The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20)

Adversarial Domain Adaptation with Domain Mixup

Minghao Xu,[1] **Jian Zhang,**[1] **Bingbing Ni,**[1,2*] **Teng Li,**[3]
Chengjie Wang,[4] **Qi Tian,**[5] **Wenjun Zhang**[1]
[1]Shanghai Jiao Tong University, China, [2]Huawei Hisilicon,
[3]Anhui University, [4]Youtu Lab, Tencent, [5]Huawei Noah's Ark Lab
{xuminghao118, stevenash0822, nibingbing, zhangwenjun}@sjtu.edu.cn, nibingbing@hisilicon.com,
tenglwy@gmail.com, jasoncjwang@tencent.com, tian.qi1@huawei.com

Abstract

Recent works on domain adaptation reveal the effectiveness of adversarial learning on filling the discrepancy between source and target domains. However, two common limitations exist in current adversarial-learning-based methods. First, samples from two domains alone are not sufficient to ensure domain-invariance at most part of latent space. Second, the domain discriminator involved in these methods can only judge real or fake with the guidance of hard label, while it is more reasonable to use soft scores to evaluate the generated images or features, *i.e.*, to fully utilize the inter-domain information. In this paper, we present adversarial domain adaptation with **domain mixup** (DM-ADA), which guarantees domain-invariance in a more continuous latent space and guides the domain discriminator in judging samples' difference relative to source and target domains. Domain mixup is jointly conducted on pixel and feature level to improve the robustness of models. Extensive experiments prove that the proposed approach can achieve superior performance on tasks with various degrees of domain shift and data complexity.

Introduction

In recent years, with the appearance of Convolutional Neural Networks (CNNs), many classification-based challenges have been tackled with an extremely high accuracy. These powerful CNN architectures, like AlexNet (Krizhevsky, Sutskever, and Hinton 2012) and ResNet (He et al. 2016), are capable of efficiently extracting low-level and high-level features with the guidance of labeled data. However, because of the existence of domain shift, models trained on a specific domain suffer from poorer performance when transferred to another domain. This problem is of vital significance in the case where labeled data are unavailable on target domain. Thus how to use these unlabeled data from target domain to fill the domain discrepancy is the main issue in the field of domain adaptation (Pan and Yang 2010).

Beginning with the work of gradient reverse (Ganin and Lempitsky 2015), a group of domain adaptation methods based on adversarial learning were proposed. In the research of this subfield, a domain discriminator is introduced to judge the domain attribute on feature or image level. In order to fool this domain discriminator, the extracted features should be domain-invariant, which is the basic motivation of adversarial domain adaptation. However, just like most variants of Generative Adversarial Networks (GANs) (Goodfellow et al. 2014), the domain discriminator is *only* guided by the hard label information and *rarely* explores the intrinsic structure of data distribution. Namely, each data point that shifts from both domains should be judged by a latent soft label, *i.e.*, a probability value, instead of a hard assignment of "1" or "0". In addition, the distribution of domain-invariant latent vectors is fitted using limited patterns of source and target features, *i.e.*, the interaction of features from two domains has not been considered to enrich feature patterns.

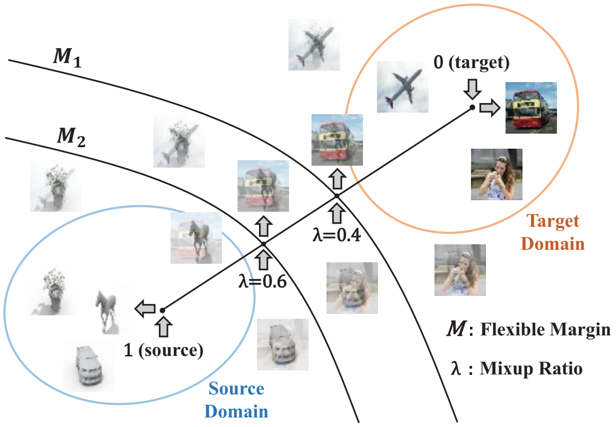


Figure 1: The mixup samples are obtained by linearly interpolating between source and target domain images with mixup ratio λ. The domain discriminator learns how to output soft scores for generated images with the guidance of mixup images and *soft labels* (mixup ratios). Also, a flexible margin is learned through comparing mixup samples' difference relative to two domains. (Best viewed in color.)

In this paper, inspired by VAE-GAN (Larsen et al. 2016), we develop a generative-adversarial-based framework to simultaneously train the classification network and generate auxiliary source-like images from learned embeddings of both domains. On the basis of this framework, **domain mixup** on pixel and feature level is proposed to alleviate two existing drawbacks. (1) Just as shown in Figure 1, we would like to instruct the domain discriminator to explore the in-

*The corresponding author is Bingbing Ni.

trinsic structure of source and target distributions through triplet loss with a flexible margin and applying domain classification with mixed images and corresponding soft labels (*i.e.*, mixup ratio), which provides abundant intermediate status between two separate domains. (2) In order to expand the searching range in latent space, linear interpolations of source and target features are exploited. Since the subsequent nonlinear neural network can easily ruin the linear mixed information, the extracted features of mixed images are not used for augmentation directly. This operation leads to a more continuous domain-invariant latent distribution, which benefits the performance on target domain when the oscillation of data distribution occurs in the test phase.

We evaluate the image recognition performance of our approach on three benchmarks with different extent of domain shift. Experiments prove the effectiveness of our approach, and we achieve state-of-the-art in most settings. The contributions of our work are summarized as follows:

- We design an adversarial training framework which maps both domains to a common latent distribution, and efficiently transfer our knowledge learned on the supervised domain to its unsupervised counterpart.
- Domain mixup on pixel and feature level accompanied with well-designed soft domain labels is proposed to improve the generalization ability of models. This method promotes the generalization ability of feature extractor and obtains a domain discriminator judging samples' difference relative to two domains with refined scores.
- We extensively evaluate our approach under different settings, and our approach achieves superior results even when the domain shift is high and the data distribution is complex.

Related Work

Domain adaptation is a frequently used technique to promote the generalization ability of models trained on a single domain in many Computer Vision tasks. In this section, we describe existing domain adaptation methods and compare our approach with them.

The transferability of Deep Neural Networks is proved in (Yosinski et al. 2014), and deep learning methods for domain adaptation can be classified into several categories. Maximum Mean Discrepancy (MMD) (Gretton et al. 2012; Tzeng et al. 2014) is a way to measure the similarity of two distributions. Weighted Domain Adaptation Network (WDAN) (Yan et al. 2017) defines the weighted MMD with class conditional distribution on both domains. The multiple kernel version of MMD (MK-MMD) is explored in (Long et al. 2015) to define the distance between two distributions. In addition, specific deep neural networks are constructed to restrict the domain-invariance of top layers by aligning the second-order statistics (Sun and Saenko 2016).

Adversarial Training (Ajakan et al. 2013; Ganin et al. 2016) is another way to transfer domain information. RevGrad (Ganin and Lempitsky 2015) designs a double branched architecture for object classification and domain classification respectively. Adversarial Discriminative Domain Adaptation (ADDA) (Tzeng et al. 2017) trains two feature extractors for source and target domains respectively, and produces embeddings fooling the discriminator. Other works optimize the performance on target domain by capturing complex multimode structures (Pei et al. 2018; Long et al. 2018), exploring task-specific decision boundaries (Saito et al. 2018b; 2018a; Tran et al. 2019), aligning the attention regions (Kang et al. 2018) and applying structure-aware alignment (Ma, Zhang, and Xu 2019). In addition, the label information of target domain is explored in recent works (Zhang et al. 2018b; Xie et al. 2018; Ma, Zhang, and Xu 2019).

Another group of methods perform adaptation by applying adversarial loss on pixel level. Source domain images are adapted as if they are drawn from target domain using generative adversarial networks in (Bousmalis et al. 2017; Liu and Tuzel 2016), and generated samples expand the training set. Furthermore, image generation and training the task-specific classifier are accomplished simultaneously in (Ghifary et al. 2016; Sankaranarayanan et al. 2018). Cycle-consistency is also considered in (Hoffman et al. 2018) to enforce the consistency of relevant semantics.

Comparison with existing GAN-based approaches. Although former works use GAN as a manner of data augmentation (Bousmalis et al. 2017; Liu and Tuzel 2016) or producing domain adaptive gradient information (Ghifary et al. 2016; Sankaranarayanan et al. 2018), they may be trapped in the mismatch between generated data and assigned hard labels. We further explore the usage of **domain mixup** on pixel and feature level to enhance the robustness of adaptation models. On one hand, pixel-level mixup prompts the domain discriminator to excavate the intrinsic structure of source and target distributions. On the other hand, feature-level mixup facilitates a more continuous feature distribution in the latent space with low domain shift.

Adversarial Domain Adaptation with Domain Mixup

In unsupervised domain adaptation, a source domain dataset $(X_s, Y_s) = \{(x_i^s, y_i^s)\}_{i=1}^{n_s}$ with n_s labeled samples and a target domain dataset $X_t = \{x_j^t\}_{j=1}^{n_t}$ with n_t unlabeled samples are available. It is assumed that source samples x^s obey the source distribution P_s, and target samples x^t obey the target distribution P_t. In addition, both domains share the same label space $Y = \{1, 2, 3, \cdots, K\}$, where K is the number of classes.

Framework of DM-ADA

In this work, a variant of VAE-GAN (Larsen et al. 2016) is applied to the domain adaptation task. Figure 2 presents an overview of the whole framework. For the input, there are three kinds: source domain images, target domain images and mixup images obtained by pixel-wise addition of source and target images. Just as conventional variational autoencoder (Kingma and Welling 2013), an encoder N_e maps inputs from source and target domains to the standard Gaussian distribution $\mathcal{N}(0, \mathbf{I})$. For every sample, a mean vector μ and a standard deviation vector σ are served as the feature embedding. On feature level, the feature embeddings of

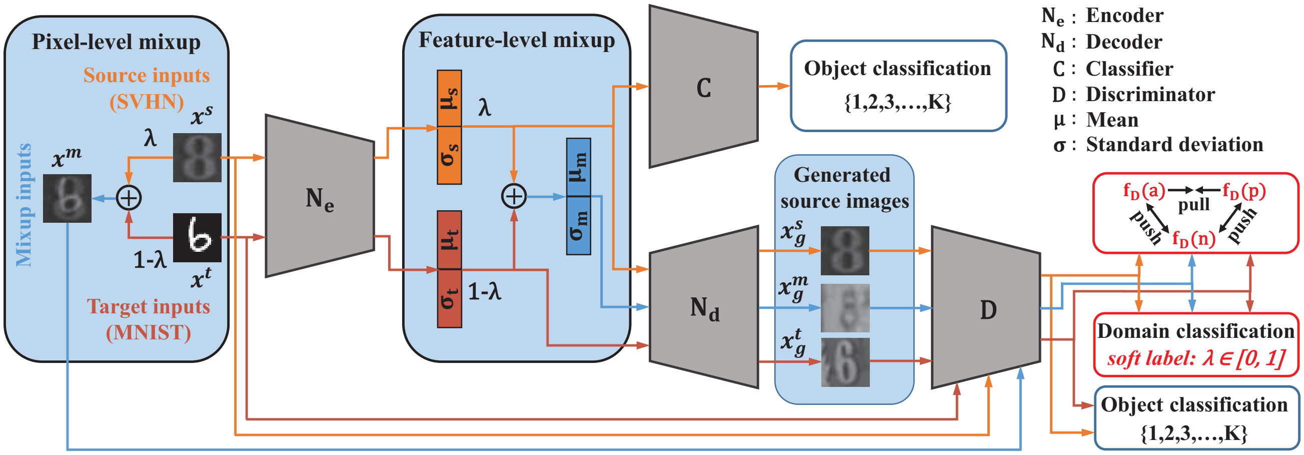


Figure 2: Illustration of the pipeline in the training phase. First, mixup inputs x^m are obtained by mixing two domains' inputs x^s and x^t. Encoder N_e maps source and target inputs to (μ_s, σ_s) and (μ_t, σ_t) respectively. In the latent space, feature embeddings of two domains are mixed to produce mixup features (μ_m, σ_m). After that, the framework is split to two branches. On one branch, classifier C performs K-way object classification. On the other branch, latent codes are decoded by N_d, and the min-max game between N_d and D facilitates domain-invariance on category level. (Best viewed in color.)

two domains are also linearly mixed to produce mixup features (μ_m, σ_m). After that, the framework is split into two branches. For one branch, the embedding of source domain is used to do K-way object classification by the classifier C. For the other branch, source and target domain are aligned on category level through enforcing the decoded images to be source-like and preserve class information of inputs. Details are stated in the following parts.

Domain mixup on two levels. To explore the internal structure of data from two domains, source domain images x^s and target domain images x^t are linearly interpolated (Zhang et al. 2018a) to produce mixup images x^m and corresponding soft domain labels l^m_{dom} as follows:

$$x^m = \lambda x^s + (1 - \lambda) x^t, \tag{1}$$

$$l^m_{dom} = \lambda l^s_{dom} + (1 - \lambda) l^t_{dom} = \lambda, \tag{2}$$

where $\lambda \in [0, 1]$ is the mixup ratio, and $\lambda \sim \text{Beta}(\alpha, \alpha)$, in which α is constantly set as 2.0 in all experiments. l^s_{dom} and l^t_{dom} represent the domain label of source and target data, which are manually set as 1 and 0.

Inputs of source and target domains are then embedded to (μ_s, σ_s) and (μ_t, σ_t) in the latent space by a shared encoder N_e. In order to yield a more continuous domain-invariant latent distribution, two domains' embeddings are linearly mixed to produce mixup feature embedding (μ_m, σ_m):

$$\mu_m = \lambda \mu_s + (1 - \lambda) \mu_t, \tag{3}$$

$$\sigma_m = \lambda \sigma_s + (1 - \lambda) \sigma_t, \tag{4}$$

where λ equals to the one used in pixel-level mixup.

Restricting encoder with priori. Just as conventional VAE (Kingma and Welling 2013), the encoder N_e is regularized by a standard gaussian priori over the latent distribution. The objective is to narrow the Kullback-Leibler divergence between posteriori and priori:

$$\min_{N_e} \mathcal{L}_{KL}, \tag{5}$$

$$\mathcal{L}_{KL} = D_{KL}\big(\mathcal{N}(\mu, \sigma) || \mathcal{N}(0, \mathbf{I})\big), \tag{6}$$

where μ and σ are the encoded mean and standard deviation of source and target images.

Supervised training for classifier. The classifier C is optimized with cross entropy loss defined on source domain, and the objective is as follows:

$$\min_{N_e, C} \mathcal{L}_C, \tag{7}$$

$$\mathcal{L}_C = -\mathbf{E}_{x^s \sim P_s} \sum_{i=1}^{K} y_i^s \log\big(C([\mu_s, \sigma_s])\big), \tag{8}$$

where $[\cdot]$ denotes concatenation. It is worth noticing that classifier C can't be replaced by the object classification branch of discriminator D, since the adapted features are only passed directly to C, which enhances C's performance on target domain.

Decoding latent codes. Before the generation phase, we first define the one-hot object class label l_{cls} and a one-dimensional uncertainty compensation l_{comp} for both domains and mixup features as below:

$$\begin{aligned} l^s_{cls} &= [0, 0, \cdots, 1, \cdots, 0], & l^s_{comp} &= 0, \\ l^t_{cls} &= [0, 0, \cdots, 0, \cdots, 0], & l^t_{comp} &= 1, \\ l^m_{cls} &= [0, 0, \cdots, \lambda, \cdots, 0], & l^m_{comp} &= 1 - \lambda, \end{aligned} \tag{9}$$

where 1 and λ are on the y^s-th position of l^s_{cls} and l^m_{cls} to indicate the known class label for both features respectively. For all features derived from target domain or mixup procedure, since the class labels remain uncertain, l_{comp} is set as a compensation to normalize the sum of vector l_{cls} and l_{comp} to 1. After that, decoder N_d predicts the auxiliary generated images x_g as below:

$$x_g = N_d([\mu, \sigma, z, l_{cls}, l_{comp}]), \tag{10}$$

where z is the noise vector randomly sampled from standard Gaussian distribution.

Adversarial domain alignment. Compared with previous adversarial-learning-based methods (Ganin and Lempitsky 2015; Tzeng et al. 2017; Pei et al. 2018), we constrain domain-invariance not only on source and target domains, but also on the intermediate representations between two domains. The min-max optimization objective on different domains are defined as follows:

$$\min_{N_e, N_d} \max_{D} \mathcal{L}_{adv}^{s} + \mathcal{L}_{adv}^{t} + \mathcal{L}_{adv}^{m}, \tag{11}$$

$$\mathcal{L}_{adv}^{s} = \mathbf{E}_{x^s \sim P_s} \log\big(D_{dom}(x^s)\big) + \log\big(1 - D_{dom}(x_g^s)\big), \tag{12}$$

$$\mathcal{L}_{adv}^{t} = \mathbf{E}_{x^t \sim P_t} \log\big(1 - D_{dom}(x_g^t)\big), \tag{13}$$

$$\mathcal{L}_{adv}^{m} = \mathbf{E}_{x^s \sim P_s, x^t \sim P_t} \log\big(1 - D_{dom}(x_g^m)\big), \tag{14}$$

where D_{dom} is the domain classification branch of D. During training process, the mixup features can well be mapped to somewhere in-between source and target domain on pixel level, and it is more proper to assign them with scores between 0 and 1. Domain classification loss $\mathcal{L}_{soft}^{m}$ is utilized to guide domain discriminator output such soft scores:

$$\min_{D} \mathcal{L}_{soft}^{m}, \tag{15}$$

$$\begin{aligned} \mathcal{L}_{soft}^{m} = &- \mathbf{E}_{x^s \sim P_s, x^t \sim P_t}\, l_{dom}^{m} \log\big(D_{dom}(x^m)\big) \\ &+ (1 - l_{dom}^{m}) \log\big(1 - D_{dom}(x^m)\big). \end{aligned} \tag{16}$$

We further introduce a triplet loss $\mathcal{L}_{tri}^{m}$ to constrain mixup samples' distance to source and target domains, which makes domain discriminator easier to converge:

$$\min_{D} \mathcal{L}_{tri}^{m}, \tag{17}$$

$$\begin{aligned} \mathcal{L}_{tri}^{m} = &\mathbf{E}_{x^s \sim P_s, x^t \sim P_t} \big[||f_D(a) - f_D(p)||_2^2 \\ &- ||f_D(a) - f_D(n)||_2^2 + f_{tri}(\lambda) \big]_{+}, \end{aligned} \tag{18}$$

where f_D is the feature extractor of D, and $[\cdot]_+ = max(0, \cdot)$ denotes the hinge loss function; $(a, p, n) = (x^m, x^s, x^t)$, when $\lambda \geqslant 0.5$, and $(a, p, n) = (x^m, x^t, x^s)$, otherwise. Considering that samples with more source or target domain components should have larger difference with the counterpart domain, a flexible margin $f_{tri}(\lambda) = |2\lambda - 1|$ is used.

Category-level domain alignment. In order to ensure the identical categories' features of two domains are mapped nearby in the latent space, classification loss $\mathcal{L}_{cls}^{s}$ and $\mathcal{L}_{cls}^{t}$ are introduced to ensure the class-consistency between decoded images and inputs:

$$\min_{N_e, N_d, D} \mathcal{L}_{cls}^{s} + \mathcal{L}_{cls}^{t}, \tag{19}$$

$$\mathcal{L}_{cls}^{s} = -\mathbf{E}_{x^s \sim P_s} \sum_{i=1}^{K} y_i^s \log\big(D_{cls}(x_g^s)\big), \tag{20}$$

Algorithm 1 Training procedure of DM-ADA

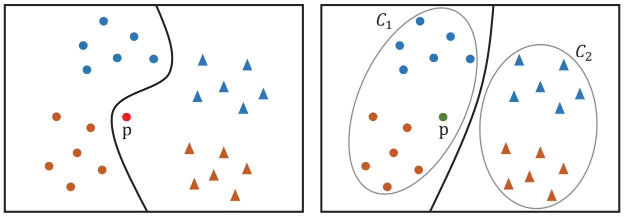
```
Input: Source domain: $X_s$ and $Y_s$, target domain: $X_t$ and the number of iterations $N$.
Output: Configurations of DM-ADA.
Initialize $\omega$ and $\varphi$
Initialize $\theta_{N_e}$, $\theta_{N_d}$, $\theta_C$ and $\theta_D$
for $n = 1$ to $N$ do
    $(x^s, y^s) \leftarrow$ RANDOMSAMPLE$(X_s, Y_s)$
    $(x^t) \leftarrow$ RANDOMSAMPLE$(X_t)$
    $\lambda \leftarrow$ RANDOMSAMPLE(Beta$(\alpha, \alpha)$)
    $(x^m, l_{dom}^m) \leftarrow$ Eq. (1, 2)            # Get mixup images
    $(\mu, \sigma) \leftarrow N_e(x)$                   # Get feature embeddings
    $(\mu_m, \sigma_m) \leftarrow$ Eq. (3, 4)           # Get mixup features
    $x_g = N_d(\mu, \sigma, z, l_{cls}, l_{comp})$      # Generate images
    Optimize four subnetworks $D$, $N_d$, $C$ and $N_e$ in turn:
        $\theta_D \overset{+}{\leftarrow} -\nabla_{\theta_D}\big(\mathcal{L}_{cls}^s + \omega(\mathcal{L}_{tri}^m + \mathcal{L}_{soft}^m) + \varphi(\mathcal{L}_{adv}^s + \mathcal{L}_{adv}^t + \mathcal{L}_{adv}^m)\big)$
        $\theta_{N_d} \overset{+}{\leftarrow} -\nabla_{\theta_{N_d}}(\mathcal{L}_{cls}^s - \varphi\mathcal{L}_{adv}^s)$
        $\theta_C \overset{+}{\leftarrow} -\nabla_{\theta_C}(\mathcal{L}_C)$
        $\theta_{N_e} \overset{+}{\leftarrow} -\nabla_{\theta_{N_e}}\big(\mathcal{L}_C + \mathcal{L}_{cls}^s + \mathcal{L}_{cls}^t + \omega\mathcal{L}_{KL} - \varphi(\mathcal{L}_{adv}^t + \mathcal{L}_{adv}^m)\big)$
end for
```

$$\mathcal{L}_{cls}^{t} = -\mathbf{E}_{x^t \sim P_t} \sum_{i=1}^{K} \hat{y}_i^t \log\big(D_{cls}(x_g^t)\big), \tag{21}$$

where D_{cls} is the object classification branch of D, and $\hat{y}^t$ is the pseudo label estimated by classifier C. So as to eliminate falsely labeled samples which harm domain adaptation, we filter out those samples whose classification confidence below a certain threshold τ. Considering the fact that domain discrepancy is gradually filled along training, τ is adaptively adjusted following the strategy in (Zhang et al. 2018b).

Training Procedure

The proposed iterative training procedure is summarized in Algorithm 1. In each iteration, the input source and target samples are first mixed on pixel level to instruct the domain discriminator to output soft labels. After the samples of two domains are mapped to the latent space, their embeddings are mixed to produce mixup features. The images generated on the basis of these feature embeddings are constrained to be source-like and preserve inputs' class information, so that the latent distribution is facilitated to be domain-invariant and discriminative. In all experiments, we set α as 2.0, since domain mixup can't effectively explore the linear space between two domains when the value of α is small, and more analysis of α can be found in supplementary material. ω and φ are hyper-parameters that trade off among losses with different orders of magnitude. According to the after sensitivity analysis, the adaptation performance of our approach is not too sensitive to the value of ω and φ, and these hyper-parameters share the same value among different tasks.



w/o domain continuity w/ domain continuity

Figure 3: The comparison between with and without domain continuity. Circle & triangle: two classes; Blue & orange: source and target domain; p: test sample. (Best viewed in color.)

Discussion

Pixel-level domain mixup. The work of (Zhang et al. 2018a) proposes the *mixup* vicinal distribution as a manner to encourage the model to behave linearly in-between training examples. Another work (Berthelot et al. 2019) improves interpolation's continuity in latent space and benefits downstream tasks. In adversarial domain adaptation, we also would like to lead the domain discriminator to behave linearly between source and target domains. As a result, the domain discriminator is of high capacity to accurately judge the generated images containing oscillations to two domains. In our implementation, such discriminator is trained with pairs of linearly mixed image $x^m = \lambda x^s + (1-\lambda)x^t$ and corresponding soft label $l^m_{dom} = \lambda$, where x^m simulates an oscillation mode to two domains and λ provides the guidance. Combined with feature-level mixup, pixel-level mixup can further narrow the domain discrepancy, which is shown in the after ablation study.

Feature-level domain mixup. Existing works attempt to map source and target domains to a common latent distribution, while limited data can not guarantee most parts of the latent space domain-invariant. In order to yield a more continuous domain-invariant latent distribution, the mixup features of two domains are exploited.

We use an intuitive example to illustrate the effectiveness of domain continuity on aligning source and target domains. As shown in Figure 3, the biased test sample p may be misclassified without the constraint of domain continuity. However, through adding the mixup feature embedding to the training process, the latent codes between the same class of two domains should also be domain-invariant, which forms the intra-class clusters C_1 and C_2. Thus the decision boundary is refined, and the biased samples in these clusters can be classified correctly.

Experiments

In this section, we first introduce the experimental setup. Then, the classification performance on three domain adaptation benchmarks are presented. Finally, ablation study and sensitivity analysis are conducted for the proposed approach.

Table 1: Classification accuracy (mean ± std %) values of target domain over five independent runs on the digits datasets. The best performance is indicated in **bold** and the second best one is $\underline{\text{underlined}}$.

Method	MN → US (p)	MN → US (f)	US → MN	SV → MN
Source only	76.0 ± 1.8	79.3 ± 0.7	59.5 ± 1.9	62.1 ± 1.2
MMD (2015)	–	81.1 ± 0.3	–	71.1 ± 0.5
RevGrad (2015)	77.1 ± 1.8	85.1 ± 0.8	73.0 ± 2.0	73.9 ± 1.2
CoGAN (2016)	91.2 ± 0.8	–	89.1 ± 1.0	–
DRCN (2016)	91.8 ± 0.1	–	73.7 ± 0.0	82.0 ± 0.1
ADDA (2017)	89.4 ± 0.2	–	90.1 ± 0.8	76.0 ± 1.8
PixelDA (2017)	–	95.9 ± 0.7	–	–
MSTN (2018)	92.9 ± 1.1	–	–	91.7 ± 1.5
GTA (2018)	92.8 ± 0.9	95.3 ± 0.7	90.8 ± 1.3	92.4 ± 0.9
ADR (2018a)	$\underline{93.2} \pm 2.5$	$\underline{96.1} \pm 0.3$	$\underline{93.1} \pm 1.3$	$\underline{95.0} \pm 1.9$
DM-ADA (ours)	**94.8** ± 0.7	**96.7** ± 0.5	**94.2** ± 0.9	**95.5** ± 1.1

Experimental Setup

In this part, we describe the network architectures and hyper-parameters of different tasks. Our approach is implemented with PyTorch deep learning framework (Paszke et al. 2017).

Digits experiments. In this part of experiments, we construct four subnetworks with train-from-scratch architectures following (Sankaranarayanan et al. 2018). Four Adam optimizers with base learning rate 0.0004 are utilized to optimize these submodels for 100 epochs. The hyper-parameters ω and φ are set as 0.1 and 0.01 respectively, and their values are constant in all experiments. All of the input images of encoder and discriminator are resized to 32×32.

Office experiments. For the encoder, the last layer of AlexNet (Krizhevsky, Sutskever, and Hinton 2012) is replaced with two parallel fully connected layers producing 256 dimensional vectors respectively, and former layers are initialized with the model pretrained on ImageNet (Russakovsky et al. 2015). The encoder is fine-tuned with base learning rate 0.0001 for 100 epochs, and the base learning rate of other three submodels is set as 0.001. The inputs of encoder and discriminator are resized to 227×227 and 64×64 respectively.

VisDA experiments. ResNet-101 (He et al. 2016) serves as the base architecture, and it is initialized with the model pretrained on ImageNet (Russakovsky et al. 2015). The learning rate setting is same as that in the office experiments, and the results are reported after 20 epochs training. The inputs of encoder and discriminator are resized to 224×224 and 64×64 respectively.

Classification on Digits Datasets

Dataset. In this set of experiments, three digits datasets are used: MNIST (Lécun et al. 1998), USPS (Hull 1994) and Street View House Numbers (SVHN) (Netzer et al. 2011). Each dataset contains ten classes corresponding to number 0 to 9. Four settings are used for measurement: MN → US (p): sampling 2000 images from MNIST and 1800 images from USPS; MN → US (f) and US → MN: using the full training set of MNIST and USPS; SV → MN: using the full training set of SVHN and MNIST.

Results. Table 1 presents the results of our approach in

Table 2: Classification accuracy (mean ± std %) values of target domain over five independent runs on the Office-31 dataset. The best performance is indicated in **bold** and the second best one is underlined.

Method	A → W	D → W	W → D	A → D	D → A	W → A	Average
AlexNet (source only) (2012)	60.6 ± 0.4	95.4 ± 0.2	99.0 ± 0.1	64.2 ± 0.3	45.5 ± 0.5	48.3 ± 0.5	68.8
TCA (2011)	59.0 ± 0.0	90.2 ± 0.0	88.2 ± 0.0	57.8 ± 0.0	51.6 ± 0.0	47.9 ± 0.0	65.8
DDC (2014)	61.0 ± 0.5	95.0 ± 0.3	98.5 ± 0.3	64.9 ± 0.4	47.2 ± 0.5	49.4 ± 0.4	69.3
DAN (2015)	68.5 ± 0.3	96.0 ± 0.1	99.0 ± 0.1	66.8 ± 0.2	50.0 ± 0.4	49.8 ± 0.3	71.7
RevGrad (2015)	73.0 ± 0.5	96.4 ± 0.3	99.2 ± 0.3	72.3 ± 0.3	52.4 ± 0.4	50.4 ± 0.5	74.1
DRCN (2016)	68.7 ± 0.3	96.4 ± 0.3	99.0 ± 0.2	66.8 ± 0.5	56.0 ± 0.5	54.9 ± 0.5	73.6
MADA (2018)	78.5 ± 0.2	$\mathbf{99.8} \pm 0.1$	$\mathbf{100} \pm 0.0$	74.1 ± 0.1	56.0 ± 0.2	54.5 ± 0.3	77.1
MSTN (2018)	80.5 ± 0.4	96.9 ± 0.1	$\underline{99.9} \pm 0.1$	74.5 ± 0.4	62.5 ± 0.4	60.0 ± 0.6	79.1
GCAN (2019)	$\underline{82.7} \pm 0.1$	$\underline{97.1} \pm 0.1$	99.8 ± 0.1	$\underline{76.4} \pm 0.5$	$\mathbf{64.9} \pm 0.1$	$\underline{62.6} \pm 0.3$	$\underline{80.6}$
DM-ADA (ours)	$\mathbf{83.9} \pm 0.4$	$\mathbf{99.8} \pm 0.1$	$\underline{99.9} \pm 0.1$	$\mathbf{77.5} \pm 0.2$	$\underline{64.6} \pm 0.4$	$\mathbf{64.0} \pm 0.5$	**81.6**

Table 3: Classification accuracy on the validation set of VisDA-2017 challenge.

Method	Accuracy (%)
ResNet-101 (source only) (2016)	52.4
RevGrad (2015)	57.4
DAN (2015)	62.8
JAN (2017)	65.7
GTA (2018)	69.5
MCD-DA (2018b)	71.9
ADR (2018a)	$\underline{73.5}$
DM-ADA (ours)	**75.6**

comparison with other adaptation approaches on the digits datasets. For the source only test, we use the same encoder and classifier architectures as the ones used in our approach. The reported results are averaged over five independent runs with random initialization. Our approach achieves the state-of-the-art performance on all four settings. Especially, it outperforms former GAN-based approaches (Bousmalis et al. 2017; Ghifary et al. 2016; Sankaranarayanan et al. 2018), which illustrates the effectiveness of the proposed architecture on aligning source and target domains.

Classification on Office-31

Dataset. Office-31 (Saenko et al. 2010) is a standard domain adaptation benchmark commonly used in previous researches. Three distinct domains, Amazon(A), Webcam(W) and DSLR(D), make up of the whole Office-31 dataset. Each domain contains the same 31 classes of office supplies. All transfer tasks of three domains are used for evaluation.

Results. Table 2 reports the performance of our method compared with other works. The results of AlexNet trained with only source domain data serves as the lower bound. Our approach obtains the best performance in three of four hard cases: A → W, W → A and A → D. For two easier cases: W → D and D → W, our approach achieves accuracy higher than 99.5% and ranks the first two places. Given the fact that the number of samples per class is limited in the Office-31 dataset, our approach manages to improve the performance by providing augmented samples and features.

Table 4: Effectiveness of pixel-level mixup (PM), feature-level mixup (FM) and triplet loss (Tri).

PM	FM	Tri	$\mathcal{A}$-distance	Accuracy (%)
			1.528	76.7
✓			1.519	78.1
✓		✓	1.508	79.4
	✓		1.497	82.1
✓	✓		1.492	83.2
✓	✓	✓	1.489	83.9

Table 5: Effectiveness of D_{cls} and pseudo target labels.

D_{cls}	pseudo	$\mathcal{A}$-distance	Accuracy (%)
		1.503	80.6
✓		1.496	82.3
✓	✓	1.489	83.9

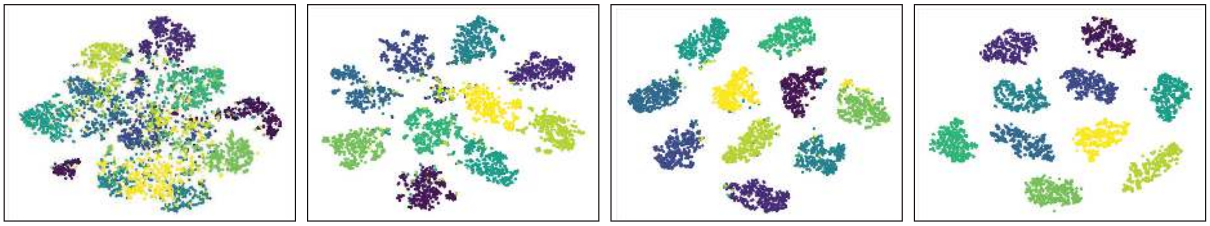
Classification on VisDA-2017

Dataset. The VisDA-2017 (Peng et al. 2017) challenge proposes a large-scale dataset for visual domain adaptation. The training domain is composed of synthetic renderings of 3D models. The validation domain is made up of photo-realistic images drawn from MSCOCO (Lin et al. 2014). Both domains contain the same 12 classes of objects.

Results. Table 3 reports the results on the VisDA-2017 cross-domain classification dataset. The ResNet-101 model pretrained on ImageNet acts as the baseline. Our approach achieves the highest accuracy among all adaptation approaches, and exceeds the baseline with a great margin. Under the condition that large domain shift exists, like transferring from synthetic objects to real images in this task, we think that the triplet loss and soft label play a critical role in excavating intermediate status between two domains.

Ablation Study

Metrics. Two metrics are employed. (1) $\mathcal{A}$-distance (Ben-David et al. 2010; Mansour, Mohri, and Rostamizadeh 2009) serves as a measure of cross-domain discrepancy. Inputted with extracted features of two domains, a SVM classifier is



(a) baseline

(b) w/ PM

(c) w/ PM & FM

(d) full model

Figure 4: The t-SNE (Maaten and Hinton 2008) visualization of target domain's feature distribution on the transfer task SVHN → MNIST under four different configurations. (Best viewed in color.)

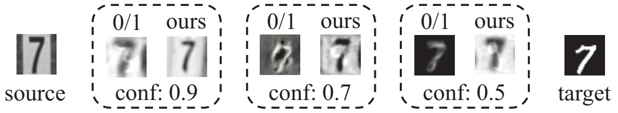


Figure 5: The generated images with confidence 0.9, 0.7 and 0.5 using 0/1 and our discriminator. (task: SV → MN)

used to classify the source and target domain features, and the generalization error is defined as ϵ. Then the $\mathcal{A}$-distance can be calculated as: $d_{\mathcal{A}} = 2(1-2\epsilon)$. (2) Classification accuracy on target domain serves as a measure of task-specific performance. In this part of experiments, both metrics are evaluated on the task A → W.

Effect of pixel-level and feature-level mixup. Table 4 examines the effectiveness of pixel-level mixup (PM) and feature-level mixup (FM). The first row only uses the images and feature embeddings from two domains for training, and it serves as the baseline. In the fourth row, feature-level mixup achieves notable improvement compared with baseline, since the domain-invariant latent space is facilitated to be more continuous in this configuration. In the fifth row, pixel-level mixup further enhance model's performance through guiding discriminator output soft scores between 0 and 1, which means it is an essential auxiliary scheme for feature-level mixup. In Figure 5, compared with traditional 0/1 discriminator, our discriminator leads to more source-like generated images, which means the domain discrepancy can be further narrowed via pixel-level mixup.

Effect of triplet loss. In Table 4, we evaluate another key component, *i.e.*, triplet loss (Tri). In the third and sixth rows, it can also be observed that model's performance is improved after adding the triplet loss to discriminator's training process, since this loss ease the convergence of domain discriminator. We further utilize t-SNE (Maaten and Hinton 2008) to visualize the feature distribution of target domain on the task SV → MN. As shown in Figure 4, the features of different classes are separated most clearly in the full model, *i.e.*, with domain mixup on two levels and triplet loss.

Effect of D_{cls} and pseudo target labels. In order to conduct category-aware alignment between source and target domains, the classification branch of discriminator D_{cls} and pseudo target labels are employed, and the effectiveness of them is examined in Table 5. After appending D_{cls}, classification accuracy increases by 1.7%, since this branch facilitates generated images to preserve the class information contained in inputs, which makes domain adaptation perform on the same categories of two domains. On such basis, pseudo target labels introduce the discriminative information of target domain to the adaptation process and make model's performance state-of-the-art.

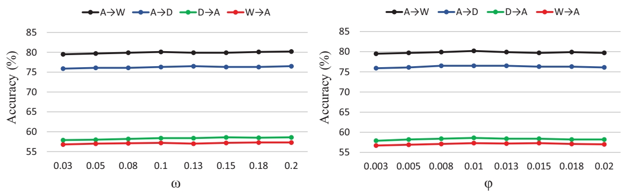


Figure 6: The sensitivity analysis of ω (left) and φ (right). Four hard-to-transfer tasks of Office-31 dataset are involved.

Sensitivity Analysis

In this section, we discuss our approach's sensitivity to hyper-parameters ω and φ which trade off among losses with different orders of magnitude. Four hard-to-transfer tasks of Office-31 dataset are used for evaluation. In Figure 6, it can be observed that the transfer performance is not sensitive to the variance of ω and φ near 0.1 and 0.01, respectively. In consequence, we can set ω and φ as 0.1 and 0.01 for all tasks, and the transfer performance should be satisfactory.

Conclusion

In this paper, we address the problem of unsupervised domain adaptation. A GAN-based architecture is constructed to transfer knowledge from source domain to target domain. In order to facilitate a more continuous domain-invariant latent space and fully utilize the inter-domain information, we propose the domain mixup on pixel and feature level. Extensive experiments on adaptation tasks with different extent of domain shift and data complexity demonstrate the predominant performance of our approach.

Acknowledge

This work was supported by National Science Foundation of China (61976137, U1611461). This work was also supported by SJTU-BIGO Joint Research Fund, and CCF-Tencent Open Fund.

References

Ajakan, H.; Germain, P.; Larochelle, H.; Laviolette, F.; and Marchand, M. 2013. Domain-adversarial neural networks. *CoRR* abs/1412.4446.

Ben-David, S.; Blitzer, J.; Crammer, K.; Kulesza, A.; Pereira, F.; and Vaughan, J. W. 2010. A theory of learning from different domains. *Machine Learning* 79(1-2):151–175.

Berthelot, D.; Raffel, C.; Roy, A.; and Goodfellow, I. 2019. Understanding and improving interpolation in autoencoders via an adversarial regularizer. In *ICLR*.

Bousmalis, K.; Silberman, N.; Dohan, D.; Erhan, D.; and Krishnan, D. 2017. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *CVPR*.

Ganin, Y., and Lempitsky, V. S. 2015. Unsupervised domain adaptation by backpropagation. In *ICML*.

Ganin, Y.; Ustinova, E.; Ajakan, H.; Germain, P.; Larochelle, H.; Laviolette, F.; Marchand, M.; and Lempitsky, V. 2016. Domain-adversarial training of neural networks. *JMLR* 17(1):2096–2030.

Ghifary, M.; Kleijn, W. B.; Zhang, M.; Balduzzi, D.; and Li, W. 2016. Deep reconstruction-classification networks for unsupervised domain adaptation. In *ECCV*.

Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A. C.; and Bengio, Y. 2014. Generative adversarial nets. In *NeurIPS*.

Gretton, A.; Borgwardt, K. M.; Rasch, M. J.; Schölkopf, B.; and Smola, A. J. 2012. A kernel two-sample test. *JMLR* 13:723–773.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*.

Hoffman, J.; Tzeng, E.; Park, T.; Zhu, J.; Isola, P.; Saenko, K.; Efros, A. A.; and Darrell, T. 2018. Cycada: Cycle-consistent adversarial domain adaptation. In *ICML*.

Hull, J. J. 1994. A database for handwritten text recognition research. *IEEE Trans. Pattern Anal. Mach. Intell.* 16(5):550–554.

Kang, G.; Zheng, L.; Yan, Y.; and Yang, Y. 2018. Deep adversarial attention alignment for unsupervised domain adaptation: The benefit of target expectation maximization. In *ECCV*.

Kingma, D. P., and Welling, M. 2013. Auto-encoding variational bayes. *CoRR* abs/1312.6114.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NeurIPS*.

Larsen, A. B. L.; Sønderby, S. K.; Larochelle, H.; and Winther, O. 2016. Autoencoding beyond pixels using a learned similarity metric. In *ICML*.

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

Lin, T.; Maire, M.; Belongie, S. J.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft COCO: common objects in context. In *ECCV*.

Liu, M., and Tuzel, O. 2016. Coupled generative adversarial networks. In *NeurIPS*.

Long, M.; Cao, Y.; Wang, J.; and Jordan, M. I. 2015. Learning transferable features with deep adaptation networks. In *ICML*.

Long, M.; Zhu, H.; Wang, J.; and Jordan, M. I. 2017. Deep transfer learning with joint adaptation networks. In *ICML*.

Long, M.; Cao, Z.; Wang, J.; and Jordan, M. I. 2018. Conditional adversarial domain adaptation. In *NeurIPS*.

Ma, X.; Zhang, T.; and Xu, C. 2019. Gcan: Graph convolutional adversarial network for unsupervised domain adaptation. In *CVPR*.

Maaten, L. V. D., and Hinton, G. 2008. Visualizing data using t-sne. *JMLR* 9(2605):2579–2605.

Mansour, Y.; Mohri, M.; and Rostamizadeh, A. 2009. Domain adaptation: Learning bounds and algorithms. In *Annual Conference on Learning Theory*.

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

Pan, S. J., and Yang, Q. 2010. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* 22(10):1345–1359.

Pan, S. J.; Tsang, I. W.; Kwok, J. T.; and Yang, Q. 2011. Domain adaptation via transfer component analysis. *IEEE Trans. Neural Networks* 22(2):199–210.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch. In *NeurIPS Workshop*.

Pei, Z.; Cao, Z.; Long, M.; and Wang, J. 2018. Multi-adversarial domain adaptation. In *AAAI*.

Peng, X.; Usman, B.; Kaushik, N.; Hoffman, J.; Wang, D.; and Saenko, K. 2017. Visda: The visual domain adaptation challenge. *CoRR* abs/1710.06924.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M. S.; Berg, A. C.; and Li, F. 2015. Imagenet large scale visual recognition challenge. *IJCV* 115(3):211–252.

Saenko, K.; Kulis, B.; Fritz, M.; and Darrell, T. 2010. Adapting visual category models to new domains. In *ECCV*.

Saito, K.; Ushiku, Y.; Harada, T.; and Saenko, K. 2018a. Adversarial dropout regularization. In *ICLR*.

Saito, K.; Watanabe, K.; Ushiku, Y.; and Harada, T. 2018b. Maximum classifier discrepancy for unsupervised domain adaptation. In *CVPR*.

Sankaranarayanan, S.; Balaji, Y.; Castillo, C. D.; and Chellappa, R. 2018. Generate to adapt: Aligning domains using generative adversarial networks. In *CVPR*.

Sun, B., and Saenko, K. 2016. Deep coral: Correlation alignment for deep domain adaptation. In *ECCV Workshop*.

Tran, L.; Sohn, K.; Yu, X.; Liu, X.; and Chandraker, M. 2019. Gotta adapt ’em all: Joint pixel and feature-level domain adaptation for recognition in the wild. In *CVPR*.

Tzeng, E.; Hoffman, J.; Zhang, N.; Saenko, K.; and Darrell, T. 2014. Deep domain confusion: Maximizing for domain invariance. *CoRR* abs/1412.3474.

Tzeng, E.; Hoffman, J.; Saenko, K.; and Darrell, T. 2017. Adversarial discriminative domain adaptation. In *CVPR*.

Xie, S.; Zheng, Z.; Chen, L.; and Chen, C. 2018. Learning semantic representations for unsupervised domain adaptation. In *ICML*.

Yan, H.; Ding, Y.; Li, P.; Wang, Q.; Xu, Y.; and Zuo, W. 2017. Mind the class weight bias: Weighted maximum mean discrepancy for unsupervised domain adaptation. In *CVPR*.

Yosinski, J.; Clune, J.; Bengio, Y.; and Lipson, H. 2014. How transferable are features in deep neural networks? In *NeurIPS*.

Zhang, H.; Cissé, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2018a. mixup: Beyond empirical risk minimization. In *ICLR*.

Zhang, W.; Ouyang, W.; Li, W.; and Xu, D. 2018b. Collaborative and adversarial network for unsupervised domain adaptation. In *CVPR*.