Journal Pre-proof

Improving the generalization performance of deep networks by dual pattern learning with adversarial adaptation

Haimin Zhang, Min Xu

Please cite this article as: H. Zhang and M. Xu, Improving the generalization performance of deep networks by dual pattern learning with adversarial adaptation, *Knowledge-Based Systems* (2020), doi: https://doi.org/10.1016/j.knosys.2020.106016.

This is a PDF file of an article that has undergone enhancements after acceptance, such as the addition of a cover page and metadata, and formatting for readability, but it is not yet the definitive version of record. This version will undergo additional copyediting, typesetting and review before it is published in its final form, but we are providing this version to give early visibility of the article. Please note that, during the production process, errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.
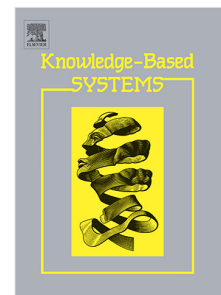
Improving the Generalization Performance of Deep Networks by Dual Pattern Learning with Adversarial Adaptation

Haimin Zhang[a], Min Xu[a,*]

[a]School of Electrical and Data Engineering, University of Technology Sydney
81 Broadway, Ultimo, NSW 2007, Australia

Abstract

In this paper, we present a dual pattern learning network architecture with adversarial adaptation (DPLAANet). Unlike conventional networks, the proposed network has two input branches and two loss functions. This architecture forces the network to learn robust features by analyzing dual inputs. The dual input structure allows the network to have a considerably large number of image pairs, which can help address the overfitting issue due to limited training data. In addition, we propose to associate the two input branches with two random interest values during training. As a stochastic regularization technique, this method can improve generalization performance. Moreover, we introduce to use the adversarial training approach to reduce the domain difference between fused image features and single image features. Extensive experiments on CIFAR-10, CIFAR-100, FI-8, the Google commands dataset, and MNIST demonstrate that our DPLAANets exhibit better performance than benchmark networks. The experimental results on subsets of CIFAR-10, CIFAR-100, and MNIST demonstrate that DPLAANets have good generalization performance on small datasets. The propose architecture can be easily extended to have more than two input branches. The experimental results on subsets of MNIST show that the architecture with three branches outperforms two branches when training set is extremely small.

Keywords: Image classification, deep neural networks, domain adaptation

1. Introduction

Convolutional neural networks (CNNs) have become a dominant approach for various machine learning applications such as computer vision [1, 2, 3, 4, 5], natural language processing [6, 7, 8], and reinforcement learning [9, 10]. Integrating feature learning and classifiers in an end-to-end manner, deep neural networks can learn features automatically from training data without human involvement during training. It has been shown that features learned by CNNs are much discriminative compared to hand-crafted features [1], and that features extracted from a CNN pretrained on a large scale image dataset can be transferred to other visual recognition tasks [11, 12].

Researchers have spent much effort in designing network architectures to improve the performance of CNNs. He *et al.* [2] proposed the residual learning framework. This framework eases the training of deep neural networks, and enables them to be considerably deep. Residual networks (ResNets) have led to performance improvement in both visual and non-visual tasks. Huang *et al.* [13] proposed densely connected networks (DenseNets), in which each layer is connected to every other layer in a feed-forward fashion. This architecture substantially reduces the number of parameters, and is highly computationally efficient as a result of feature reuse.

State-of-the-art deep neural networks usually consist of many layers with a large number of parameters. The ResNeXt-19 ($8 \times 64d$) [14] contains approximately 3×10^7 parameters to model the 5×10^4 images in CIFAR-10 [15]. The VGGNet-16 [16] has around 10^8 parameters to model the 10^6 images in ImageNet [17]. The large number of parameters makes deep networks prone to overfitting; therefore, training deep networks requires huge amounts of data. However, collecting data and labelling them are laborious work, especially when domain experts are necessary to distinguish between fine-grained visual categories. For some tasks, it is extremely difficult to collect samples.

In this work, we focus on efficient feature learning with limited data using CNNs. We observe that humans can learn knowledge from two given images by analyzing and comparing two images. An illustration is shown in Figure 1. Inspired by this observation, we propose a dual pattern learning (DPL) network architecture, referred to as DPLNet in this paper. Unlike previous deep networks, the DPLNet is trained using image pairs. The number of image pairs can be significantly large even for small datasets. This can help to address the overfitting issue due to lack of training data.

As shown in Figure 2, we design two input branches and two loss functions so that the DPLNet can perform dual pattern learning. Two input images are processed

---

*Corresponding author
Email addresses: `Haimin.Zhang@student.uts.edu.au` (Haimin Zhang), `Min.Xu@uts.edu.au` (Min Xu)
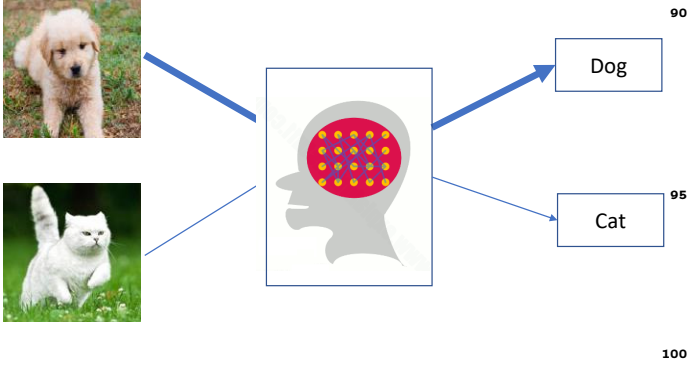
Figure 1: An illustration that shows humans learn knowledge by analyzing dual images. They may have more interest in learning one image than the other image. In this figure, the human is more interested in, or pays more attention to, learning dog (boldness of lines represents interest value).

by the two input branches in parallel, and feature maps generated by the two branches are then fused together to backbone network. Analyzing dual inputs simultaneously, this architecture is able to learn robust features. In addition, we observe that people may have more interest in one image than the other image when given two images to learn. This might be due to reasons such as personal preference and/or prior knowledge. Inspired by this observation, we propose to associate the two input branches with two random interest values for learning corresponding images during training. This method can help improve the generalization performance.

Moreover, we introduce to use the adversarial training framework [18] to minimize the domain difference between fused image features and single image features. We refer to the DPLNet with adversarial adaptation as DPLAANet in this paper. State-of-the-art deep networks can be easily adapted to DPLAANets. We show that our DPLAANets exhibit better performance.

There has been research which uses the mixture of two images as for training deep neural networks, such as Mixup [19] and between class (BC) learning [20]. Unlike their work, we propose a new architecture with dual input branches which are trained using image pairs. Moreover, we incorporate the adversarial training framework to reduce the domain difference between fused image features and single image features.

This paper provides the following three contributions:

- We propose the DPLAANet architecture towards learning with limited data. The dual input structure of the DPLAANet enables the network to learn robust features by analyzing dual inputs simultaneously. With the dual input structure, we have a large number image pairs to train the network. This helps to address the overfitting issue due to lack of training data.

- The adversarial training framework is incorporated to reduce the domain difference between fused image features and single image features. We show that

this method effectively contributes to performance improvement.

- The DPLAANets are evaluated on five benchmark datasets, i.e., CIFAR-10, CIFAR-100, FI-8, Google commands dataset, and MNIST, wherein they lead to performance improvement compared to benchmark networks. The experimental results on subsets of CIFAR-10, CIFAR-10, and MNIST demonstrate that our DPLNets have outstanding generalization performance on small datasets. Extended experiments on subsets of MNIST show that the architecture with three branches outperforms two branches when training set is extremely small.

The rest of the paper is organized as follows. Section 2 reviews related work on deep networks. The proposed DPLAANet architecture is introduced in section 3. Experimental results are presented and discussed in section 4. Finally, we conclude this paper in section 5.

2. Related Work

2.1. Deep Neural Networks

Neocognitron [21], which was proposed by Fukushima et al., in 1980, is the first type of CNNs. It is a hierarchical multi-layered network inspired by neurophysiological findings on the visual systems of mammals. Later, LeCun et al., [22] proposed a CNN architecture called LeNet-5 for handwritten digit recognition in 1998. While CNNs have a long history, the success of CNNs has been achieved only recently thanks to the availability of large scale visual datasets and powerful GPU computing resources. In 2012, Krizhevsky et al. [1] proposed the AlexNet architecture, which consists of five convolutional layers and three fully connected layers. This is the first CNN model proposed for large scale image classification. The AlexNet achieved superior performance compared to hand-crafted features.

Since the introduction of AlexNet, many CNN architectures have been developed to improve performance. These studies include exploring increasing the depth (the number of layer) and increasing the width (the number of channels in each layer) of CNNs. For example, Simonyan et al. [16] investigated the effect of the network depth on its accuracy, and proposed VGGNets with 16 and 19 layers. In [2], He et al. introduced identity shortcut connections, and proposed the ResNet architecture. This architecture makes very deep networks easy to optimize. ResNets have achieved performance improvement for many tasks. In [13], Huang et al. explored increasing the width of networks and proposed densely connected networks (DenseNets). For each layer in DenseNets, the feature-maps of all preceding layers are used as inputs. The DenseNet architecture encourages feature reuse, and can considerably reduce the number of parameters. In addition to exploring increasing depth and width, Xie et al. [14] researched increasing the cardinality of CNNs, which refers to the size of the set of

2

transformations, and proposed the ResNeXt architecture. The ResNeXt is constructed by repeating a building block that aggregates a set of transformations with the same topology. They showed that increasing cardinality is effective to improve performance compared to increasing the depth or the width of CNNs.

Deep neural networks usually contain a large number of parameters; therefore, training deep neural networks requires huge amounts of data to reduce overfitting. To reduce overfitting on training data, label-preserving transformations [1] are often applied to enlarge training data. Commonly used data augmentation methods include random crop, color jittering, horizontal or vertical flip of images. Recently, researcher started to use generative adversarial networks to generate adversarial samples for data augmentation. For example, Zheng et al. [23] proposed to use adversarial samples to improve person re-identification baselines. Xie et al. [24] proposed to employ adversarial samples for semantic segmentation and object detection.

In addition to network development and data augmentation methods, researchers have developed regularization techniques to improve the generalization performance. For example, Srivastava et al. [25] proposed a method referred to as dropout. The key idea of dropout is to randomly drop units of the neural network during training phase. The neurons which are dropped out in this way do not contribute to the forward pass and do not participate in back-propagation. This technique forces networks to learn more robust features that are useful in conjunction with many different random subsets of the other neurons. Ioffe et al. [26] proposed the batch normalization method. This method draws its strength from making normalization a part of the model architecture and performing the normalization for each training mini-batch. With Batch normalization, we can use much higher learning rates and be less careful about parameters initialization to train deep networks.

2.2. Domain adaptation

Domain adaptation methods attempt to transfer the knowledge obtained from source domain to target domain. Over the last few years, adversarial approaches have been explored to domain adaptation [27, 28]. The adversarial approaches are based on generative adversarial networks (GANs) [18]. A typical GAN framework contains two models: a generative model G that captures the data distribution, and a discriminative model D that learns to determine whether a sample is from the training data or the generator G. In [27], Tzeng proposed the adversarial discriminative domain adaptation (ADDA) framework, which attempts to learn a discriminative mapping of target images to the source feature space by fooling a domain discriminator. The ADDA method achieved good performance on cross-domain digit classification and cross-modality object classification tasks. In [29], Zhang proposed collaborative and adversarial networks (CAN) for unsupervised domain adaptation. In the CAN model, each CNN block is connected to a

domain classifier. Using this method, CAN models can learn domain informative representations at lower blocks by collaborative learning and learn domain uninformative representations from higher blocks by adversarial learning.

3. Methodology

3.1. Dual pattern learning

Empirical risk minimization (ERM) [30] has been the rule for training neural networks. Given a training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ with N labelled training data, where x_i and y_i represent the i-th data and its corresponding label, respectively. Under the ERM rule, a neural network $f(\boldsymbol{x}; \theta_f)$ with parameters θ_f is trained by minimizing the following risk:

$$R(f) = \frac{1}{N} \sum_{i=1}^{N} \ell(f(\boldsymbol{x}_i), y_i), \qquad (1)$$

where f takes a single image as input and ℓ is a loss function for penalizing errors between prediction result and actual target.

In this paper, we propose a dual pattern learning architecture. This architecture intends to predict for dual inputs simultaneously. Unlike f for empirical risk minimization, the prediction function $g(\boldsymbol{x}, \boldsymbol{x}'; \theta_g)$ with parameters θ_g for dual pattern learning takes two samples as input, and is obtained by minimizing dual prediction risk, which is given below:

$$R_{DPL}(f) = \frac{1}{N} \sum_{i} \ell(g(\boldsymbol{x}_i^1, \boldsymbol{x}_i^2), y_i^1, y_i^2), \qquad (2)$$

where the loss function ℓ is to be defined to penalize dual prediction errors. We refer to training neural networks by minimizing Equation (2) as empirical dual prediction risk minimization.

An overview of the proposed DPLNet architecture is shown in Figure 2. The DPLNet contains several blocks, each of which consists of a number of convolutional layers. Feature maps generated within the same block have the same height and width. We design two input branches and use a loss function to penalize dual prediction errors. Two input images are processed by the two input branches in parallel, and feature maps generated by the two input branches are then fused together to backbone network which ends with a fully connected layer with softmax. The two input branches share the same parameters. This guarantees that the two input branches generate consistent feature maps for dual inputs, which means that feature maps fused to backbone network are the same regardless of input orders. The DPLNet architecture forces the network to learn discriminative features by analyzing dual inputs simultaneously; therefore, DPLNets encourage learned features to have large inter-class margins compared to conventional networks.

We use random weighted combination of feature maps generated by the two input branches as input to backbone
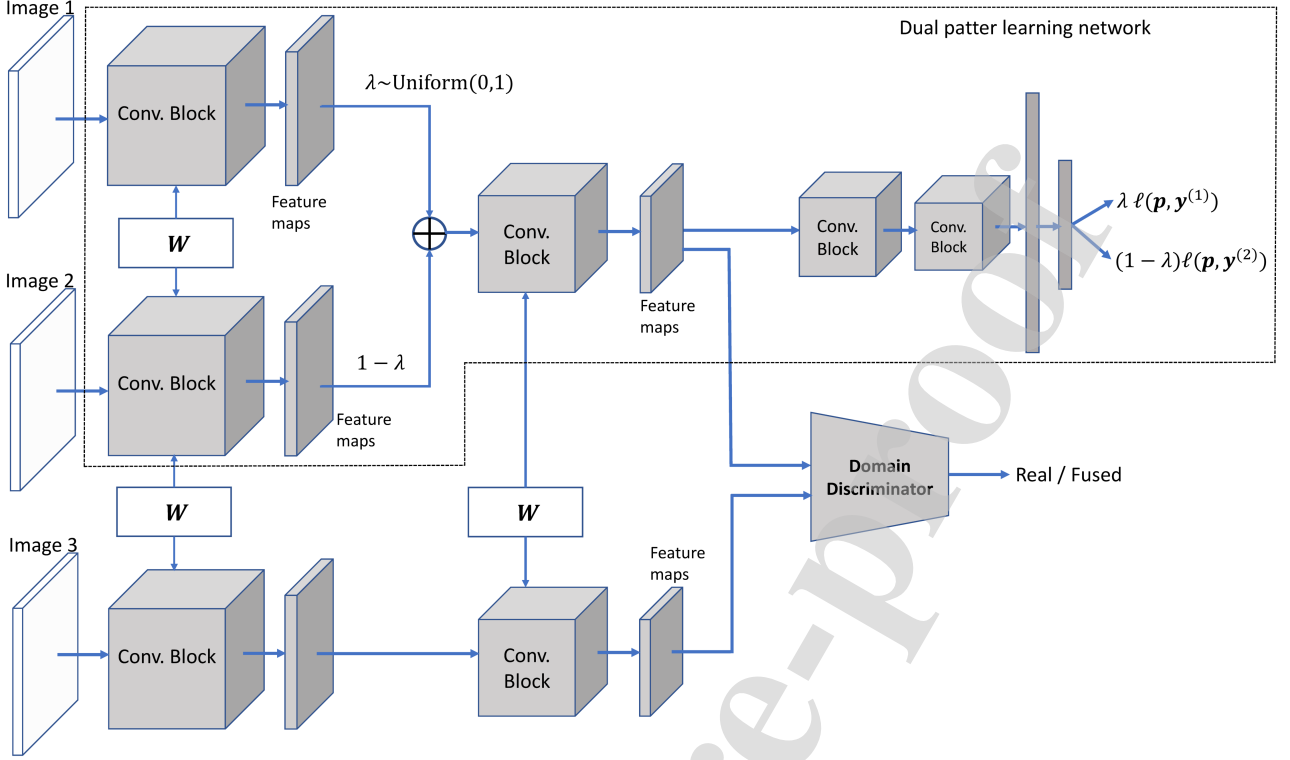
3

Figure 2: An illustration of the proposed DPLAANet framework. This framework consists of a DPLNet and an adversarial adaptation module. The DPLNet has two input branches which share the same parameters. Feature maps generated by the two input branches are fused together to backbone network. We perform random weighted fusion. A value λ is sampled from the standard uniform distribution as weight for one branch, and $1 - \lambda$ for the other branch. The weight associated with each branch can be considered as an interest value for learning the corresponding image. A third branch is introduced for adaptation, and a domain discriminator is defined to distinguish training data distribution from testing data distribution at feature level.

network during training. In particular, we randomly sample a value λ from the standard uniform distribution, as given below:

$$\lambda \sim Uniform(0, 1). \tag{3}$$

The value of λ is used as weight for one branch, and $1 - \lambda$ for the other branch. The two weights can be considered as interest values for learning corresponding input. When λ is close to 0 or 1, the fused feature maps come primarily from one branch. In this case, the network is close to using a single input branch like conventional deep networks. In our experiments, we clamp λ into range $[0.2, 0.8]$. This achieves a little bit better performance than without using clamping, because the effectiveness of the dual pattern learning approach for performance improvement. The fused feature maps is represented as the convex combination of the two sets of feature maps:

$$\mathbf{Conv} = \lambda \mathbf{Conv_1} + (1 - \lambda)\mathbf{Conv_2}, \tag{4}$$

where $\mathbf{Conv_1}$ and $\mathbf{Conv_2}$ represent feature maps generated by the two branches, respectively. The fused feature maps $\mathbf{Conv}$ are taken as input to a backbone network which is followed by a global average pooling layer and ends with a softmax activation layer.

The overall loss function of the DPLNet is defined to penalize dual prediction errors, which is given as follows:

$$L_{DPL} = \lambda \ell_{cls}(\boldsymbol{p}, \boldsymbol{y}^{(1)}) + (1 - \lambda)\ell_{cls}(\boldsymbol{p}, \boldsymbol{y}^{(2)})$$
$$= -\lambda \sum_{i=1}^{C} y_i^{(1)} log(p_i) - (1 - \lambda) \sum_{i=1}^{C} y_i^{(2)} log(p_i), \tag{5}$$

where λ is the same as in Equation (4), C represents the number of output categories, $\boldsymbol{p} = (p_1, ..., p_C) \in \mathbb{R}^C$ denotes the predicted probability distribution produced by the softmax layer, and $\boldsymbol{y}^{(1)}$ and $\boldsymbol{y}^{(2)}$ are one-hot encoding labels for images given as input to the first branch and the second branch, respectively. As shown in Equation 5, the cross entropy loss is used as classification loss ℓ_{cls}. Let the output vector before the softmax layer be denoted $\boldsymbol{a} = (a_1, ..., a_C)$, probability p_i can be represented as follows:

$$p_i = \frac{exp(a_i)}{\sum_{j=1}^{C} exp(a_j)}, i = 1, ...C. \tag{6}$$

If $0.5 < \lambda \le 0.8$, the DPLNet is more interested in learning the corresponding image than the other image, and it receives more supervision for learning this image. In this case, the other image can be seen as an auxiliary image for learning. If λ is equal to 0.5, the DPLNet has equal
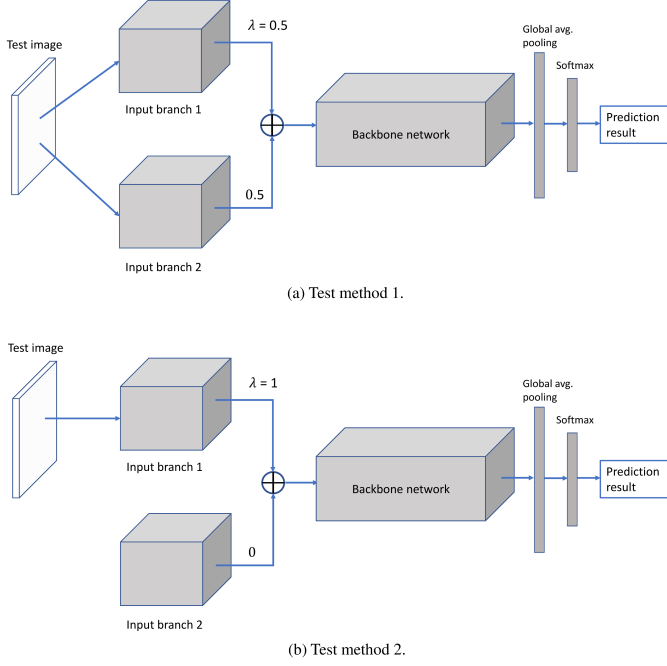
4

(a) Test method 1.

(b) Test method 2.

Figure 3: Two test approaches at test time: (a) Pass a test image to both input branches and set λ to 0.5; (b) Give an image as input to one branch and set corresponding λ to 1 while ignoring the other input branch.

interest in learning two input images. If $0.2 \leq \lambda < 0.5$, the DPLNet is more interested in learning the other input images.

At test time, there are two approaches to test an image (see Figure 3). We do not use random image pairs for testing as our goal is to use the trained network for single image classification. The first approach is to pass the test image to the two input branches while setting λ to 0.5. The other approach is to pass the image to one input branch and set the corresponding λ to 1 while ignoring the other input branch. The final softmax layer produces a probability distribution over all categories. Because the parameters of the two input branches are tied and feature maps generated by the two branches are fused by convex combination, the fused feature maps are the same for the two test approaches. Therefore, the two test approaches produces the same prediction result. In our experiments, we use the second approach for testing for time efficiency.

3.2. Adversarial domain adaptation

Domain adaptation aims to address the domain shift issue, *i.e.,* training data and testing data have different distributions in feature space. In the dual pattern learning framework, models are trained using random image pairs. We wish to use trained models for single image classification. As introduced in section 3.1, our test approach equivalently uses a test image and its duplicate as a pair for testing. We see that the distribution of training data is different from that of testing data. In this work, we propose to add

a domain adaptation module to the DPLNet to reduce the domain difference between training data and testing data.

We employ the adversarial learning method for domain adaptation. As shown in Figure 2, we add a third branch, and define a discriminator D to distinguish training data distribution from testing data distribution. As with in the GAN framework, the discriminator $D(\boldsymbol{h}; \theta_d)$ is a multilayer neural network with parameters θ_d which outputs a single scalar. We train D to maximize the probability of assigning the correct label to to both feature maps from the DPLNet and feature maps produced by the third branch. The adaptation is performed at feature level [31] instead of at pixel-level [32]. We refer to the DPLNet with adversarial adaptation as DPLAANet in this paper. We update the discriminator and the DPLNet by alternating two steps. Given a batch of image pairs $\boldsymbol{x} = \{x_i^1, x_i^2\}_{i=1}^{B}$ and a batch of images $\boldsymbol{t} = \{t_i\}_{i=1}^{B}$. Let the feature maps for the image pairs and the feature maps for the real images be denoted $\boldsymbol{z}_{x_i}$ and $\boldsymbol{z}_{t_i}$, respectively. At the first step, the two sets of feature are taken as input to the domain discriminator D, and we update the parameters of the domain discriminator D using the following hinge loss [33]:

$$
\begin{aligned}
L_D = \sum_i min(0, -1 + D(\boldsymbol{z}_{x_i})) \\
+ \sum_j min(0, -1 - D(\boldsymbol{z}_{t_j})).
\end{aligned}
\tag{7}
$$

At the second step, we fix D and update the parameters of the dual pattern learning network using the following loss:

$$
L_C = L_{DPL} - \beta \sum_i D(\boldsymbol{z}_{x_i}),
\tag{8}
$$

where L_{DPL} is the dual prediction loss (see Equation 5) and β is the weight for adjusting the losses. The spectral normalization method [33] is used to normalize the layers of D. This guarantees the discriminator satisfies the 1-Lipschitz constraint.

Unlike convectional network architectures, the proposed architecture has two characteristics: (1) With the dual input structure, we can have a considerably large number of image pairs to train the network, which can help address the overfitting issue due to limited training data. Our networks are suitable for tasks wherein training data are difficult to collect. (2) A adversarial training module is introduced to reduce the domain different between training data and testing data. The two characteristics make the proposed architecture perform better than conventional networks.

4. Experiments

We conducted experiments on a diverse of recognition tasks to show that the proposed framework is a general technique to improve the performance of deep networks. We further evaluated DPLAANets on small datasets to show their usefulness when training dataset is small.

Model	Error rate (%)	
	CIFAR-10	CIFAR-100
ResNet-18	4.96	22.78
ResNet-18 + DPLAANet	**4.45**	**19.95**
ResNet-34	4.86	21.64
ResNet-34 + DPLAANet	**4.10**	**19.28**
ResNet-50	4.62	21.89
ResNet-50 + DPLAANet	**3.96**	**18.75**
ResNet-101	4.44	20.81
ResNet-101 + DPLAANet	**3.81**	**18.32**

Table 1: Test errors (%) on CIFAR-10 and CIFAR-100.

Model	Error rate (%)	
	CIFAR-10	CIFAR-100
PreAct ResNet-18	4.90	22.48
PreAct ResNet-18 + DPLAANet	**4.02**	**19.44**
PreAct ResNet-34	4.69	21.08
PreAct ResNet-34 + DPLAANet	**3.97**	**18.97**
PreAct ResNet-50	4.52	20.60
PreAct ResNet-50 + DPLAANet	**3.88**	**18.53**
PreAct ResNet-101	4.38	20.51
PreAct ResNet-101 + DPLAANet	**3.74**	**18.02**

Table 2: Test errors (%) on CIFAR-10 and CIFAR-100.

Model	Error rate (%)	
	CIFAR-10	CIFAR-100
DenseNet-121 ($k = 32$)	4.55	22.0
DenseNet121 + DPLAANet	**4.21**	**18.75**
DenseNet-169 ($k = 32$)	4.46	20.21
DenseNet-169 + DPLAANet	**4.14**	**18.05**

Table 3: Test errors (%) on CIFAR-10 and CIFAR-100. k indicates the growth rate of network.

Model	Error rate (%)	
	CIFAR-10	CIFAR-100
ResNeXt-29 $8 \times 64d$ [14]	3.65	17.77
ResNeXt-29 $8 \times 64d$ + DPLAANet	**3.27**	**16.93**
ResNeXt-29 $16 \times 64d$ [14]	3.58	17.31
ResNeXt-29 $16 \times 64d$ + DPLAANet	**3.08**	**16.66**

Table 4: Test errors (%) on CIFAR-10 and CIFAR-100.

2e-4. At test time, we only evaluated the original 32×32 image. The value of λ was chosen from 0.01, 0.001, 0.0001 and 0.00001.

4.1. CIFAR-10 and CIFAR-100

We first conducted experiments on CIFAR-10 and CIFAR-100 [15]. The CIFAR-10 and CIFAR-100 datasets consist of 32×32 color images categorized into 10 and 100 classes, respectively. The training and testing sets contain 50,000 and 10,000 images, respectively.

We used PyTorch [34] for implementation. We implemented DPLAANets based on four state-of-the-art deep networks, *i.e.,* ResNets, pre-activation ResNets (PreAct ResNets) [35], DenseNets and ResNeXts. The classification accuracies achieved by the vanilla ResNets are used as baselines for comparison. Each input branch of our DPLAANets consists of one block, and the backbone networks consist of two/three blocks. Following [2, 35, 13], we applied zero-padding of four pixels to training images for training DPLAANets based on ResNets, PreAct ResNets, and DenseNets. Following [14], we applied zero-padding of eight pixels to training images for training DPLAANets based on ResNeXts. A 32×32 image was randomly cropped from the padded image or its horizontal flip as input data to train the models. Each channel of input data were normalized to have zero mean and unit variance. We did not use dropout, following the practice in [26]. All models were trained from scratch using SGD for 300 epochs with a mini-batch of 128/64 examples. The learning rate for training DPLNets started from 0.1 and was divided by 10 at epoch 150 and 225. The values of weight decay and momentum were set to 0.0005 and 0.9, respectively. We used the Adam [36] algorithm to train the discriminator. The learning rate for training discriminators was set to

4.1.1. DPLAANets based on ReseNets

We implemented DPLAANets based on four ResNet architectures, *i.e.,* ResNet-18, ResNet-34, ResNet-50, and ResNet-101. The experimental results are shown in Table 1. From this table, we observe that as with ResNets, the performance of our DPLAANets improves as the number of layers increases. The DPLAANet based on ResNet-101 achieves the best performance on the two datasets, whereby it achieves 3.81% and 18.32% error rates, respectively. By using the proposed framework, performance improves for the four ResNet architectures, with at least 0.51% and 2.49% performance improvements on the two datasets, respectively. On average, DPLAANets achieve 0.64% and 2.71% performance gains on the two datasets, respectively. The performance improvements yielded by using DPL with adversarial adaptation are higher on CIFAR-10 than on CIFAR-100.

4.1.2. DPLAANets based on PreAct ReseNets

We investigated DPLAANets based on four PreAct ResNets architectures, *i.e.,* PreAct ResNet-18, PreAct ResNet-34, PreAct ResNet-50, and PreAct ResNet-101. The experimental results are shown in Table 2. From this table, we find that using the proposed method helps to improve classification accuracy for the four PreAct ResNet architectures. The DPLAANets based on PreAct ResNet-101 achieve the best performance on the two datasets. The DPLNets exhibit better performance based on PreAct ResNet than based on ResNet.

4.1.3. DPLAANets based on DenseNets

We implemented DPLAANets based on two DenseNet architectures, *i.e.,* DenseNet-121 and DenseNet-169. The

Model	Num. of input branches	Error rate (%)	
		CIFAR-10	CIFAR-100
ResNet-18 + DPLAANet	1 (vanilla)	4.96	22.78
	2	**4.45**	**19.95**
	3	4.71	19.98
ResNet-34 + DPLAANet	1 (vanilla)	4.86	21.64
	2	**4.10**	**19.28**
	3	4.16	19.35

Table 6: Impact of number of input branches on performance. We did not use adversarial adaptation for branch number equal to 1.

Base model	Method	Error rate (%)	
		CIFAR-10	CIFAR-100
ResNet-18	Vanilla	4.96	22.78
	DPLNet	4.63	20.70
	DPLADNet	4.45	19.95
ResNet-34	Vanilla	4.86	21.64
	DPLNet	4.35	19.83
	DPLAANet	4.10	19.28

Table 5: Ablation study. Performance comparison among DPLAANets, DPLNets, and vanilla ResNets on CIFAR-10 and CIFAR-100.

experimental results are shown in Table 3. We see from Table 3 that our DPLAANets achieve better performance than vanilla DenseNets. On average, the DPLAANets achieve 0.33% and 2.71% performance improvements on the two datasets, respectively.

4.1.4. DPLAANets based on ResNeXts

We investigated DPLNets based on two ResNeXt architectures, *i.e.*, ResNeXt-29 (8×64d) and ResNeXt-29 (16×64d). The comparison of results between our DPLAANets and vanilla ResNeXts is show in Table 4. From this table, we observe that by using dual pattern learning with adversarial adaptation, performance improves compared to original ResNeXts. The DPLAANet based on ResNeXt-29 (16×64d) achieves the best performance on the two datasets, wherein it achieves 3.08% and 16.66% error rates, respectively.

We have seen that the proposed DPLAANet architecture helps to improve performance based on the four types of deep network architectures, *i.e.*, ResNets, PreAct ResNets, DenseNets, and ResNeXts. This demonstrates that the performance of the DPLAANet framework is stable. We observe that the performance improvements achieved by DPLAANets are more significant on CIFAR-100 than on CIFAR-10. In CIFAR-10 and CIFAR-100, each category has 5000 and 500 samples, respectively. The results show that DPLNets are very helpful for small training sets.

We conducted an ablation study to understand how each module of the proposed method contributes performance improvement. We trained DPLNets based on ResNet-18 and ResNet-34. The performance comparison among ResNets, DPLNets, and DPLAANets are shown in Table 5. We observe that the performance improves 0.42% and 1.95% on the two datasets, respectively, on average by using dual pattern learning. The adversarial adaptation module further yields 0.22% and 0.65% improvements on average on on the two datasets, respectively. The test errors evolutions on CIFAR-10 and CIFAR-100 for ResNets, DPLNets, and DPLAANets are shown in Figure 4. We further evaluated the impact of the number of input branches on performance. The experimental results are shown in Table 6, from which we see that increasing input branch number from 2 to 3 can not further improve classification accuracy.

Comparison with previous work To show the advantage of the proposed approach, we compared the performance of the proposed approach with recent work on CIFAR-10 and CIFAR-100. The comparison results are shown in Table 7.

We see from Table 7 that the proposed method achieves comparable performance with the recent work. The ideas behind mixup [19] and between-class (BC) learning [20] are the same. They use a mixture of two images as input to train deep networks; however, the mixture of two images does not visually make sense. Based on PreAct ResNet-18, our DPLNets and DPLAANets achieve low error rates on the two datasets compared to the mixup method. While the proposed method does not perform as good as the BC method on CIFAR-10 based on ResNeXt-29, our DPLNets and DPLAANets achive 1.03% and 1.27% lower error rates on CIFAR-100 than the BC+ method.

4.2. Image emotion recognition

For image emotion recognition, experiments were carried out on the FI-8 dataset [43]. This dataset was collected from Flickr and Instagram. There are totally 23,308 images labelled with eight emotion categories. The FI-8 dataset is randomly split into 80% training, 5% validation, and 15% testing sets. In our experiments, all training images were resized with the size of the shorter side equal to 256 while maintaining the original aspect ratio. A 224×224 image was randomly cropped from original image or its horizontal flip as input data to networks. Each channel of input data was normalized to have zero mean and unit variance. At test time, the network made a prediction by cropping 10

Model	Error rate (%)	
	CIFAR-10	CIFAR-100
ResNet with stochastic depth [37]	5.25	24.98
ResNet-1001 [35]	4.92	22.71
Wide ResNet-28 [38]	4.17	20.50
PyramidNet [39]	4.70	22.77
CliqueNet [40]	5.06	21.83
DCNet-32 [41]	4.75	20.23
ResNet-18 + cutout [42]	3.99	21.96
ResNet-18 + DPLAANet (Ours)	**4.45**	**19.95**
PreAct ResNet-18 + mixup [19]	4.20	21.10
PreAct ResNet-18 + DPL (Ours)	4.16	20.15
PreAct ResNet-18 + DPLAANet (Ours)	**4.02**	**19.44**
ResNeXt-29 + BC+ [20]	**2.81**	17.93
ResNeXt-29 + DPL (Ours)	3.32	16.90
ResNeXt-29 + DPLAA (Ours)	3.08	**16.66**

Table 7: Comparison with previous work on CIFAR-10 and CIFAR-100.
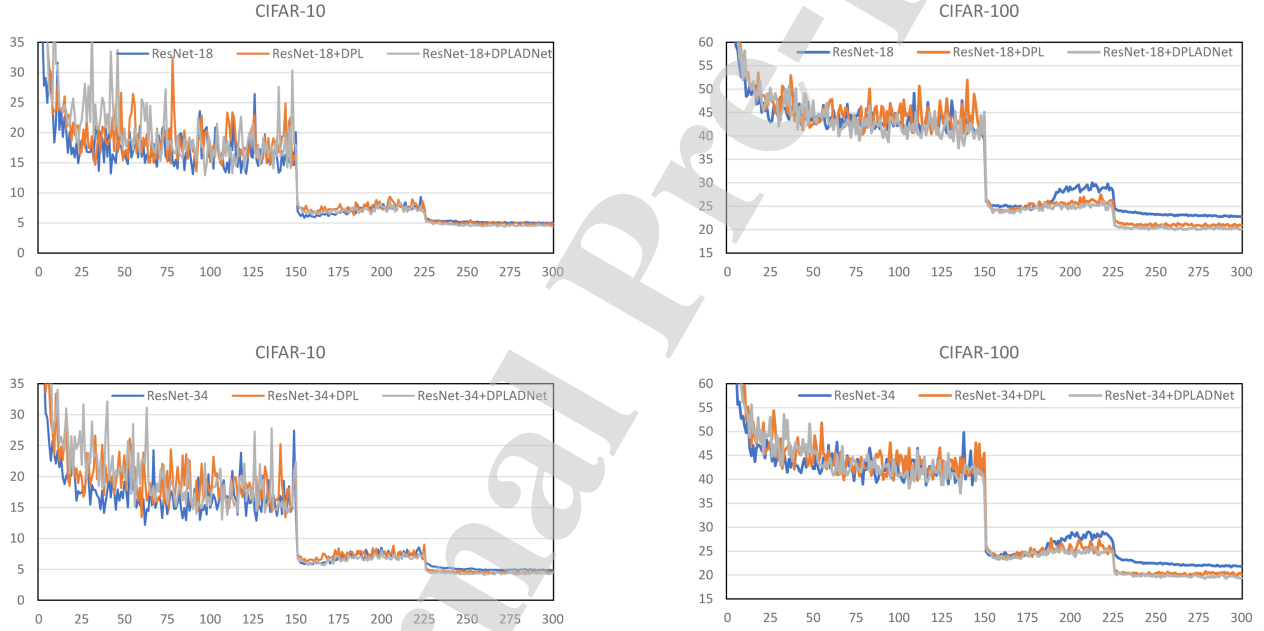

Figure 4: Test errors on CIFAR-10 and CIFAR-100 for ResNets, DPLNets, and DPLAANets.

regions of the size of 224×224 (four corners and one center, and their horizontal flip) from a test image, and averaging the predictions made by the network's softmax layer on the ten patches.

We implemented DPLAANets based on ResNets which are pre-trained on ImageNet. Each input branch of the DPLAANets consists of one block, and the backbone networks consist of three blocks. We trained the DPLAANets using SGD for 90 epochs with a mini-batch of size 128/64. The values of weight decay and momentum were set to 0.0001 and 0.9, respectively. The learning rate for training DPLNets started from 0.1 and was divided by 10 after 30 and 60 epochs. We used the Adam algorithm to train

Model	Recognition accuracy (%)
ResNet-18	64.48
ResNet-18 + DPLAANet	**66.56**
ResNet-34	65.08
ResNet-34 + DPLAANet	**68.17**
ResNet-50	65.99
ResNet-50 + DPLAANet	**68.60**
ResNet-101	66.56
ResNet-101 + DPLAANet	**69.24**

Table 8: Recognition accuracies (%) on FI-8.

8

Num. of samples per category	100	200	300	400	500
ResNet-18	45.76	34.70	27.57	21.07	18.46
ResNet-18 + DPLAANet	**38.02**	**25.14**	**18.84**	**16.25**	**14.47**
PreAct ResNet-18	45.32	33.61	25.62	20.71	18.38
PreAct ResNet-18 + DPLAANet	**33.45**	**22.97**	**17.89**	**15.60**	**13.91**

Table 11: Error rates (%) on subsets of CIFAR-10.

Num. of samples per category	100	200	300	400	500 (Full dataset)
ResNet-18	44.65	32.58	27.06	24.22	22.78
ResNet-18 + DPLAANet	**38.21**	**27.79**	**23.02**	**20.80**	**19.95**
PreAct ResNet-18	55.95	33.03	28.16	25.05	22.48
PreAct ResNet-18 + DPLAANet	**37.63**	**26.96**	**23.34**	**22.92**	**19.44**

Table 12: Error rates (%) on subsets of CIFAR-100.

Num. of samples per category	10	20	30	50	100
LeNet-5	28.73	16.97	12.81	8.31	4.58
LeNet-5 + DPLAANet	20.57	13.89	10.21	6.73	3.94
LeNet-5 + TPLAANet	20.04	13.83	9.98	6.62	3.74

Table 13: Error rates (%) on subsets of MNIST. TPLAANet represents triple pattern learning with adversarial adaptation networks, in which three input branches are used.

Model	Validation set	Test set
LeNet-5	9.8	10.3
LeNet-5 + Mixup ($\alpha = 0.1$)	10.1	10.8
LeNet-5 + DPLAANet	**8.2**	**8.5**

Table 9: Error rates (%) on the Google commands dataset.

Model	Test set
LeNet-5 [no distortions] [22]	0.95
LeNet-5 [huge distortions] [22]	0.85
LeNet-5 [distortions] [22]	0.8
LeNet-5 [no distortions] + DPLAANet	**0.52**

Table 10: Error rates (%) on MNIST.

the discriminator. The learning rate was set to 2e-4. The value of λ was chosen from 0.01, 0.001, 0.0001 and 0.00001. The experimental results are shown in Table 8. From this table, we observe that as with ResNets, the performance of DPLAANets improves as the number of layers increases. The DPLAANets achieve better performance than original ResNets. The DPLAANet based on ResNet-101 achieves the best classification accuracy of 69.24% on this dataset. Using the proposed DPLAANet framework yields an average of 2.63% performance improvement.

4.3. Google commands dataset

We further conducted experiments on speech data. We used the Google commands dataset [44]. This dataset consists of 65,000 utterances which were recorded by thousands of different people. There are totally 30 categories. Each utterance is about one-second long and belongs to one out of 30 short words, such as "yes", "no", "down", and "left". Following the work of [19], we down-sampled from the original waveforms with the sampling rate equal to 16 kHz, and extracted normalized spectrograms. We applied zero-padding to the spectrograms such that their size equal to 160×101. We implemented DPLAANet based on LeNet-5 [22]. Each input branch consists of a convolutional and a subsampling layer, and feature maps generated by two input branches are then fused to backbone network. The first fully connected layer contains 16280 neurons, and the second fully connected layer contains 1000 neurons. The models were trained using SGD with a mini-batch of 100 examples. The learning rated started at 0.001 and was divided by 10 after 50 epochs. The discriminator was trained using the Adam [10] algorithm with learning rate set to 2e-4. The experimental results are shown in Table 9. From this table, we see that our DPLAANet yields 1.6% and 1.8% performance improvement on the validation set and the testing set, respectively.

4.4. MNIST classification

The MNIST digit dataset consists of 60,000 training and 10,000 testing images of ten handwritten digits ("0" to "9"), each with 28×28 pixels. We implemented DPLAANet based on LeNet-5 [22]. The LeNet-5 consists of two convolutional layers, which are followed by subsampling layers, and three fully connected layers with a final softmax. In

9

our implementation, each input branch contains a convolutional and a subsampling layer, and the backbone network consists of a convolutional layer, a subsampling layer, and three fully connected layers. The model was trained from scratch using Adam for 500 epochs with a mini-batch of 128 examples. The learning rate was set to 0.001 and 2e-4 for training the DPLNet and the discriminator, respectively. The experimental results are shown in Table 10. From this table, we find the DPLAANet achieves 0.43% higher performance than original LeNet-5. Our approach also achieve better performance than LeNet-5 trained with distortions of input.

4.5. Experiments on Small Datasets

We conducted experiments on small datasets to demonstrate the advantage of DPLAANets when training data are extremely small. We used subsets of CIFAR-10, CIFAR-100, and MNIST. For experiments on subsets of CIFAR-10 and CIFAR-100, we randomly selected 100, 200, 300, 400, and 500 training samples from each category. The DPLAANets were implemented based on ResNet-18 and PreAct ResNet-18. The parameter setting and the training procedures were the same as in section 4.1. For experiments on subsets of MNIST, we randomly selected 10, 20, 30, 50, and 100 training samples from each category. We used the same DPLAANet structure, parameter setting, and training procedures as in section 4.4, excepted that we used a smaller batch size of 50. The experimental results, which are calculated by averaging three runs, are shown in Table 11, Table 12, and Table 13, respectively.

From Table 11, we find that the DPLAANet based on ResNet-18 and the DPLAANet based on PreAct ResNet-18 yield the heighest performance improvements of 9.56% and 11.87%, respectively, with each category has 200 and 100 training samples, respectively, on CIFAR-10. As the number of training samples in each category increases from 200 to 500, the performance improvement decreases. Overall, they achieve at least 3.99% and 4.47% performance improvements, respectively. The DPLAANets achieve an average of 4.30% and 6.87% performance improvement on subsets of CIFAR-100, respectively, compared to original networks. The performance improvements are higher on CIFAR-10 than on CIFAR-100. This is because CIFAR-100 has more categories, which makes network difficult to distinguish from each other. The DPLAANet yields an average of 3.76% performance improvement on MNIST (see Table 13). Moreover, we see from Table 13 that increasing the number of input branches from 2 to 3 further improves performance.

A general observation from the three tables is that the performance improvement yielded by DPLAANets is high on subsets with each category has a small number of samples. The experimental results show that the proposed dual patter learning with adversarial adaptation framework is very much helpful when training data are limited. This framework would be promising for other tasks in which training samples are extremely difficult to collect.

5. Conclusion

In this paper, we have presented the DPLAANet architecture which comprises a DPLNet and an adversarial adaptation module. This DPLNet can learn robust features by analyzing dual inputs simultaneously compared to conventional networks. The dual input structure of the DPLNet enables the network to have a large number of image pairs to train the network, which can help address the overfitting issue due to limited training data. The adversarial training approach is incorporate to reduce the domain difference between fused image features and single image features. We evaluated DPLAANets on on a diverse of classification tasks including image classification, image emotion recognition, handwritten digit recognition and speech recognition. The experimental results show that DPLAANets could lead to performance improvement over state-of-the-art networks. The experimental results on small datasets show that our DPLAANets have good generalization performance when limited training samples are available. This paper provides a promising approach for applying deep neural networks to tasks in which training samples are extremely difficult to collect.

6. Acknowledgement

References

References

[1] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 2012, pp. 1097–1105.

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

[3] S. Sabour, N. Frosst, G. E. Hinton, Dynamic routing between capsules, in: Advances in Neural Information Processing Systems, 2017, pp. 3859–3869.

[4] X. Liu, X. Zhu, M. Li, L. Wang, E. Zhu, T. Liu, M. Kloft, D. Shen, J. Yin, W. Gao, Multiple kernel k-means with incomplete kernels, IEEE transactions on pattern analysis and machine intelligence.

[5] X. Yu, Blurred trace infrared image segmentation based on template approach and immune factor, Infrared Physics & Technology 67 (2014) 116–120.

[6] Y. Kim, Convolutional neural networks for sentence classification, arXiv preprint arXiv:1408.5882.

[7] X. Zhang, J. Zhao, Y. LeCun, Character-level convolutional networks for text classification, in: Advances in neural information processing systems, 2015, pp. 649–657.

[8] Y. Kim, Y. Jernite, D. Sontag, A. M. Rush, Character-aware neural language models., in: AAAI, 2016, pp. 2741–2749.

[9] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, A. Farhadi, Target-driven visual navigation in indoor scenes using deep reinforcement learning, in: Robotics and Automation (ICRA), 2017 IEEE International Conference on, IEEE, 2017, pp. 3357–3364.

[10] G. Lample, D. S. Chaplot, Playing fps games with deep reinforcement learning., in: AAAI, 2017, pp. 2140–2146.

[11] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, T. Darrell, Decaf: A deep convolutional activation feature for generic visual recognition, in: International conference on machine learning, 2014, pp. 647–655.

[12] H. Zhang, M. Xu, Recognition of emotions in user-generated videos with kernelized features, IEEE Transactions on Multimedia.

[13] G. Huang, Z. Liu, K. Q. Weinberger, L. van der Maaten, Densely connected convolutional networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, Vol. 1, 2017, p. 3.

[14] S. Xie, R. Girshick, P. Dollár, Z. Tu, K. He, Aggregated residual transformations for deep neural networks, in: Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on, IEEE, 2017, pp. 5987–5995.

[15] A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images.

[16] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556.

[17] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, International Journal of Computer Vision 115 (3) (2015) 211–252.

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

[19] H. Zhang, M. Cisse, Y. N. Dauphin, D. Lopez-Paz, mixup: Beyond empirical risk minimization, arXiv preprint arXiv:1710.09412.

[20] Y. Tokozume, Y. Ushiku, T. Harada, Between-class learning for image classification, arXiv preprint arXiv:1711.10284.

[21] K. Fukushima, Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position, Biological cybernetics 36 (4) (1980) 193–202.

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

[23] Z. Zheng, L. Zheng, Y. Yang, Unlabeled samples generated by gan improve the person re-identification baseline in vitro, arXiv preprint arXiv:1701.07717 3.

[24] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie, A. Yuille, Adversarial examples for semantic segmentation and object detection, in: The IEEE International Conference on Computer Vision (ICCV), 2017.

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

[26] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, arXiv preprint arXiv:1502.03167.

[27] E. Tzeng, J. Hoffman, K. Saenko, T. Darrell, Adversarial discriminative domain adaptation, in: Computer Vision and Pattern Recognition (CVPR), Vol. 1, 2017, p. 4.

[28] L. Hu, M. Kan, S. Shan, X. Chen, Duplex generative adversarial network for unsupervised domain adaptation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 1498–1507.

[29] W. Zhang, W. Ouyang, W. Li, D. Xu, Collaborative and adversarial network for unsupervised domain adaptation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 3801–3809.

[30] V. Vapnik, Statistical learning theory. 1998, Wiley, New York, 1998.

[31] Z. Ren, Y. J. Lee, Cross-domain self-supervised multi-task feature learning using synthetic imagery, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[32] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, R. Webb, Learning from simulated and unsupervised images through adversarial training, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2107–2116.

[33] T. Miyato, T. Kataoka, M. Koyama, Y. Yoshida, Spectral normalization for generative adversarial networks, arXiv preprint arXiv:1802.05957.

[34] A. Paszke, S. Gross, S. Chintala, G. Chanan, Pytorch (2017).

[35] K. He, X. Zhang, S. Ren, J. Sun, Identity mappings in deep residual networks, in: European Conference on Computer Vision, Springer, 2016, pp. 630–645.

[36] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.

[37] G. Huang, Y. Sun, Z. Liu, D. Sedra, K. Q. Weinberger, Deep networks with stochastic depth, in: European Conference on Computer Vision, Springer, 2016, pp. 646–661.

[38] S. Zagoruyko, N. Komodakis, Wide residual networks, arXiv preprint arXiv:1605.07146.

[39] D. Han, J. Kim, J. Kim, Deep pyramidal residual networks, in: Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on, IEEE, 2017, pp. 6307–6315.

[40] Y. Yang, Z. Zhong, T. Shen, Z. Lin, Convolutional neural networks with alternately updated clique, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2413–2422.

[41] A. DeFonzo, J. Tauc, Decoupled networks: An approach to low symmetry vibrational problems, Solid State Communications 18 (8) (1976) 937–940.

[42] T. DeVries, G. W. Taylor, Improved regularization of convolutional neural networks with cutout, arXiv preprint arXiv:1708.04552.

[43] Q. You, J. Luo, H. Jin, J. Yang, Building a large scale dataset for image emotion recognition: The fine print and the benchmark., in: AAAI, 2016, pp. 308–314.

[44] P. Warden, Launching the speech commands dataset, https://research.googleblog.com/2017/08/launching-speech-commands-dataset.html.

Credit Author Statement

Haimin Zhang: Conceptualization, Methodology, Data Curation, Software, Writing - Original Draft.

Min Xu: Conceptualization, Supervision, Writing - Review & Editing.

Conflict of Interest and Authorship Conformation Form

Please check the following as appropriate:

- o   All authors have participated in (a) conception and design, or analysis and interpretation of the data; (b) drafting the article or revising it critically for important intellectual content; and (c) approval of the final version.

- o   This manuscript has not been submitted to, nor is under review at, another journal or other publishing venue.

- o   The authors have no affiliation with any organization with a direct or indirect financial interest in the subject matter discussed in the manuscript

- o   The following authors have affiliations with organizations with direct or indirect financial interest in the subject matter discussed in the manuscript:

Author's name   Affiliation

| Haimin Zhang | University of Technology Sydney |
| A/Prof. Min Xu | University of Technology Sydney |