR-10 [16], using PreAct-Resnet [27]. Since λ has 100 values, we will test 100 steps. The single-class prediction accuracy and multi-class prediction accuracy of the network are shown in Figure 2. The test-x_1x_2 in the Figure 1 represents the prediction accuracy of both categories at the same time, and test-x_2x_1 represents the precision of the neural network, which classifies two samples into each other. From Figure 2, we can see that with the increase of λ, the accuracy of sample x_1 increases with "S" curve and the accuracy of sample x_2 decreases with "S" curve, and test-x_1x_2 and test-x_2x_1 are also similar. The difference is that the curve of the accuracy of the mixup varies steeply at 20-80 steps, while the ERM is relatively flat, indicating that mixup will perform better than ERM, both in a single category and in multiple categories. In addition, this is also the most effective range of the mixup's λ value. It is noteworthy that, in about 70 steps, the test-x_1x_2 accuracy of both methods peaked. The difference is that the prediction accuracy of the mixup (Figure 2.a red dotted line) begins to decline, while the ERM stays the same. The reason for this phenomenon is that when λ in [0.3,0.7], the mixup learns two categories at the same time, and beyond this range, the training mode is dominated by the category with larger label information and becomes the ERM training mode. It can be seen that both approaches will converge to the consistent position, around 54%. The reason for this is the accuracy of one category is 96%, while the other is about 10% of the random forecasts, taking the average of the two as the prediction accuracy of multiple categories.

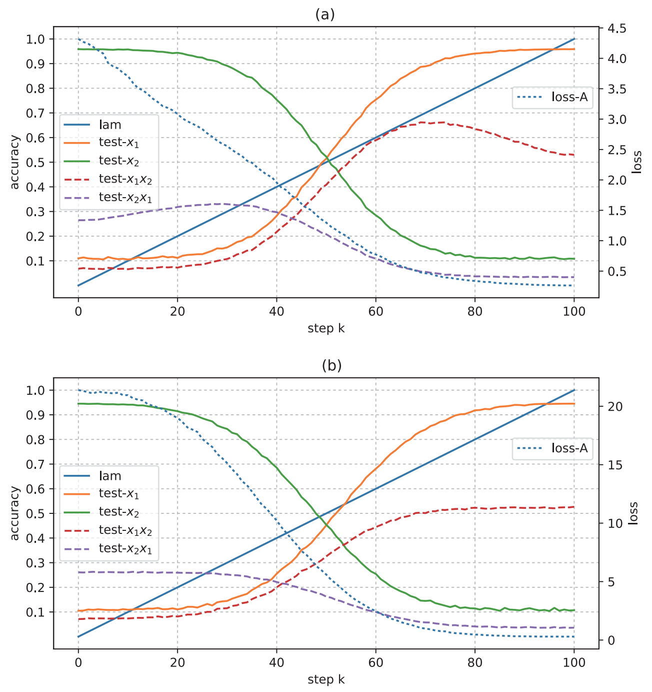


FIGURE 2. Testing of different mixed-ratio samples of trained neural networks using mixup and ERM training methods. The two y-axes represent the accuracy and loss of the network, respectively. and the x-axis represents the test step. The test-x_1x_2 and test-x_2x_1 means the top-2 accuracy: when input is $\lambda x_1 + (1 - \lambda)x_2$ or $\lambda x_2 + (1 - \lambda)x_1$, the corresponding top-2 categories is predicted as (y_1, y_2). (a) Mixup. (b) ERM.

This phenomenon also illustrates the difference between the mixup and ERM decision surfaces, see Section III-C for visualizing both training methods.

By interpolating two samples to test the neural network, the variation of the loss function of the neural network can also be obtained. The maximum cross-entropy of mixup in Figure 2 is about 4.3 and the ERM is about 21.5. This shows that the loss function of mixup is smoother in multi-category prediction, while the loss of ERM is larger. This further shows that mixup has a smoother decision surface, which makes it easier to predict the interpolation between multiple categories.

C. VISUALIZATION OF LOSS FUNCTIONS

The visualization of the loss function of different training methods helps to understand the decision-making behavior of neural networks. In this section, we visualize the loss function of the two training methods, mixup and ERM. Reference [33] proposed to project the network parameters onto the vector of random directions for visualization, but did not consider the effect of vector size on the loss function. Too small a random vector will get a huge loss value, making most of the loss value is not within the normal range, resulting in limited visualization capabilities. Furthermore, the parameters of BN are not considered in [33], and BN has scaling invariance to the network [35]. Reference [35] makes the network parameters and random vectors have the same Frobenius norm, making it possible to perturb the parameters in the BN layer of

Algorithm 1 Visualizing Loss Function

```
Require: $net, M, V, L, mom, x$ (net is a trained network. M and V are the mean and variance of the normal distribution, respectively. L is the approximate loss value. The mom is the momentum that changes the value of the variance. The x represents test samples.)
1:  $\tilde{W}_i \leftarrow \mathbf{0}$ // The $\tilde{W}_i$ are initialized to $\mathbf{0}$ tensors with the same shape as $W$.
2:  function GetLoss($V, mom$)
3:      $V \leftarrow (1 + mom)V$
4:      $T_i \leftarrow normal(mean = M, variance = V, size = W)$ //The tensor $T_i$ is initialized from a normal distribution with mean $M$, variance $V$, and size $W$. $W$ is obtained from the net parameters.
5:      $\tilde{W}_i \leftarrow T_i \cdot W$
6:      $loss \leftarrow net(\tilde{W}_i, x)$
7:      return $loss$, $V$
8:  end function
9:  for $i = 1 \rightarrow 2$ do
10:     $loss$, $V \leftarrow$ GetLoss($V$, 0)
11:     if $loss < L$ then
12:         while $loss < L$ do
13:             $loss$, $V \leftarrow$ GetLoss($V, mom$)
14:         end while
15:     else
16:         while $loss > L$ do
17:             $loss$, $V \leftarrow$ GetLoss($V, -mom$)
18:         end while
19:     end if
20: end for
21: for $\alpha = -1 \rightarrow 1$ do
22:     for $\beta = -1 \rightarrow 1$ do
23:         $loss(\alpha, \beta) = net(W + \alpha\tilde{W}_1 + \beta\tilde{W}_2, x)$
24:     end for
25: end for
26: return $loss(\alpha, \beta)$
```

the network. Unlike the above method, our method considers perturbing the parameters of the network starting from an appropriate random tensor and gradually adjusting its perturbation range. When the network's loss value reaches a predetermined size, save these random tensors. In the second stage, we use the network parameters and these random tensors for linear interpolation to get the different loss values of the network. Algorithm 1 gives the detailed steps of the process.

In order to better visualize the loss of the model trained using the mixup method, it is necessary to properly set the parameters in Algorithm 1. On the network architecture, we use trained PreAct-Resnet [27] or VGG-11 [7] models because they have different architectures. The mean M and variance V of the normal distribution in Algorithm 1 are set to 0 and 0.2, respectively. The parameter *mom* can conveniently control the magnitude and direction of the disturbance to the network parameters, which can better control the projection of the network parameters in the disturbance direction. In our experiment, the *mom* is set to 0.1.

Different optimization methods find different local minima [34]. We visualize the loss function of Resnet-like [6] and VGGNet-like [7] networks on the CIFAR-10 [16] dataset. It can be seen from Figure 3 that the loss functions of the two training methods, mixup and ERM, are very similar. However, the size of the basin near the local minimum of the mixup is much larger than that of the ERM, indicating that the loss surface near the local minima of the mixup is smoother and the ERM is relatively sharp. The more flat the decision surface, the more conducive to the network to predict the interpolation between samples. This also allows the network to have a better robustness against the attack by adversarial sample.

In addition to using the weighted noise perturbation method to visualize the loss function of mixup and ERM, we also compared the visualization method proposed in this paper with the method in [33]. Since the method proposed by [33] can only visualize the one-dimensional loss function, we extend [33] into a two-dimensional visualization method and embed it into our visualization algorithm, using random weight tensors instead of perturbing network weights. Fig. 3.g and Fig. 3.h show the results of visualizing network loss using the method in [33]. Comparing the two visualization methods, we can find that our visualization method produces a smoother contour and the loss function is relatively flat, indicating that the network weight is projected more smoothly. However, the contours produced by [33] are more rugged, which indicates that the random network weights are significantly different from the original network weights, and the projected loss values have a large disturbance. This phenomenon also shows that our visualization method can better project the loss function onto the disturbed weight tensor instead of the random weight tensor, which is more practical.

IV. SPATIAL MIXUP

A. STITCH THE IMAGE

Mixing spatial images is a very straightforward trick to use the spatial information of the image to create a new synthetic sample. Stitching the space domain of two images will have a variety of stitching options. As shown in Figure 4, we can stitch the image vertically, or stitch the image horizontally, or randomly create rectangles as the stitch area, etc. Assuming that the two images occupy the different proportions of area in the synthetic image as λ and $1-\lambda$ respectively, the proportion of their labels in the synthetic label will be set as λy_1 and $(1-\lambda)y_2$.

We assume that the image's tensor is [H,W,C], and each dimension of the tensor represents the height, width and number of channels of the image, respectively. Two randomly selected samples from the training set are denoted by (x_i, y_i) and (x_j, y_j). The random value λ $(0 \leq \lambda \leq 1)$ is taken from the Beta distribution. We use the following procedure to produce

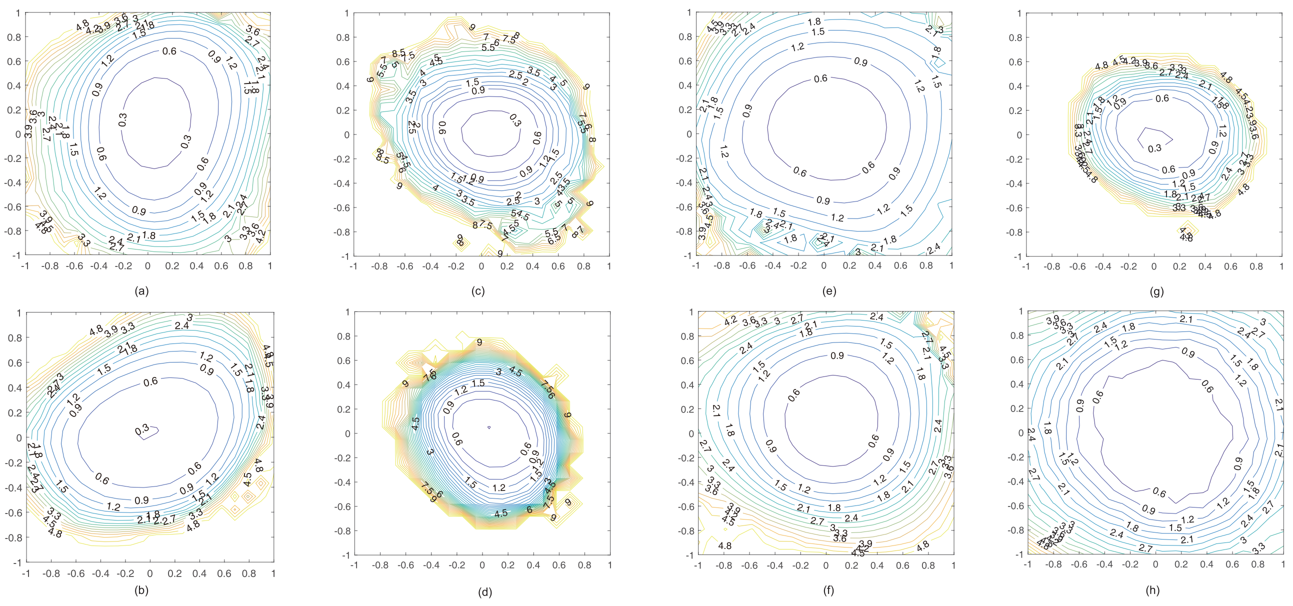


FIGURE 3. Visualize networks using different training methods and different architectures. The title of each subfigure contains the training method, network, maximum loss, and test accuracy. The subfigure (g) and (h) show the results of visualizing network loss using the method in [33]. (a) Mixup, PreAct ResNet-18, L = 5, 95.8%. (b) ERM, PreAct ResNet-18, L = 5, 94.64%. (c) Mixup, PreAct ResNet-18, L = 10, 95.8%. (d) ERM, PreAct ResNet-18, L = 10, 94.64%. (e) Mixup, VGG-11, L = 5, 92.65%. (f) ERM, VGG-11, L = 5, 91.67%. (g) Mixup, PreAct ResNet-18, L = 5, 95.8%. (h) Mixup, VGG-11, L = 5, 92.65%.

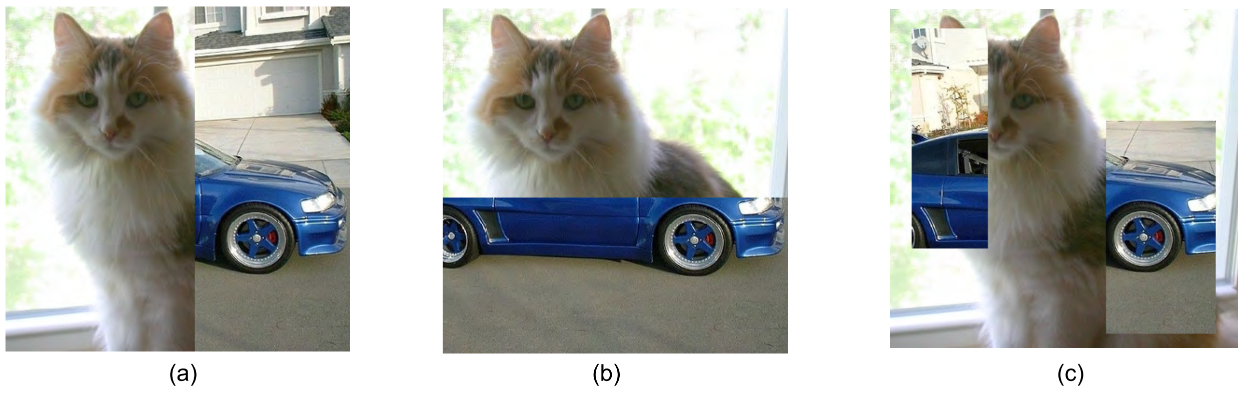


FIGURE 4. The space domain of the two images is stitched in different ways to produce different synthetic images. (a) Vertical stitching. (b) Horizontal stitching. (c) Random rectangle stitching.

a synthetic image:

$$\begin{aligned} P &= \lfloor \lambda \cdot W \rfloor \\ \tilde{x} &= cat((x_1[:, : P, :], x_2[:, P :, :]), 1) \\ \tilde{y} &= \lambda y_1 + (1 - \lambda) y_2 \end{aligned} \tag{2}$$

where *cat* represents a function that combines two tensors in one of its dimensions. The brackets in the Equation 2 represent numpy-style tensor indexes, which means that all tensor values less than P (in x_1) or greater than P (in x_2) in the second dimension will be selected.

Image stitching can provide a data augmentation because stitching different images produces different samples. In addition, it allows the filter to span two images while extracting two incomplete sample information, which gives neurons more chances to contribute to different samples for better robustness. We will record the horizontally stitched image as mixup-H and the vertically stitched image as mixup-V. For the sake of comparison, we will record the original mix of image channels as mixup-C. The new samples produced by several different mixup methods, so these methods can exist as complementary.

B. MIX MULTIPLE MIXUPS

The difference between image stitching and image channel interpolation is quite different. Instead of simply interpolating between samples, image stitching interpolates on spatial information, which is not as intuitively understood as image channel interpolation. There will be two different spatial distributions of the synthetic samples, which allow the neural network to clearly capture the interpolation information between the two category of spatial domain.

Mixing multiple mixup training methods will produce more diverse training samples. Neural networks can not only make use of synthetic samples on the channel, but also make

TABLE 1. Test error comparison of many mixup methods. The experimental results of mixup-C and ERM refer to [14].

Dataset	Model	ERM	mixup-C	mixup-H	mixup-HV	mixup-HC
CIFAR-10	PreAct ResNet-18	5.6	3.9	3.9	3.4	3.5
	DenseNet-BC-190	3.7	2.7	2.7	2.3	2.3
	ResNext29-8-64	3.7	2.7	-	2.4	-
CIFAR-100	PreAct ResNet-18	25.6	21.1	21	19.8	19.9
	ResNext29-8-64	17.4	16.8	-	14.5	-
ImageNet	ResNet-50	23.5	23.3	23.3	23	23.1
	ResNet-101	22.1	21.5	21.7	20.3	20.5
	ResNeXt-101 64×4d	20.4	19.8	19.9	19.3	19.3

use of synthetic samples in the spatial domain. This will allow neural networks to have more opportunities to observe more samples and increase their generalization performance. In addition, in the training process, using different mixed mode samples, the same label information can correspond to different mixed mode samples, which will produce better regularization effect. Conveniently, we distinguish the different mixing methods by adding the suffix. For example, we will designate mixup-HV as a training method that uses both horizontal and vertical stitching. The training method that uses both horizontal stitching and channel mixing is denoted as mixup-HC. The performance of different mixups is compared in the experiment Section V-A.

V. EXPERIMENT

A. CLASSIFICATION PERFORMANCE

We use representative benchmark datasets: CIFAR-10 [16] and CIFAR-100 [16] to evaluate the performance of our algorithm. CIFAR-10 and CIFAR-100 each contain 32 × 32-pixel color images, consisting of 50k training images and 10k testing images, respectively. In CIFAR-10, it includes 10 classes, and CIFAR-100 includes 100 classes. Channel means are computed and subtracted in preprocessing. We also apply standard augmentation [6], [30], [37]–[43]: horizontal flipping and translation by 4 pixels are adopted before using the mixup method.

When training on the CIFAR [16] dataset, the networks are trained on two Tesla k80 GPUs using stochastic gradient descent (SGD). We use a weight decay of 10^{-4} and a Nesterov momentum [44] of 0.9 without dampening. The batch size on each GPU is set to 128 for 200 epochs. The initial learning rate is set to 0.1 and is divided by 10 at 50% and 75% of the total number of training epochs. The value of α is set to 1 on this dataset.

We also conducted more studies on the ILSVRC 2012 classification dataset [45]. It contains 1.2 million images for training, 50k for verification, and a total of 1000 predefined category labels. We use the same data augmentation scheme as [14] for the training image and apply 224 × 224 center cropping to the image during testing.

The ImageNet-based models are initialized similarly to the CIFAR [16] models, but some settings are different. Training is carried out with SGD over a total of 90 training epochs with learning rate of 0.1, and learning rate decayed by factor of 0.1 after 30, 60 and 80 epochs. We train the models in multi-GPU mode with 8 Tesla k80 GPUs, splitting each mini-batch into 8 portions. The training mini-batch size is 256. These training settings are similar to those used to train ImageNet-based ResNets [6]. In order to facilitate comparison, the value of α is consistent with [14].

1) PERFORMANCE ON CIFAR-10

We use three network architectures to compare the performance of the two training methods, mixup and ERM. We used PreAct-ResNet18 [27] to compare the performance of several mixup methods on CIFAR [16]. It can be seen that mixup-C (original mixup) and mixup-H have almost the same performance, while mixup-HV and mixup-HC have relatively higher performance gains. We also experimented with DenseNet-BC-190 [46] because the network needed to preserve all the features of the layer during training, so the memory of a single 12G GPU did not support training for that network. We chose memory-optimized DenseNets [47] to train on two GPUs. The network architecture in the CIFAR [16] test error is 2.3%, achieved the state of the art result. ResNext29-8-64 [48] is a network of relatively few layers, use it to test the mixup-HV method, find its generalization error is better than mixup-C too.

2) PERFORMANCE ON IMAGENET

We performed a performance comparison of several mixup methods on the ImageNet [17] dataset. As shown in Table 1, the performance of mixup-HV and mixup-HC is higher than that of other mixup training methods, and the performance difference between mixup-H and mixup-C is not obvious. We analyzed that the reason is that mixup-HV and mixup-HC use more data augmentation than mixup-C and mixup-H, because the regularization of the loss layer caused by the mixed labels during their training is the same. This experiment fully proves that the spatial mixup training method has a relatively large performance advantage over the original mixup and ERM training methods.

B. MIXGAN

Mixup [14] has demonstrated that introducing a mixup approach into GAN will result in more stable training, as well as allowing GAN to generate the diversity of images. But it just tested the toy dataset, did not use datasets that fit the natural distribution, and used the mixup method only for the discriminators. We apply mixups to the more general problem, and we use CycleGAN [22], which uses unsupervised training to edit images for two different domains. It generates an image of the A domain to B domain using the generation network G_A while generating an image from the B domain to the A domain using the generator G_B as an inverse mapping of G_A. It uses a loss called cyclic consistency to minimize the information loss of a sample from the A domain to the B domain and then back to the A domain.

Training CycleGAN [22] requires solving the nimimax problem of generator G and discriminator D parameters. Since generators G and discriminators D are often parameterized as deep convolutional neural networks, this minimax problem is very difficult in practice. Therefore many proposals have been proposed to better train CycleGAN [22]. However, using the Mixup method to train CycleGAN [22], its loss function can be reconstructed into the following formalization:

$$
\begin{aligned}
P &= \lfloor \lambda \cdot W \rfloor \\
\tilde{x} &= cat((x_1[:, : P, :], x_2[:, P :, :]), 1) \\
&\min_G \max_D \; \mathbb{E}[L(D(\lambda\tilde{x} + (1-\lambda)G(\tilde{x})), \lambda)] \qquad (3)
\end{aligned}
$$

where *cat* and P are consistent with the representation in Equation 2. The loss function represented by Equation 3 is different from the loss function of traditional CycleGAN [22]. Using this loss function to train CycleGAN [22] can not only provide more efficient loss function, but also better regularize the generated network and discriminant network. This method make GAN [18] have more opportunity to access a wider variety of training samples, allowing GAN [18] to be trained faster and with more diverse patterns to avoid mode collapse.

We use the mixup-HV method in the generator of MixGAN and the mixup-C in the discriminator of MixGAN without changing the loss function of the generator, and only changing the discriminator's loss function into a mixup form. We reduced the training process of the original 200 epochs to 150 epochs, and use horizontally and vertically stitched images with equal probability for more data combinations. We call this method MixCycleGAN. The image generation quality of MixCycleGAN are tested on a Google Maps [36] dataset. This dataset contains 1096 training images scraped from Google Maps [36] with image size 256×256, and these data was split into train and test about the median latitude of the sampling region. Figure 5 shows the input and output images during training.

Figure 6 compares the quality of the images generated by MixCycleGAN and CycleGAN [22], showing that MixCycleGAN produces more variety and richness of image detail. But MixCycleGAN also produced some examples of failure. In the last row of Figure 6, for large green areas,

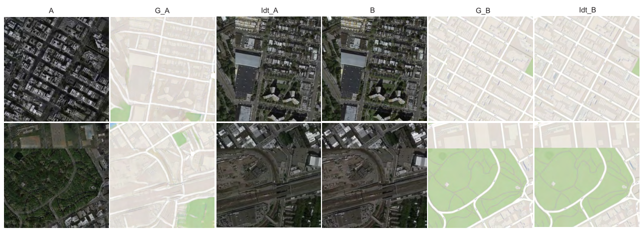


FIGURE 5. MixCycleGAN training process, the first row is vertically stitched images, the second row is horizontally stitched images. Idt_A represents an image from the A domain through the G_A to the B domain and then back to the A domain by G_B. and Idt_B has the opposite process.

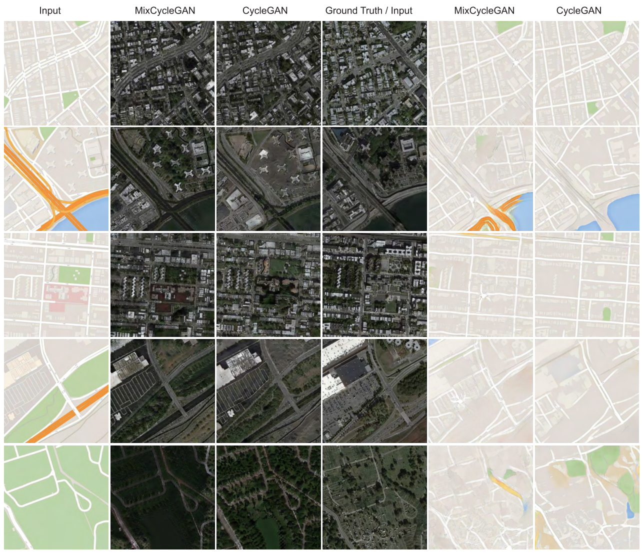


FIGURE 6. The images generated by MixCycleGAN and CycleGAN. Each row shows the results of bidirectional generation from maps to photos and from photos to maps.

MixCycleGAN did not produce all of the plants, but created the ocean. This is related to its training method. Generators trained by the mixup-HV method will generate two very dissimilar spatial domain information in a forward process, so that the generator will fill some space domain information. This is the main reason for its diversity, but it is also why it is difficult to fit large areas of similarities. MixCycleGAN tends to generate more detail, resulting in enhanced image contrast, giving different details for the same area over a large area. We can avoid this by using a small percentage of images that are not stitched in each iteration.

VI. CONCLUSION

In this paper, we first analyze the effect of linear interpolation of samples and their labels on the generalization performance of neural networks, and find that mixup is better at separating multiple categories at the same time. We propose a method to visualize the loss function of neural networks by weighting noise perturbations. By visualizing the loss function of the mixup training method, we find that the classification decision-making surface of the mixup is smoother than the ERM, which is beneficial for the network to predict the interpolated samples and to make the network more robust against attacks. Based on the conclusion that the input information of the mixup has positive correlation with the label information, we propose a training method based on the spatial image stitching, which has the same performance as the mixup method. Finally, our experiments prove that the combination of multiple mixup training methods can further improve the generalization performance of neural networks, and the mixed method has achieved the state of the art result on the CIFAR and ImageNet datasets. Applying mixup-HV and mixup-C to GAN generators and discriminators, respectively, will reduce the GAN training process and increase the diversity of generated images.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[2] G. Hinton *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.
[3] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
[4] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *Proc. ICLR*, 2016.
[5] L. J. Ba and B. Frey, "Adaptive dropout for training deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Red Hook, NY, USA: Curran Associates, 2013, pp. 3084–3092.
[6] 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.
[7] K. Simonyan and A. Zisserman. (2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556
[8] 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.
[9] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proc. 30th Int. Conf. Mach. Learn. (ICML)*, vol. 28, S. Dasgupta and D. McAllester, Eds. Atlanta, GA, USA: PMLR, no. 3. May 2013, pp. 1058–1066.
[10] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015.
[11] M. D. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," in *Proc. ICLR*, 2013.
[12] L. Xie, J. Wang, Z. Wei, M. Wang, and Q. Tian, "DisturbLabel: Regularizing CNN on the loss layer," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR),*, Jun. 2016, pp. 4753–4762.
[13] H. Inoue. (2018). "Data augmentation by pairing samples for images classification." [Online]. Available: https://arxiv.org/abs/1801.02929
[14] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," in *Proc. ICLR*, 2018.
[15] V. N. Vapnik, *Statistical Learning Theory*. Hoboken, NJ, USA: Wiley, 1998.
[16] A. Krizhevsky, "Learning multiple layers of features from tiny images," M.S. thesis, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, USA, 2009.
[17] 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. (CVPR)*, Jun. 2009, pp. 248–255.
[18] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Red Hook, NY, USA: Curran Associates, 2014, pp. 2672–2680.
[19] A. Radford, L. Metz, and S. Chintala. (2015). "Unsupervised representation learning with deep convolutional generative adversarial networks." [Online]. Available: https://arxiv.org/abs/1511.06434
[20] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2016, pp. 2234–2242.
[21] I. Goodfellow *et al.*, "Generative adversarial networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Red Hook, NY, USA: Curran Associates, 2014, pp. 2672–2680.
[22] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. (2017). "Unpaired image-to-image translation using cycle-consistent adversarial networks." [Online]. Available: https://arxiv.org/abs/1703.10593
[23] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. Image Process.*, vol. 15, no. 12, pp. 3736–3745, Dec. 2006.
[24] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in *Proc. 27th Annu. Conf. Comput. Graph. Interact. Techn.*, New York, NY, USA, 2000, pp. 417–424.
[25] Q. Luan, F. Wen, D. Cohen-Or, L. Liang, Y.-Q. Xu, and H.-Y. Shum, "Natural image colorization," in *Proc. 18th Eurographics Conf. Rendering Techn.*, Aire-la-Ville, Switzerland, 2007, pp. 309–320.
[26] K. Nasrollahi and T. B. Moeslund, "Super-resolution: A comprehensive survey," *Mach. Vis. Appl.*, vol. 25, no. 6, pp. 1423–1468, 2014.
[27] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Computer Vision—ECCV*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham, Switzerland: Springer, 2016, pp. 630–645.
[28] G. Kang, X. Dong, L. Zheng, and Y. Yang. (2017). "PatchShuffle regularization." [Online]. Available: https://arxiv.org/abs/1707.07103
[29] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang. (2017). "Random erasing data augmentation." [Online]. Available: https://arxiv.org/abs/1708.04896
[30] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *Computer Vision—ECCV*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham, Switzerland: Springer, 2016, pp. 646–661.
[31] S. Singh, D. Hoiem, and D. Forsyth, "Swapout: Learning an ensemble of deep architectures," in *Proc. Adv. Neural Inf. Process. Syst.*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2016, pp. 28–36.
[32] X. Bouthillier, K. Konda, P. Vincent, and R. Memisevic. (2015). "Dropout as data augmentation." [Online]. Available: https://arxiv.org/abs/1506.08700
[33] I. Goodfellow, O. Vinyals, and A. M. Saxe, "Qualitatively characterizing neural network optimization problems," in *Proc. ICLR*, 2015.
[34] D. J. Im, M. Tao, and K. Branson. (2016). "An empirical analysis of the optimization of deep network loss surfaces." [Online]. Available: https://arxiv.org/abs/1612.04010
[35] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein. (2017). "Visualizing the loss landscape of neural nets." [Online]. Available: https://arxiv.org/abs/1712.09913
[36] P. Isola, J. Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5967–5976, doi: 10.1109/CVPR.2017.632.
[37] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. (2014). "Deeply-supervised nets." [Online]. Available: https://arxiv.org/abs/1409.5185
[38] M. Lin, Q. Chen, and S. Yan, "Network in network," in *Proc. ICLR*, 2014.
[39] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," in *Proc. ICLR*, 2015.
[40] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. (2014). "Striving for simplicity: The all convolutional net." [Online]. Available: https://arxiv.org/abs/1412.6806
[41] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2015, pp. 2377–2385.
[42] D. Liang, F. Yang, T. Zhang, J. Tian, and P. Yang, "WPNets and PWNets: From the perspective of channel fusion," *IEEE Access*, vol. 6, pp. 34226–34236, 2018.
[43] G. Larsson, M. Maire, and G. Shakhnarovich, "FractalNet: Ultra-deep neural networks without residuals," in *Proc. ICLR*, 2017.
[44] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1139–1147.

[45] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[46] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2017, pp. 2261–2269.

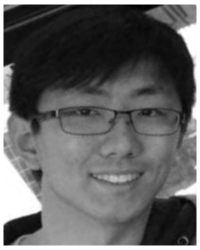
[47] G. Pleiss, D. Chen, G. Huang, T. Li, L. van der Maaten, and K. Q. Weinberger. (2017). "Memory-efficient implementation of densenets." [Online]. Available: https://arxiv.org/abs/1707.06990

[48] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5987–5995.

DAOJUN LIANG received the B.S. degree in computer science from TaiShan University, China, in 2016. He is currently pursuing the master's degree with the School of Information Science and Engineering, Shandong Normal University. His research interests include deep learning and computer vision.

FENG YANG received the B.S. and the M.S. degrees in electronics and communications from Shandong University in 1985 and 1988, respectively. He is currently a Professor with the School of Information Science and Engineering, Shandong Normal University, where he is also the Director of the Department of Communication Engineering. He has participated in one national fund project. He has published over 30 papers and written 5 books. He holds seven national invention patents and four utility model patents. He presided over five provincial and university-level education reform projects. He was a recipient of the Provincial-Level Teaching Achievement Award and the School-Level Teaching Achievement Award. He was a recipient of three provincial awards for scientific and technological progress.

TIAN ZHANG received the B.S. and M.S. degrees and the Ph.D. degree from Shandong Normal University, Jinan, China, in 2006, 2009, and 2014, respectively . He was also a Visiting Ph.D. student with Tsinghua University from 2010 to 2013. He has been with Shandong Normal University since 2014, where he is currently an Associate Professor. His research interests include wireless communications and smart grid. He served as a TPC Member of the IEEE GLOBECOM 2017. He was a recipient of the 2010 Science and Technology Progress Award of Shandong Province (2nd class), the 2015 Excellent Doctoral Dissertation Award of Shandong University, and the 2016 Shandong Province Higher Educational Science and Technology Award (3rd class).



PETER YANG received the B.S. degree from Shandong University, Jinan, China, in 2013, and the M.S. degree from the University of California at Irvine, Irvine, USA, in 2015. He is currently a Software Engineer with Amazon, where he is responsible for Amazon Web services. His research interests include machine learning, software engineering, and data analysis.