Image Classification Based on Unsupervised Domain Adaptation Methods

Yiju Yang

Submitted to the graduate degree program in Department of Electrical Engineering and Computer Science and the Graduate Faculty of the University of Kansas in partial fulfillment of the requirements for the degree of Master of Science.

Committee members

Taejoon Kim, Chairperson

Cuncong Zhong, Member

Andrew B Williams, Member

Guanghui Wang, Member

Date defended: May 24, 2021

The Thesis Committee for Yiju Yang certifies
that this is the approved version of the following thesis :

Image Classification Based on Unsupervised Domain Adaptation Methods

Taejoon Kim, Chairperson

Date approved:

Abstract

Convolutional Neural Networks (CNNs) have achieved great success in broad computer vision tasks. However, due to the lack of labeled data, many available CNN models cannot be widely used in many real scenarios or suffer from significant performance drop. To solve the problem of lack of correctly labeled data, we explored the capability of existing unsupervised domain adaptation (UDA) methods on image classification and proposed two new methods to improve the performance.

1. An Unsupervised Domain Adaptation Model based on Dual-module Adversarial Training: we proposed a dual-module network architecture that employs a domain discriminative feature module to encourage the domain invariant feature module to learn more domain invariant features. The proposed architecture can be applied to any model that utilizes domain invariant features for UDA to improve its ability to extract domain invariant features. Through the adversarial training by maximizing the loss of their feature distribution and minimizing the discrepancy of their prediction results, the two modules are encouraged to learn more domain discriminative and domain invariant features respectively. Extensive comparative evaluations are conducted and the proposed approach significantly outperforms the baseline method in all image classification tasks.

2. Exploiting maximum classifier discrepancy on multiple classifiers for unsupervised domain adaptation: The adversarial training method based on the maximum classifier discrepancy between the two classifier structures has been applied to the unsupervised domain adaptation task of image classification. This method is straightforward and has achieved very good results. However, based on our observation, we think the structure of two classifiers, though simple, may not explore the full power of the algorithm. Thus, we propose to add more classifiers to the model. In the proposed method, we construct a discrepancy loss function for multiple classifiers following the principle that the classifiers are different from each other. By constructing this loss function,

we can add any number of classifiers to the original framework. Extensive experiments show that the proposed method achieves significant improvements over the baseline method.

Acknowledgements

Throughout the writing of this thesis, I have received a great deal of support and assistance.

I would first like to thank my supervisor, Professor Guanghui Wang, whose expertise was invaluable in formulating the research questions and methodology. Your insightful feedback pushed me to sharpen my thinking and brought my work to a higher level.

I would also like to thank my advisor, Professor Taejoon Kim, for his valuable guidance throughout my thesis defense preparation.

I would also like to thank my committee members, each of whom has provided patient advice and guidance throughout the research process. Thank you all for your unwavering support.

In addition, I would like to thank my parents for their wise counsel and sympathetic ear. You are always there for me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the last ten years, with the emergence of the Activation function ReLU (which solves the problem of gradient disappearance in deep neural networks) and GPUs are used to accelerate network computations, deep neural networks have been widely used in various fields of computer science. As AlexNet won the ImageNet competition in 2012, people found that its powerful performance far surpassed the second algorithm SVM at the time. Because of ALexNet, Convolutional Layer and the deep Convolutional Neural Network (CNN) structures began to receive extensive attention from researchers.

At present, thanks to the excellent ability of CNN in feature extraction, networks such as VGGNet and ResNet have achieved great success in image classification tasks. In addition to the excellent network itself is a key factor, there is another important factor that is the labeled data. Since most of the current methods are based on supervised learning, correctly labeled data has become another key factor in whether these algorithms can be applied to real-world scenarios. If we meet the above two factors at the same time, then our model can be used in almost all scenarios. However, in practical applications, it is difficult for us to meet the conditions of the data. The specific reasons often come from the following two aspects:

1. Scene changes. Scene changes are the easiest thing to happen in practical applications. In addition to the changes caused by performing tasks in different places, there will also be the changes in weather, movement of objects, withering of vegetation, etc., which can cause scene changes in fixed views and locations. If we only use data from certain scenarios to train a model, then the performance of the model will greatly reduce when we perform tasks in new scenarios. Therefore, when a model performs tasks in a brand-new environment, we need to supplement the

model with a large amount of labeled data of the new scene.

2. Labeled data in professional areas. In many professional fields, such as medical image processing, agricultural image analysis, geological image analysis, deep-sea exploration and other application fields, the task of labeling data will be much more difficult. The main reason is that the people who can label these images can only be people in related fields, and this makes the labeled data very precious and expensive.

Motivation & Challenge: In order to enable those existing excellent models to be used in more scenarios and to reduce costs as much as possible, unsupervised domain adaptation has become an important idea to solve the problem of labeling data. The core idea of the unsupervised domain adaptation method is to fuse two different domains by aligning their distributions. In these two distributions, we have a source domain dataset and a target domain dataset, and the classes in these two data sets are the same. The data in the source domain dataset is labeled, while the data in the target domain dataset is unlabeled. The two data sets will have some commonalities, so they are not two completely different domains. Therefore, how to align the two domains to minimize the domain shift has become one of the main challenges of the unsupervised domain adaptation method. Under this kind of thinking, we hope that by using unlabeled data, the performance of the model in processing the target domain tasks can reach a level close to or even beyond supervised learning.

Contribution: In my thesis, we proposed two unsupervised domain adaptation methods, and used these methods to solve image classification tasks. In the first chapter, we proposed a method based on dual-module adversarial training. In the second chapter, we propose a method of maximum classifier discrepancy based on the structure of multiple classifiers. These two methods not only have a significant performance improvement compared to the previous method, but also have reached the state-of-the-art performance level.

Chapter 2

Related Works

Domain adaptation is a commonly used technique in many computer vision tasks to improve the generalization ability of a model trained on a single domain. In this chapter, we describe some existing domain adaptation methods.

2.1 Learn domain invariant features.

Recently, (Chen et al., 2020b) explored what enables the contrastive prediction tasks to learn useful representations. (Carlucci et al., 2019) learns the semantic labels in a supervised fashion, and broadens its understanding of the data by learning from self-supervised signals how to solve a jigsaw puzzle on the same images.

2.2 Distance-based methods.

Aligning the distribution between the source domain and the target domain is a very common method in solving unsupervised domain adaptation problems. Maximum Mean discrepancy (MMD) (Gretton et al., 2012; Long et al., 2017; Tzeng et al., 2014; Long et al., 2015) is a method of measuring the difference of two distributions. DAN (Long et al., 2015) explored the multi-core version of MMD to define the distance between two distributions. JAN (Long et al., 2017) learned a transfer network by aligning the joint distributions of multiple domain-specific layers across the domains based on a joint maximum mean discrepancy (JMMD) criterion. (Long et al., 2016) enabled the classifier adaptation by plugging several layers into the deep network to explicitly learn the residual function with reference to the target classifier. CMD (Zellinger et al., 2017) is a metric on the

set of probability distributions on a compact interval.

To solve the problem of unbalanced datasets, Deep Asymmetric Transfer Network (DATN) (Wang et al., 2018) proposed to learn a transfer function from the target domain to the source domain and meanwhile adapting the source domain classifier with more discriminative power to the target domain. DeepCORAL (Sun & Saenko, 2016) builds a specific deep neural network by aligning the distribution of second-order statistics to limit the invariant domain of the top layer. (Chen et al., 2020a) proposed a Higher-order Moment Matching (HoMM) method to minimize the domain discrepancy.

2.3 Adversarial methods.

Adversarial training is another very effective method to transfer domain information. Inspired by the work of gradient reversal layer (Ganin & Lempitsky, 2015), a group of domain adaptation methods has been proposed based on adversarial learning. RevGrad (Ganin & Lempitsky, 2015) proposed to learn the global invariant feature by using a discriminator that is used to reduce the discriminative features in the domain. Deep Reconstruction-Classification Networks (DRCN) (Ghifary et al., 2016) jointly learned a shared encoding representation for supervised classification of the labeled source data, and unsupervised reconstruction of the unlabeled target data. (Bousmalis et al., 2016) extracted image representations that are partitioned into two subspaces. Adversarial Discriminative Domain Adaptation (ADDA) (Tzeng et al., 2017) trained two feature extractors for the source and target domains respectively, to generate embeddings to fool the discriminator.

Maximum Classifier Discrepancy (MCD) (Saito et al., 2018) proposed to explore task-specific decision boundaries. CyCADA (Hoffman et al., 2018) introduced a cycle-consistency loss to match the pixel-level distribution. SimeNet (Pinheiro, 2018) solved this problem by learning the domain invariant features and the categorical prototype representations. CAN (Kang et al., 2019) optimized the network by considering the discrepancy of the intra-class domain and the inter-class domain. Graph Convolutional Adversarial Network (GCAN) (Ma et al., 2019) realized the unsupervised domain adaptation by jointly modeling data structure, domain label, and class label in a unified

deep model. (Gong et al., 2019) proposed a domain flow generation (DLOW) model to bridge two different domains by generating a continuous sequence of intermediate domains flowing from one domain to the other. (Cai et al., 2019) employed a variational auto-encoder to reconstruct the semantic latent variables and domain latent variables behind the data. Drop to Adapt (DTA) (Lee et al., 2019) leveraged adversarial dropout to learn strongly discriminative features by enforcing the cluster assumption. Instead of representing the classifier as a weight vector, (Lu et al., 2020) modeled it as a Gaussian distribution with its variance representing the inter-classifier discrepancy.

Chapter 3

Dual-Module Adversarial training

3.1 Introduction

With the advent of gradient reversal layers (GRL) from Domain-Adversarial Training of Neural Networks (DANN) Ganin & Lempitsky (2015), more and more people realize that adversarial training has a significant effect on aligning the feature of the source and target domains. DANN Ganin & Lempitsky (2015) extracts the global invariant features by training a domain discriminator to fool the feature extractor. The domain discriminator is a component composed of several fully connected layers. Its function is to distinguish the input data from the source domain or the target domain. If the discriminator could not recognize the extracted feature map, it means that the extracted feature map comes from the common space of these two domains. The global invariant features are from this space, and DANN utilizes a classifier for adversarial training to ensure the effectiveness of the features learned by the network. In addition to the adversarial training method based on GRL, there are many other adversarial training methods based on generative adversarial nets (GANs) Goodfellow et al. (2014); Liu & Tuzel (2016); Tran et al. (2017); Hu et al. (2018). Most of them have one thing in common, i.e., they adapt to the target domain by learning global invariant features.

Taking DANN Ganin & Lempitsky (2015) as an example, although it has the advantage to use a discriminator to encourage the model to learn invariant features, there is also a bottleneck. When we only rely on the discriminator to control the extraction ability of the domain invariant features, the conditions to extract domain invariant features will become very limited. When the extracted domain invariant features are strong enough to fool the discriminator, it will become difficult for

the model to further improve its ability to extract the domain invariant features. In addition, when we only use the source domain labeled data to adjust the classifier, the extracted features will be more biased towards the source domain, which further limits its performance on the target task. This challenge also exists in other methods that employ the domain discriminator to extract the domain invariant features.

In order to further improve the extraction ability of domain invariant features, we propose a dual-module architecture to solve this challenge. The proposed network is composed of a discriminative feature learning module and a domain invariant feature learning module, and the domain discriminative feature module is employed to encourage a domain invariant feature module to learn more domain invariant features.

3.2 Method

3.2.1 Network Structure

In this section, we will elaborate on the proposed network structure in detail. As shown in Figure 3.1, our network employs a dual-module structure. M1 is the domain invariant feature module, and M2 is the domain discriminative feature module. M1 is composed of a feature extractor G1, a discriminator D1, two classifiers C1 and C2, and a Linear transformation layer T1. M2 consists of a feature extractor G2, a discriminator D2, two classifiers C3 and C4, and a Linear transformation layer T2. Since the loss of our dual-module architecture needs to calculate the discrepancy of the two feature distributions, we need an independent linear transformation layer to convert the feature map into a feature distribution. For each module, We embed a linear transformation layer after the feature extractor. For the linear transformation layer, we set its input and output sizes as that of the feature extractor, so as to ensure that there is no information loss in this process.

In addition, the domain discriminators in these two modules have completely different functions. The discriminator D1 plays the same role as the discriminator in DANN to fuse the two domains. However, the discriminator D2 is employed to separate the two domains. Therefore, M1

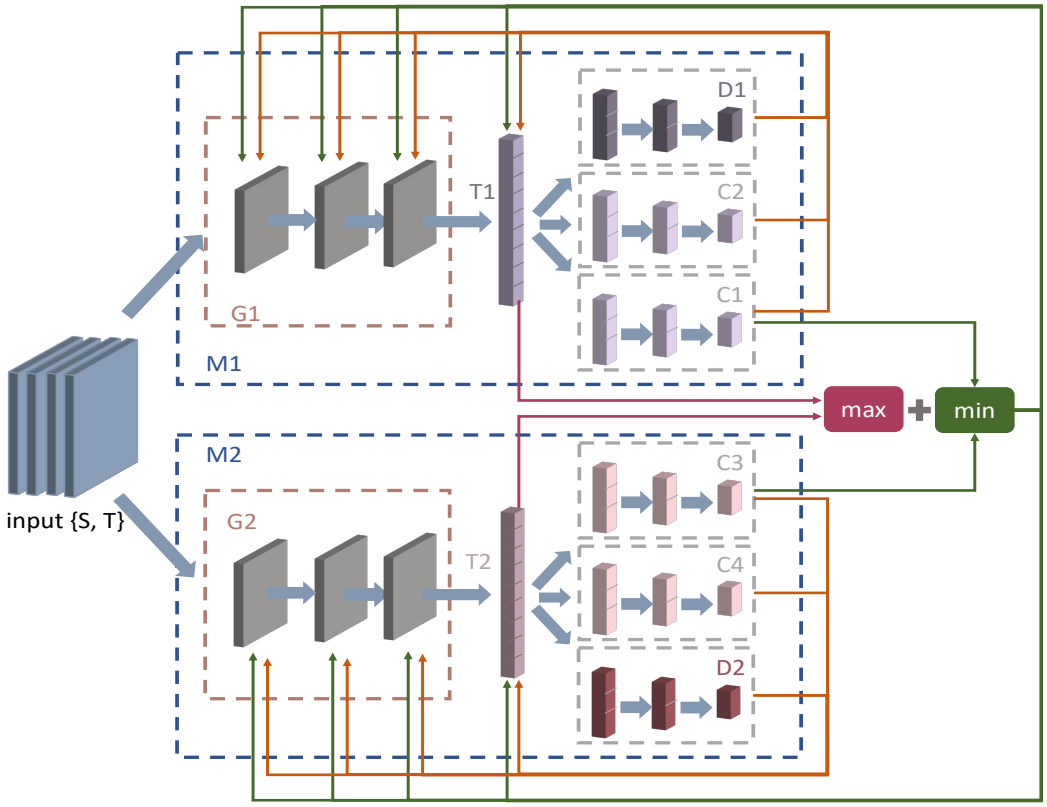


Figure 3.1: Network Structure.
The proposed network architecture has two modules. M1 learns the domain invariant features, and M2 learns the domain discriminative features. G1 is the feature extractor of M1, and G2 is the feature extractor of M2. T1 and T2 are learner transformation layers. D1 and D2 are domain discriminators. C1, C2, C3, and C4 are classifiers. Orange lines denote the backward process of training steps 1 and 2, and Green lines denote the backward process of training step 3.

learns domain invariant features, while M2 learns the domain discrimination features. Please note that the sub-components used in our two modules have exactly the same structure. For example, G1 and G2 have the same structure, D1 and D2 are the same, and C1, C2, C3, and C4 are the same.

3.2.2 Training Steps

Step 1: Our model learns the decision boundary by using MCD Saito et al. (2018). Learning the decision boundary is essential to learning the discriminative feature. The main reason we put this as the first step is to avoid conflict between the learning process of the decision boundary and the domain invariant feature learning. In order to bring our model closer to the ideal state of domain fusion, our subsequent steps can effectively reduce the redundant domain discriminative features obtained from the decision boundary, thereby reducing the negative impact of the conflict. Following the setting of Saito et al. (2018), we fix the number of iterations to 4 for learning the boundary in target domain in all our experiments. Since the function of the linear transformation layer T is to convert the feature map into a feature distribution, our linear transformation layer only updates the parameters when the feature extractor updates the parameters.

Step 2: In this step, we continue to train the two modules separately. Taking the M1 module as an example, we use the adversarial loss of DANN for training. In other words, we train the model according to the training method from the original algorithm. After this step, the two modules begin to have some differences. This step is necessary for the proposed dual-module structure. For the algorithms that do not use MCD for pre-processing, this will be the first step in the entire training process.

For M1, we conduct adversarial training through gradient reversal layer (grl) to learn the domain invariant features.

$$L_C = L_{C1}(f_\theta(X_s), Y_s) + L_{C2}(f_\theta(X_s), Y_s) \tag{3.1}$$

$$L_{M1} = L_C + grl(L_{D1}(f_\theta(X_s))) + grl(L_{D1}(f_\theta(X_t))) \tag{3.2}$$

where L_{M1} is the total loss of the whole M1 module, L_C is the total loss of two classifiers, L_{C1}, L_{C2},

and L_{D1} are the cross-entropy loss for the classifiers and discriminator.

For M2, we don not apply gradient reversal layer, so the discriminator will prompt the feature extractor to learn the domain discriminative feature.

$$L_C = L_{C3}(f_\theta(X_s), Y_s) + L_{C4}(f_\theta(X_s), Y_s) \tag{3.3}$$

$$L_{M2} = L_C + L_{D2}(f_\theta(X_s)) + L_{D2}(f_\theta(X_t)) \tag{3.4}$$

where L_{M2} is the total loss of the whole M2 module, L_C is the total loss of the two classifiers, L_{C3}, L_{C4}, and L_{D2} are the cross-entropy loss for the classifiers and discriminator.

Step 3: In this step, we conduct an adversarial loss function L for our two modules. We input the same set of data into the two modules and extract the output from the transformation layer T and the classifier C. We use the linear transformation layer to convert the feature maps into feature distributions, and calculate the discrepancy between the two modules. At the same time, we use C1 and C3 to predict the results for the same input and calculate the discrepancy between the two modules. We use a gradient reversal layer (*grl*) to maximize the discrepancy of the feature distributions. At the same time, we minimize the discrepancy of the prediction results. Our adversarial loss function is to play a Min-Max game with these two discrepancies.

$$grl(dis(t)) = \max(dis(t_1^s, t_2^s) + dis(t_1^t, t_2^t)) \tag{3.5}$$

$$dis(c) = \min(dis(c_1^s, c_3^s) + dis(c_1^t, c_3^t)) \tag{3.6}$$

$$L = grl(dis(t)) + dis(c) \tag{3.7}$$

where L is the total loss, $dis()$ is the discrepancy loss. t_1^s means the output is from T1 and the input is from the source domain, and t_2^s means the output from T2 and input from the source domain. t_1^t means the output from T1 and input from the target domain, and t_2^t means the output from T2 and input from the target domain. c_1^s means the probability output from C1 and takes input from the source domain, and c_1^t means the probability output from C1 and takes input from the target

domain. c_3^s means the probability output from C3 and takes input from the source domain, and c_3^t means the probability output from C3 and takes input from the target domain.

3.3 Experiments

We conducted extensive experiments to evaluate the proposed architecture and the effect of different components in the architecture. We conduct our experiments on three digits datasets, two traffic sign datasets, and one object classification dataset. In the following, **OURS** mentioned in the results refers to the case where only the dual-module structure is used. **MCD+DANN** refers to the case where MCD is directly employed to solve the imbalance problem in DANN. **OURS+1M** refers to the case where MCD is only used by the M1 module. **OURS+2M** refers to the case where MCD is used by both M1 and M2 modules. We also compare the performance with DANN and MCD as the baselines.

3.3.1 Experiments on Digits and Traffic Signs Datasets

In this section, we evaluate our model using the following five datasets: MNIST LeCun & Cortes (2010), Street View House Numbers (SVHN) Netzer et al. (2011), USPS Hull (1994), Synthetic Traffic Signs (SYN SIGNS) Moiseev et al. (2013), and the German Traffic Signs Recognition Benchmark (GTSRB) Stallkamp et al. (2011).

MNIST: The dataset contains images of digits 0 to 9 in different styles. It is composed of $60,000$ training and $10,000$ testing images.

USPS: This is also a digit dataset with $7,291$ training and $2,007$ testing images.

SVHN: Another digit dataset with $73,257$ training, $26,032$ testing, and $53,1131$ extra training images.

SYN SIGNS: This is a synthetic traffic sign dataset, which contains $100,000$ labeled images, and 43 classes.

GTSRB: A dataset for German traffic signs recognition benchmark. The training set contains

39,209 labeled images and the test set contains 12,630 images. It also contains 43 classes.

We evaluate the unsupervised domain adaptation model on the following four transfer scenarios:

- SVHN → MNIST
- USPS → MNIST
- MNIST →USPS
- SYNSIG → GTSRB

During the experiments, we employ the CNN architecture and the input size used in Saito et al. (2018). We used mini-batch stochastic gradient descent (SGD) to optimize our model and set the learning rate at 0.002 in all experiments. We follow DANN Ganin & Lempitsky (2015) and employ the SGD training schedule for the part of learning domain invariant feature: the learning rate adjusted by $\eta_p = \frac{\eta_0}{(1+\alpha p)^\beta}$, where p denotes the process of training iterations that is normalized in [0, 1], and we set η_0 = 0.002, α = 10, and β = 0.75; the hyper-parameter λ is initialized at 0 and is gradually increased to 1 by $\lambda_p = \frac{2}{1+exp(-\gamma p)} - 1$, where we set γ = 10. For the maximun classifier discrepancy, we set the hyper-parameter k = 4 in all experiments. We set the batch size to 128 in all experiments. We follow the protocol of unsupervised domain adaptation and do not use validation samples to tune the hyper-parameters.

We present the digit classification and sign classification performance in Table 4.1. From the table, it is clear that the proposed method outperforms previous models in all settings, where **OURS+2M** is the top-performing variant. In order to explore the direct impact of MCD on DANN, we combined them and compared the experimental results with the state-of-the-art. We can find that the performance of **MCD+DANN** is lower than MCD in both SVHN→MNIST and SYNSIG→GTSRB tasks. The result demonstrates that when MCD acts directly on DANN, it sometimes may cause conflict with DANN, while the structure of **OURS+2M** can effectively resolve this conflict. Compared with MCD (baseline), we obtain an improvement of 3.1% in

Method	SVHN to MNIST	MNIST to USPS	USPS to MNIST	SYNSIG to GTSRB
Source only	67.1	79.4	63.4	85.1
DANN Ganin & Lempitsky (2015)	71.1	85.1	73.0±0.2	88.7
ADDA Tzeng et al. (2017)	76.0±1.8	-	90.1±0.8	-
CoGAN Liu & Tuzel (2016)	-	-	89.1±0.8	-
PixelDA Bousmalis et al. (2017)	-	95.9	-	-
ASSC Haeusser et al. (2017)	95.7±1.5	-	-	82.8±1.3
UNIT Liu et al. (2017)	90.5	96.0	93.6	-
CyCADA Hoffman et al. (2018)	90.4±0.4	95.6±0.2	96.5±0.1	-
GTA Sankaranarayanan et al. (2018)	92.4±0.9	95.3±0.7	90.8±1.3	-
DeepJDOT Bhushan Damodaran et al. (2018)	96.7	95.7	96.4	-
SimNet Pinheiro (2018)	-	96.4	95.6	-
GICT Qin et al. (2019)	98.7	96.2	96.6	-
STAR Lu et al. (2020)	98.8±0.05	97.8±0.1	97.7±0.05	95.8±0.2
MCD Saito et al. (2018)	96.2±0.4	96.5±0.3	94.1±0.3	94.4±0.3
MCD+DANN	91.4±0.2	97.3±0.3	96.8±0.1	90.7±0.2
ours	98.9±0.1	95.1±0.4	96.1±0.2	91.1±0.2
ours+1M	98.3±0.1	97.1±0.2	97.0±0.1	90.8±0.2
ours+2M	**99.3±0.1**	**98.0±0.4**	**97.7±0.1**	**97.0±0.2**

Table 3.1: The performance on digit classification and sign classification.
We report the mean and the standard deviation of the accuracy obtained over 5 trials.

SVHN→MNIST, 1.5% in MNIST→USPS, 3.6% in USPS→MNIST, and 2.6% in SYNSIG→GTSRB. In addition, our model outperforms the state-of-the-art methods on all tasks.

3.3.2 Experiments on VisDA Classification Dataset

We further evaluate our model on the large VisDA-2017 dataset Peng et al. (2017). The VisDA-2017 image classification is a 12-class domain adaptation dataset used to evaluate the adaptation from synthetic-object to real-object images. The source domain consists of 152,397 synthetic images, where 3D CAD models are rendered from various conditions. The target domain consists of 55,388 real images taken from the MS-COCO dataset Lin et al. (2014).

In this experiment, we employ Resnet-18 He et al. (2016) as our feature extractor and the parameters are adopted from the ImageNet pre-trained model. The pre-trained model of our Resnet-

Method	plane	bcycl	bus	car	horse	knife	mcycl	person	plant	sktbrd	train	truck	mean
Source Only	30.2	4.3	27.9	60.6	31.0	2.1	82.7	7.4	67.7	12.9	79.4	2.3	40.3
DANN Ganin & Lempitsky (2015)	72.3	53.1	64.7	31.8	58.2	14.3	80.7	**60.0**	70.0	41.4	**89.7**	20.7	55.9
MCD Saito et al. (2018)	82.2	18.7	86.6	**62.1**	73.6	**41.0**	89.1	58.9	80.7	64.3	74.2	12.5	63.1
OURS	83.6	60.0	64.1	56.4	65.8	12.5	91.4	39.3	66.7	55.0	78.3	31.2	60.1
OURS + 1M	85.2	58.5	76.5	47.1	73.5	24.7	89.3	58.9	75.0	62.2	80.1	**32.0**	63.4
OURS + 2M	**86.0**	**61.5**	**88.3**	61.6	**83.8**	6.7	**92.9**	56.8	**89.9**	**68.8**	87.3	23.0	**69.2**

Table 3.2: The performance on VisDA-2017.

Results of unsupervised domain adaptation on VisDA2017 Peng et al. (2017) image classification track. The accuracy is obtained by fine-tuning ResNet-18 He et al. (2016) model pre-trained on ImageNet Deng et al. (2009). This task evaluates the adaptation capability from synthetic CAD model images to real-world MS COCO Lin et al. (2014) images. Our model achieves the best performance in most categories.

18 comes from Pytorch Paszke et al. (2017) and all experimental implementations are based on Pytorch.

The input images are of the size 224×224. First, we resize of the input image to 256, and then crop the image to 224×224 in the center. When we train the model using only the source domain, we just modify the output size of the original last fully connected layer to a size that conforms to VisDA-2017 Peng et al. (2017). In other tasks, we utilize a three-layer fully connected layer structure to replace the one-layer fully connected layer structure of the original classifier. For algorithms that require a discriminator, we employ a discriminator with a three-layer fully connected layer structure. In order to eliminate the interference factors, except for the source only, all other algorithms use the same classifier and the same discriminator. We uniformly use SGD as the optimizer for training, and use 1×10^{-3} for the learning rate of all methods. We use 64 as the batch size for training.

The results on VisDA-2017 are shown in Table 3.2. We can see that our model achieves an accuracy much higher than previous methods. In addition, our method performs better than the source only model in all classes, whereas MCD and DANN perform worse than the source only model in some classes such as train, motorcycle, and car. Same as the last experiment, **OURS+2M** is the top-performing variant. On average, we obtain an improvement of 6.1% compared with MCD, and 13.3% compared with DANN. Our model also achieves the best performance in eight out of the twelve categories in this experiment.

dif-Module	ddf-Module	dif-MCD	ddf-MCD	S→M	M→U	U→M	S-SIG→GTSRB	Avg
√				71.1	85.1	73.0±0.2	88.7	79.5
√		√		91.4±0.2	97.3±0.3	96.8±0.1	90.7±0.2	94.1
√	√			98.9±0.1	95.1±0.4	96.1±0.2	91.1±0.2	95.3
√	√	√		98.3±0.1	97.1±0.2	97.0±0.1	90.8±0.2	95.8
√	√	√	√	**99.3±0.1**	**98.0±0.4**	**97.7±0.1**	**97.0±0.2**	**98.0**

Table 3.3: Ablation study

Ablation study of our method for unsupervised domain adaptation on digital and traffic sign datasets.

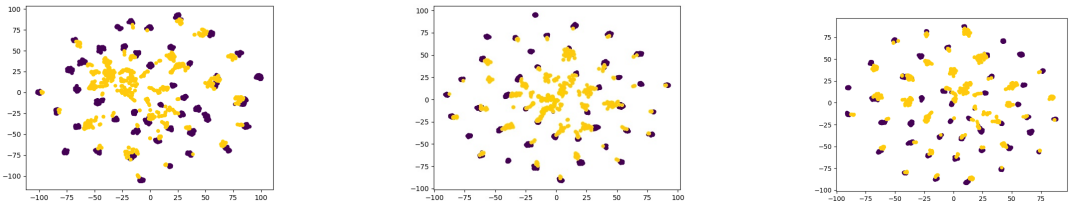
3.3.3 Ablation Study

We conducted ablation studies based on digital and traffic sign datasets with the same unsupervised domain adaptation setting as **subsection 4.1**. The algorithm we proposed has two modules, and each module has two partial components, namely **dif-module**, **ddf-module**, **dif-MCD** and **ddf-MCD**. Therefore, we design the ablation study to test the influence of each component on the overall algorithm performance. **dif-module** refers to the component used to learn domain invariant features in the model. This is a necessary module in the algorithm and also our baseline. **ddf-module** refers to the component used to learn domain discriminative features in the model. The function of this module is to expand the discrepancy in feature distribution during training. **dif-MCD** refers to the use of MCD to align the class distribution of domain invariant feature module classifiers, and **ddf-MCD** refers to the use of MCD to align to align the class distribution of domain discriminative feature module classifiers.

As the table 3.3 shows, the performance of dual-module adversarial training has a significant improvement over using a single domain adaptation method. There are two most intuitive examples. **1.** the performance of *DANN* with dual-module adversarial training is 18.5% higher than the one without dual-module adversarial training. **2.** the performance of $DANN + MCD$ with dual-module adversarial training is 3.9% higher than the original single module.

3.3.4 Visualization

In Figure 3.2, we used T-SNE van der Maaten & Hinton (2008) to visualize the models we trained. We chose the task that transfer SYN SIGNS Moiseev et al. (2013) to GTSRB Stallkamp et al. (2011). SYN SIGNS is the source domain, and GTSRB is the target domain. After training, we



Source Only Adapted: Ours Adapted: Ours+2M

Figure 3.2: Visualization by using T-SNE.

We take 2000 images from the task SYN SIGNS $\rightarrow$ GTSRB. Through visualization, we can easily find that our proposed method can fuse the source domain with the target domain well.

selected 2000 data for visualization, where 1000 images from the source domain and 1000 images from the target domain. It can be clearly seen from the figures that, compared to the Source Only method, our proposed model has a significant effect on reducing the domain shift, especially for the **OURS+M2** variant.

Chapter 4

Multi-Classifier Discrepancy

4.1 Introduction

Recently, Maximum Classifier Discrepancy (MCD) Saito et al. (2018) was proposed as an adversarial training framework. In this framework, the two classifiers are trained by using the maximum classifier discrepancy, and the feature extractor parameters are adjusted inversely through the decision boundary of the two classifiers so that the source domain and the target domain are fused together. In MCD, only two classifiers are used. Is it possible to employ more classifiers and will the performance benefit from multiple classifiers? The paper will answer this question and explore an approach to adding multiple classifiers to the system.

In this study, we propose a very straightforward method to train multiple classifiers based on the MCD Saito et al. (2018) framework. We also compare the performance difference between the multi-classifier structure and the 2-classifier structure.

4.2 Method

We only consider the close-set unsupervised domain adaptation problem. Suppose we have a source domain $D_s = \{(X_s, Y_s)\} = \{(x_s^i, y_s^i)\}_{i=1}^{n_s}$ with n_s labeled samples and a target domain $D_t = \{(X_t)\} = \{(x_t^i)\}_{i=1}^{n_t}$ with n_t unlabeled samples. The two domains share the same label space $Y = \{1, 2, 3, ..., K\}$, where K is the number of categories. We assume that the source sample x_s belongs to the source distribution P_s, and the target sample x_t belongs to the target distribution P_t, where $P_s \neq P_t$. Our goal is to train a classifier $f_\theta(x)$ that can minimize the target risk $\varepsilon_t = E_{x \in D_t}[f_\theta(x) \neq$

$y_t]$, where $f_\theta(x)$ represents the output of the deep neural network, and θ represents the model parameters to be learned.

4.2.1 Discrepancy Loss for 2-classifier

We follow the discrepancy loss in Saito et al. (2018) and use the absolute value of the difference between the probability outputs of the two classifiers as the discrepancy loss:

$$dis(p^1, p^2) = \frac{1}{K} \sum_{k=1}^{K} |p_k^1 - p_k^2| \tag{4.1}$$

where p^1 and p^2 are the probability outputs of the two classifiers respectively, which are the prediction scores for all the categories, and K is the number of categories, and p_k^1 and p_k^2 are the specific values of their k_{th} category.

4.2.2 Discrepancy Loss for multi-classifier

We conduct the Discrepancy Loss of the multi-classifiers $Dis()$ based on the principle that the classifiers are different from each other. In our training step, we need to maximize the difference between the classifiers. If our classifiers cannot maintain the principle of mutual difference, then there will be a situation where the overall discrepancy is maximized but some local discrepancy is close to zero. Some classifier parameters will become the same as some classifiers in this process. If the parameters of multiple classifiers tend to be the same when training multi-classifiers, this will lead to the collapse of the model training. This conflict will make the performance of the model even worse.

The following three Discrepancy Losses are designed for the cases of 3 classifiers, 4 classifiers, and n classifiers.

For 3 classifiers:

$$Dis(p^1, p^2, p^3) = dis(p^1, p^2) + dis(p^1, p^3) + dis(p^2, p^3)$$

For 4 classifiers:

$$\begin{aligned} Dis(p^1,p^2,p^3,p^4) = dis(p^1,p^2)+dis(p^1,p^3)+dis(p^1,p^4) \\ +dis(p^2,p^3)+dis(p^2,p^4) \\ +dis(p^3,p^4) \end{aligned}$$

For n classifiers:

$$\begin{aligned} Dis(p^1,\ldots,p^n) = dis(p^1,p^2)+dis(p^1,p^3)+\cdots+dis(p^1,p^n) \\ +dis(p^2,p^3)+\cdots+dis(p^2,p^n) \\ +dis(p^3,p^4)+\cdots+dis(p^3,p^n) \\ +\cdots+dis(p^{n-1},p^n) \end{aligned}$$

where n is the number of classifiers, and p^1, p^2,..., p^n are the probability outputs of the n classifiers respectively.

4.2.3 Training Steps

Our proposed improvement is based on the framework of two classifiers. Therefore, when we use multiple classifiers, we replace the original discrepancy loss with the multi-classifier discrepancy loss under the original framework.

Step 1: We directly use the source domain data to train the model once so that the model can initially get some source domain information (features). After this step, the decision boundary from each classifier can classify the features extracted from the source domain.

$$\begin{aligned} Loss_1 = L_{C1}(f_{\theta 1}(X_s),Y_s)+L_{C2}(f_{\theta 2}(X_s),Y_s)+\cdots \\ +L_{Cn}(f_{\theta n}(X_s),Y_s) \end{aligned} \tag{4.2}$$

where $C1, C2, \ldots, Cn$ denote the 1st, 2nd, and nth classifiers, respectively. $L()$ is the CrossEntropy Loss function.

Step 2: In this step, we use both the source and target domain data to train our model. During the training, we fix the parameters of the feature extractor and only adjust the parameters of the classifiers. When we train using the source domain data, we follow the same way as the first step. When we train using the target domain data, we cannot perform supervised training since they have no labels. We employ the discrepancy loss for the target domain training at this step. We use the n-classifier discrepancy loss to maximizes the discrepancy between the results of the classifiers predicting the target domain. We maximize the discrepancy by reverse the gradient.

For the source data:

$$\begin{aligned} Loss_s = L_{C1}(f_{\theta 1}(X_s), Y_s) + L_{C2}(f_{\theta 2}(X_s), Y_s) + \cdots \\ + L_{Cn}(f_{\theta n}(X_s), Y_s) \end{aligned} \tag{4.3}$$

For the target data:

$$Loss_t = Dis(f_{\theta 1}(X_t), \ldots, f_{\theta n}(X_t)) \tag{4.4}$$

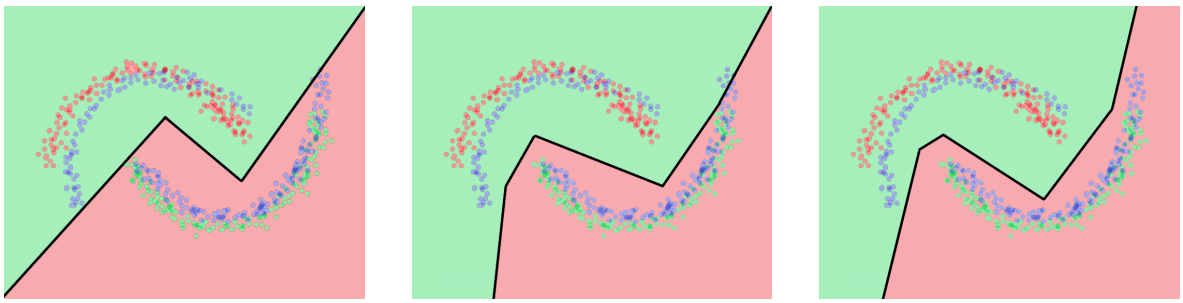
For joint training:

$$Loss_2 = Loss_s - Loss_t \tag{4.5}$$

Step 3: In this step, we minimize the discrepancy in the prediction results of the features of the target domain extracted by the feature extractor. During this process, we fix the n-classifier parameters and update the CNN's parameters by back-propagation. Since the parameters of the classifiers are fixed, in order to minimize the prediction discrepancy of the n-classifier, the CNN must adjust the parameters so that the features extracted from the target domain are consistent with the features obtained from the source domain, then they can achieve similar prediction results with totally different parameters.

$$Loss_3 = Dis(f_{\theta 1}(X_t), ..., f_{\theta n}(X_t)) \tag{4.6}$$

The source domain is trained by the labeled data, even when the classifier has the maximum discrepancy to the target domain, it will not have much impact when detecting the source domain.



a) Source Only b) Adapted: MCD c) Adapted: Ours

Figure 4.1: Experiments on the toy dataset.
The red points and the green points are the two classes of source domain data, and the blue points are the target domain data. The dividing line in the middle is the decision boundary. We can clearly see that the method we proposed is more effective than others. a) is the model trained only on source samples. b) is the original MCD Lu et al. (2020) with the 2-classifier structure. c) is our proposed method with a 3-classifier structure.

4.3 Experiments

4.3.1 Experiments on the Toy Dataset

In this experiment, we compared our proposed method with MCD Lu et al. (2020) and the Source Only method. The purpose is to compare the decision boundaries of different methods under the same task. We followed the same experimental setting as MCD Lu et al. (2020). We generated 300 source and target domain data and only gave labels to the source domain data. In Fig 4.1, it is very obvious that the decision boundary under multi-classifiers has better unsupervised classification capabilities than MCD with a 2-classifier structure and the Source Only method. The decision boundaries are drawn considering both the source and target samples. The outputs of three or multiple classifiers make nearly the same prediction for the target samples, and they classified most target samples correctly, so we randomly pick one of them for illustration in Fig 4.1.

4.3.2 Experiments on Digits and Signed Datasets

In this section, we evaluate our model using the following five datasets: MNIST LeCun & Cortes (2010), Street View House Numbers (SVHN) Netzer et al. (2011), USPS Hull (1994), Synthetic Traffic Signs (SYN SIGNS) Moiseev et al. (2013), and the German Traffic Signs Recognition

Method	SVHN to MNIST	MNIST to USPS	USPS to MNIST	SYNSIG to GTSRB
Source only	67.1	79.4	63.4	85.1
DANN Ganin & Lempitsky (2015)	71.1	85.1	73.0±0.2	88.7
ADDA Tzeng et al. (2017)	76.0±1.8	-	90.1±0.8	-
CoGAN Liu & Tuzel (2016)	-	-	89.1±0.8	-
PixelDA Bousmalis et al. (2017)	-	95.9	-	-
ASSC Haeusser et al. (2017)	95.7±1.5	-	-	82.8±1.3
UNIT Liu et al. (2017)	90.5	96.0	93.6	-
CyCADA Hoffman et al. (2018)	90.4±0.4	95.6±0.2	96.5±0.1	-
GTA Sankaranarayanan et al. (2018)	92.4±0.9	95.3±0.7	90.8±1.3	-
DeepJDOT Bhushan Damodaran et al. (2018)	96.7	95.7	96.4	-
SimNet Pinheiro (2018)	-	96.4	95.6	-
GICT Qin et al. (2019)	98.7	96.2	96.6	-
MCD Saito et al. (2018)	96.2±0.4	96.5±0.3	94.1±0.3	94.4±0.3
ours (n = 3)	98.2±0.1	**98.5±0.2**	97.0±0.1	95.0±0.2
ours (n = 4)	98.6±0.1	98.4±0.3	**97.1±0.1**	95.1±0.1
ours (n = 5)	98.8±0.2	98.1±0.1	96.1±0.3	**95.5±0.2**
ours (n = 6)	**98.9±0.1**	98.0±0.1	96.6±0.3	95.3±0.3

Table 4.1: The performance on digit classification and sign classification.
The variable *n* refers to the number of classifiers. We report the mean and the standard deviation of the accuracy obtained over 5 trials.

Benchmark (GTSRB) Stallkamp et al. (2011).

MNIST: The dataset contains images of digits 0 to 9 in different styles. It is composed of $60,000$ training and $10,000$ testing images.

USPS: This is also a digit dataset with $7,291$ training and $2,007$ testing images.

SVHN: Another digit dataset with $73,257$ training, $26,032$ testing, and $53,1131$ extra training images.

SYN SIGNS: This is a synthetic traffic sign dataset, which contains $100,000$ labeled images, and 43 classes.

GTSRB: A dataset for German traffic signs recognition benchmark. The training set contains $39,209$ labeled images and the test set contains $12,630$ images. It also contains 43 classes.

We evaluate the unsupervised domain adaptation model on the following four transfer scenarios:

- SVHN $\rightarrow$ MNIST
- USPS $\rightarrow$ MNIST
- MNIST $\rightarrow$USPS
- SYNSIG $\rightarrow$ GTSRB

Results: The experimental results are shown in Table 4.1. We can clearly see that our proposed method has a significant improvement over the original MCD model. In the SVHN $\rightarrow$ MNIST task, our method has improved the performance by a margin of 2.7%. In the MNIST $\rightarrow$USPS task, our method improved by 2.0%. In the task of USPS $\rightarrow$ MNIST, our method improved by 3.0%. In the SYNSIG $\rightarrow$ GTSRB task, our method improved by 1.1%.

It can also be seen from Table 4.1 that blindly adding classifiers does not increase the model performance indefinitely. As the number of classifiers increases, the growth rate of the model's performance will decrease and the model's performance will approach a peak range. When the model reaches this peak range, adding more classifiers will cause a decrease in its performance.

The most significant increase happens when we increase the number of classifiers from 2 to 3. For example, for USPS-MNIST, the accuracy is increased by 2.0% when 3 classifiers are employed, however, the performance is only improved by 0.1% when the number of classifiers is increased from 3 to 4.

Method	plane	bcycl	bus	car	horse	knife	mcycl	person	plant	sktbrd	train	truck	mean
Source Only	55.1	53.3	61.9	59.1	80.6	17.9	79.7	31.2	81.0	26.5	73.5	8.5	52.4
MMD	72.3	53.1	64.7	31.8	58.2	14.3	80.7	60.0	70.0	41.4	89.7	20.7	55.9
DANN	81.9	77.7	82.8	44.3	81.2	29.5	65.1	28.6	51.9	54.6	82.8	7.8	57.4
MCD	87.0	60.9	83.7	64.0	88.9	79.6	84.7	76.9	88.6	40.3	83.0	25.8	71.9
OURS (n = 3)	91.1	70.0	84.1	68.8	93.5	85.3	88.2	76.7	89.2	75.0	86.2	31.2	78.3
OURS (n = 4)	93.3	77.6	85.3	77.4	91.8	90.2	86.4	79.3	89.5	80.4	88.5	33.1	81.1
OURS (n = 5)	95.2	81.4	82.9	82.7	91.9	89.6	86.7	79.3	91.0	66.6	88.1	35.7	80.9
OURS (n = 6)	95.0	80.1	85.5	83.1	90.5	89.9	85.9	79.2	92.8	85.3	88.3	33.9	82.5

Table 4.2: The performance on VisDA-2017.
Results of unsupervised domain adaptation on VisDA2017 Peng et al. (2017) image classification task. The accuracy is obtained by fine-tuning ResNet-101 He et al. (2016) model pre-trained on ImageNet Deng et al. (2009). This task evaluates the adaptation capability from synthetic CAD model images to real-world MS COCO Lin et al. (2014) images. Our model achieves the best performance in most categories.

4.3.3 Experiments on VisDA Classification Dataset

We further evaluate our method on the large VisDA-2017 dataset Peng et al. (2017). The VisDA-2017 image classification is a 12-class domain adaptation dataset used to evaluate the adaptation from synthetic-object to real-object images. The source domain consists of 152,397 synthetic images, where 3D CAD models are rendered from various conditions. The target domain consists of 55,388 real images taken from the MS-COCO dataset Lin et al. (2014).

In this experiment, we employ Resnet-101 He et al. (2016) as our feature extractor, and the parameters are adopted from the ImageNet pre-trained model. The pre-trained model of our Resnet-18 comes from Pytorch Paszke et al. (2017) and all experimental implementations are based on Pytorch. The input images are of the size 224×224. First, we resize the input image to 256, and then crop the image to 224×224 in the center. When we train the model using only the source domain, we just modify the output size of the original last fully connected layer to a size that conforms to VisDA-2017 Peng et al. (2017). In other tasks, we utilize a three-layer fully connected

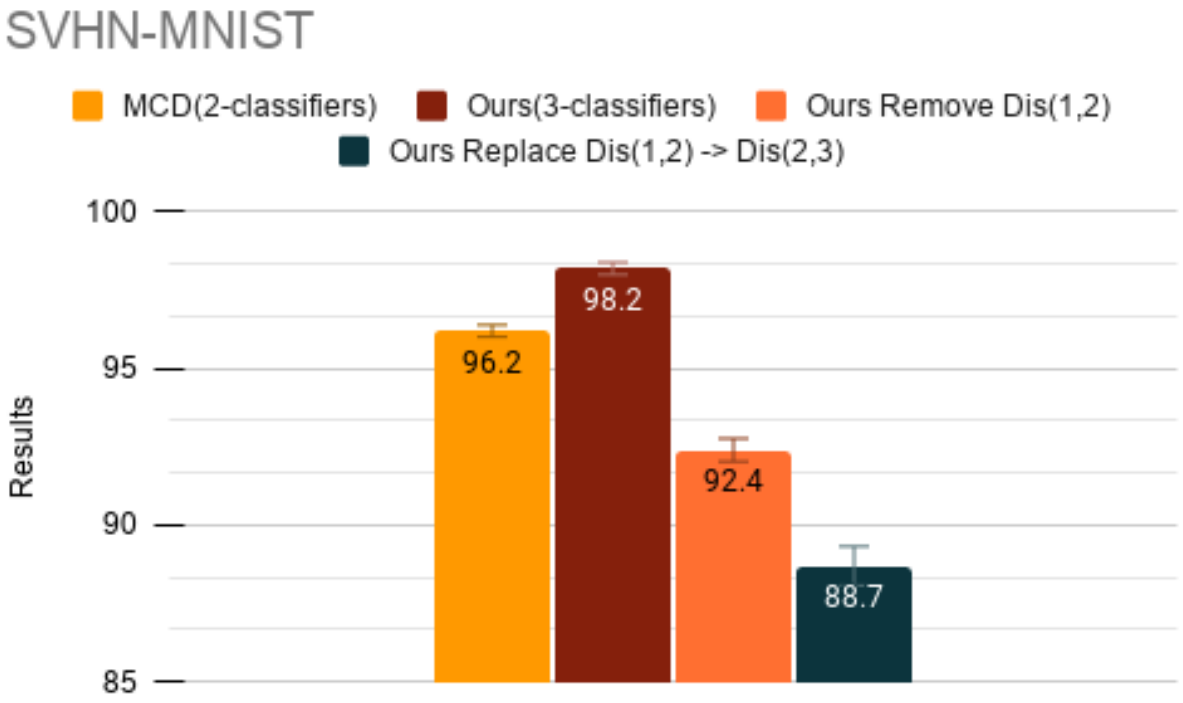


Figure 4.2: Result of the ablation study

layer structure to replace the one-layer fully connected layer structure of the original classifier. In order to eliminate the interference factors, except for the source only, all other algorithms use the same classifier. We uniformly use SGD as the optimizer for training, and use 1×10^{-3} for the learning rate of all methods. We use 16 as the batch size for training.

Results: The experimental results of this part are shown in Table 4.2. In this experiment, we achieve similar experimental results with the digits dataset. Compared with the original MCD Lu et al. (2020) method, our proposed multi-classifier implementation method significantly improves the performance in all experiments in comparison with the 2-classifier structure. The most prominent increase happens when the number of classifiers is increased from 2 to 3 with the mean accuracy promoted from 71.9% to 78.3%.

4.3.4 Ablation Study

We conducted the ablation study based on the digital datasets (SVHN $\rightarrow$ MNIST) with the same unsupervised domain adaptation setting as Section 4.3.2. Since our proposed method can add any number of classifiers, in order to simplify the work, we only use the structure with 3 classifiers as an example in this experiment. We compare our proposed method under three different situations: (i) the original MCD; (ii) arbitrarily remove a pair of discrepancy loss from our Loss; and (iii) arbitrarily replace a discrepancy loss with any existing discrepancy loss. The latter two cases simulate the situation where there is a gap in the loss closed loop we proposed. The first one is caused by the missing, and the second is caused by duplication. In this regard, we hope to prove the necessity of the closed structure of the loss function.

The results of the experiment are shown in Figure 4.2. It can be seen from the figure that whether it is missing or duplicated, both cases will cause a significant drop in the performance of our proposed method, even lower than the baseline MCD. In the case of repetition, the repeated sub-loss conflicts between the training of the two classifiers, which makes the function difficult to fit, and therefore all classifiers are affected.

4.3.5 Convergence and Efficiency Analysis

In this section, we make further analysis on the convergence and efficiency of the models. First, we explore the effect of the number of classifiers on the convergence speed of the model using the USPS-MNIST dataset as an example. We record the changes in the model's average discrepancy loss with iterations and presented them in Fig 4.3. From this experiment, we find that the convergence speeds of different models are roughly close to each other. As the number of classifiers increases, the convergence speed of the model will slightly increase.

Second, we compare the average time required to train an epoch with different numbers of classifier structures. We record the training time complexity in the USPS-MNIST experiment and show the result in Fig 4.4. From this figure, we can see that, as the number of classifiers increases, the time complexity will increase accordingly. However, the time increase is not too prominent

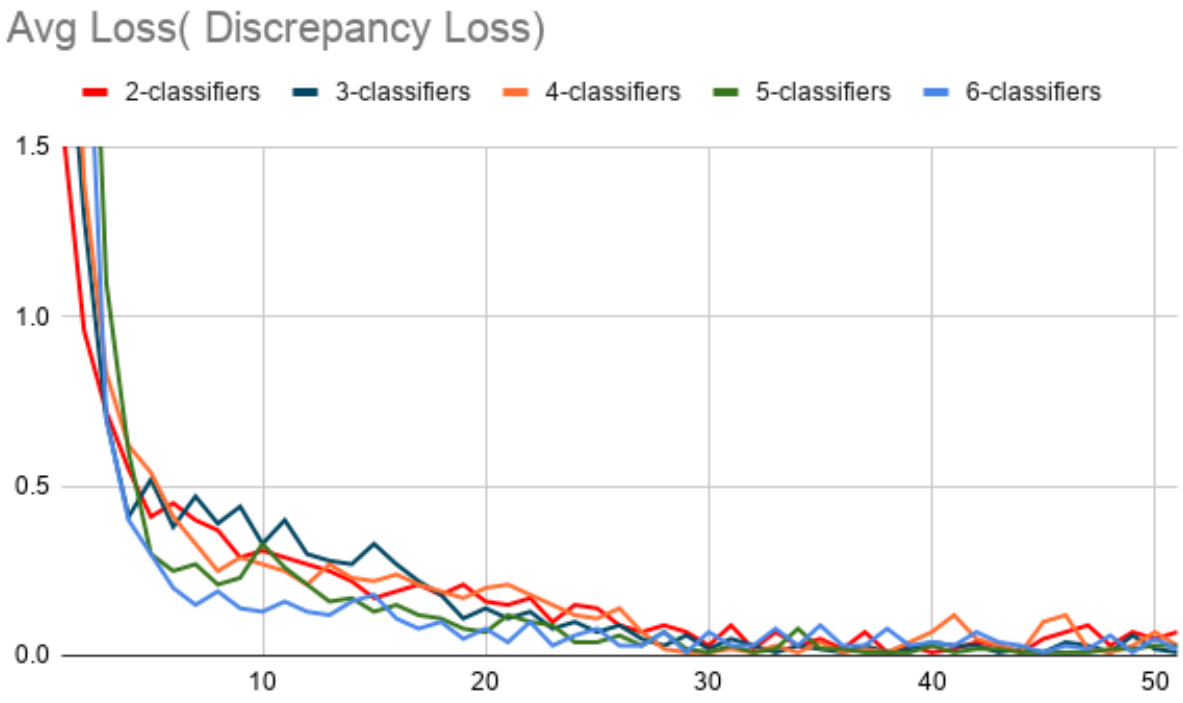


Figure 4.3: Convergence
The convergence of different numbers of classifier structures for the USPS-MNIST task. The convergence speed of discrepancy loss will be a little bit faster for more classifiers.

when the number of classifiers is increased from 2 to 3.

Based on the above experiments, we can see that the classification accuracy is boosted dramatically for all datasets when we increase the number of classifiers from 2 to 3, while the computation time only increases slightly. Therefore, as a trade-off between performance and efficiency, we believe the structure with 3 classifiers is the best choice in practice for most unsupervised domain adaptation tasks.

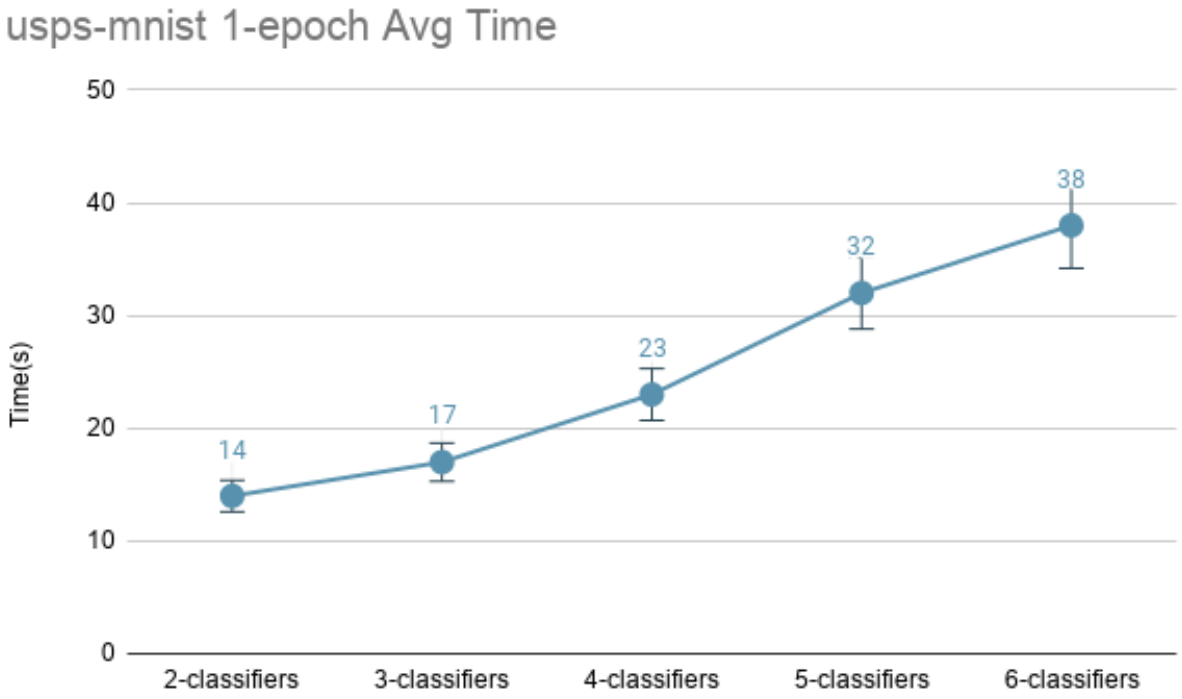


Figure 4.4: Time Complexity
The time complexity of our proposed method for the task of USPS-MNIST. The experiment is conducted using a RTX 2070 single GPU, and set the batch size to 256, and the other settings are the same as Section 4.3.2. From the data, we can see that the time complexity increases with the increase of the number of classifiers, however, the time it takes under the structure of three-classifier is close to that of two-classifier.

Chapter 5

Conclusion

In this thesis, we proposed two new unsupervised domain adaptation method for image classification tasks.

1. we have proposed a dual-module network architecture that can strongly encourage domain invariant feature learning. The network architecture is composed of a discriminative feature learning module and a domain invariant feature learning module. We have proposed an adversarial loss function using the difference between the feature distributions of the two modules and the similarity of their predicted results. The two modules will compete with each other to maximize the difference in feature distribution. The proposed model employs the maximum classifier discrepancy to solve the imbalance problem of domain discriminative feature extraction in the target domain in the two modules. Extensive experiments demonstrate that the proposed method achieves state-of-the-art performance on the standard unsupervised domain adaptation benchmarks and significantly improves its performance.

2. we proposed a straightforward method of adding classifiers for the adversarial training framework of the maximum classifier discrepancy of the multi-classifier structure. In order to prevent conflicts during training of multiple classifiers, we propose a discrepancy loss function based on the principle that the classifiers are different from each other. This loss function allows us to add any number of classifiers under the original Maximum classifier discrepancy framework. Through experiments, we found that the multi-classifier structure has obvious performance advantages over the original 2-classifier structure. In all classification tasks, even if only the structure of three classifiers is used, its performance will be much higher than the original method.

References

Bhushan Damodaran, B., Kellenberger, B., Flamary, R., Tuia, D., & Courty, N. (2018). Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 447–463).

Bousmalis, K., Silberman, N., Dohan, D., Erhan, D., & Krishnan, D. (2017). Unsupervised pixel-level domain adaptation with generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3722–3731).

Bousmalis, K., Trigeorgis, G., Silberman, N., Krishnan, D., & Erhan, D. (2016). Domain separation networks. In *Advances in neural information processing systems* (pp. 343–351).

Cai, R., Li, Z., Wei, P., Qiao, J., Zhang, K., & Hao, Z. (2019). Learning disentangled semantic representation for domain adaptation. In *IJCAI: proceedings of the conference*, volume 2019 (pp. 2060).: NIH Public Access.

Carlucci, F. M., D'Innocente, A., Bucci, S., Caputo, B., & Tommasi, T. (2019). Domain generalization by solving jigsaw puzzles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Chen, C., Fu, Z., Chen, Z., Jin, S., Cheng, Z., Jin, X., & Hua, X.-S. (2020a). Homm: Higher-order moment matching for unsupervised domain adaptation. *order*, 1(10), 20.

Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020b). A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Ganin, Y. & Lempitsky, V. (2015). Unsupervised domain adaptation by backpropagation. In *International conference on machine learning* (pp. 1180–1189).

Ghifary, M., Kleijn, W. B., Zhang, M., Balduzzi, D., & Li, W. (2016). Deep reconstruction-classification networks for unsupervised domain adaptation. In *European Conference on Computer Vision* (pp. 597–613).: Springer.

Gong, R., Li, W., Chen, Y., & Gool, L. V. (2019). Dlow: Domain flow for adaptation and generalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2477–2486).

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 27* (pp. 2672–2680). Curran Associates, Inc.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., & Smola, A. (2012). A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1), 723–773.

Haeusser, P., Frerix, T., Mordvintsev, A., & Cremers, D. (2017). Associative domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 2765–2773).

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

Hoffman, J., Tzeng, E., Park, T., Zhu, J.-Y., Isola, P., Saenko, K., Efros, A., & Darrell, T. (2018). Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning* (pp. 1989–1998).

Hu, L., Kan, M., Shan, S., & Chen, X. (2018). Duplex generative adversarial network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Hull, J. J. (1994). A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5), 550–554.

Kang, G., Jiang, L., Yang, Y., & Hauptmann, A. G. (2019). Contrastive adaptation network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4893–4902).

LeCun, Y. & Cortes, C. (2010). MNIST handwritten digit database.

Lee, S., Kim, D., Kim, N., & Jeong, S.-G. (2019). Drop to adapt: Learning discriminative features for unsupervised domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 91–100).

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision* (pp. 740–755).: Springer.

Liu, M.-Y., Breuel, T., & Kautz, J. (2017). Unsupervised image-to-image translation networks. In *Advances in neural information processing systems* (pp. 700–708).

Liu, M.-Y. & Tuzel, O. (2016). Coupled generative adversarial networks. In *Advances in neural information processing systems* (pp. 469–477).

Long, M., Cao, Y., Wang, J., & Jordan, M. (2015). Learning transferable features with deep adaptation networks. In *International conference on machine learning* (pp. 97–105).

Long, M., Zhu, H., Wang, J., & Jordan, M. I. (2016). Unsupervised domain adaptation with residual transfer networks. In *Advances in neural information processing systems* (pp. 136–144).

Long, M., Zhu, H., Wang, J., & Jordan, M. I. (2017). Deep transfer learning with joint adaptation networks. In *International conference on machine learning* (pp. 2208–2217).: PMLR.

Lu, Z., Yang, Y., Zhu, X., Liu, C., Song, Y.-Z., & Xiang, T. (2020). Stochastic classifiers for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 9111–9120).

Ma, X., Zhang, T., & Xu, C. (2019). Gcan: Graph convolutional adversarial network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 8266–8276).

Moiseev, B., Konev, A., Chigorin, A., & Konushin, A. (2013). Evaluation of traffic sign recognition methods trained on synthetically generated data. In J. Blanc-Talon, A. Kasinski, W. Philips, D. Popescu, & P. Scheunders (Eds.), *Advanced Concepts for Intelligent Vision Systems* (pp. 576–583). Cham: Springer International Publishing.

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

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). Automatic differentiation in pytorch.

Peng, X., Usman, B., Kaushik, N., Hoffman, J., Wang, D., & Saenko, K. (2017). Visda: The visual domain adaptation challenge.

Pinheiro, P. O. (2018). Unsupervised domain adaptation with similarity learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 8004–8013).

Qin, C., Wang, L., Zhang, Y., & Fu, Y. (2019). Generatively inferential co-training for unsupervised domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision Workshops* (pp. 0–0).

Saito, K., Watanabe, K., Ushiku, Y., & Harada, T. (2018). Maximum classifier discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3723–3732).

Sankaranarayanan, S., Balaji, Y., Castillo, C. D., & Chellappa, R. (2018). Generate to adapt: Aligning domains using generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 8503–8512).

Stallkamp, J., Schlipsing, M., Salmen, J., & Igel, C. (2011). The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *IEEE International Joint Conference on Neural Networks* (pp. 1453–1460).

Sun, B. & Saenko, K. (2016). Deep coral: Correlation alignment for deep domain adaptation. In *European conference on computer vision* (pp. 443–450).: Springer.

Tran, L., Yin, X., & Liu, X. (2017). Disentangled representation learning gan for pose-invariant face recognition. In *In Proceeding of IEEE Computer Vision and Pattern Recognition* Honolulu, HI.

Tzeng, E., Hoffman, J., Saenko, K., & Darrell, T. (2017). Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7167–7176).

Tzeng, E., Hoffman, J., Zhang, N., Saenko, K., & Darrell, T. (2014). Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*.

van der Maaten, L. & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9, 2579–2605.

Wang, D., Cui, P., & Zhu, W. (2018). Deep asymmetric transfer network for unbalanced domain adaptation. In *AAAI* (pp. 443–450).

Zellinger, W., Grubinger, T., Lughofer, E., Natschläger, T., & Saminger-Platz, S. (2017). Central moment discrepancy (cmd) for domain-invariant representation learning. *arXiv preprint arXiv:1702.08811*.