Generalization to unseen visual domains via data augmentation and representation learning

March 2023

YU ZHE

Generalization to unseen visual domains via data augmentation and representation learning

Graduate School of Systems and Information Engineering

University of Tsukuba

March 2023

YU ZHE

Generalization to unseen visual domains via data augmentation and representation learning
データ増強と表現学習による未知の視覚ドメインへの汎化

Student No.: 201930179
Name: Yu Zhe

Machine learning models are often trained under the assumption that the training distribution (source domain) and test distribution (target domain) are independent and identically distributed. However, in reality, machine learning models are often deployed in applications where test data and training data are drawn from different distributions. The predictive performance of models trained in such a situation may decrease significantly.

One of the most common strategies to improve machine learning models' performance on out-of-distribution data is unsupervised domain adaptation (UDA). UDA assumes that labeled data from the source domain and unlabeled data from the target domain are obtainable. With these data, UDA methods are able to train a model that can perform well on the target domain.

However, in reality, it is often difficult to obtain unlabeled data from the target domain. We consider two situations in this thesis. First, machine learning models need to handle unseen environments in many applications. In this situation, the target domain is unpredictable. Hence, it is impossible to collect data from the target domain. Second, certain types of data are hard to collect. For example, when the task we want to solve is related to private data, since eliminating privacy is expensive, collecting private data is difficult. In this situation, it is hard to collect task-relevant data from the target domain.

In this thesis, we consider two settings, domain generalization (DG) and zero-shot domain adaptation (ZSDA), to improve the machine learning model's domain generalization ability in the above two situations. DG aims to utilize data from multiple source domains to build machine learning models that can handle unseen target domains. ZSDA sim to utilize task-irrelevant data from the target domain to build machine learning models that can solve task-of-interest in the target domain.

To solve DG and ZSDA tasks, we consider two strategies, data augmentation and representation learning. First, data augmentation aims to provide new data to the training data. The domain generalization ability of the machine learning model can be improved when the augmented data can simulate the domain changes. Second, representation learning aims to design the features extracted from the training data carefully. When extracted features are domain-invariant, the resulting models are naturally able to handle different domains.

In this thesis, we discuss how to utilize data augmentation and representation learning to solve DG and ZSDA tasks.

Academic Advisors: Principal: Jun Sakuma
Secondary: Youhei Akimoto

Contents

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Approach . . . 3
1.3 Contribution . . . 4
1.4 Outline . . . 5
1.5 List of publications . . . 6

2 Literature Review **7**
2.1 Preliminary . . . 7
2.2 Unsupervised Domain Adaptation . . . 8
2.2.1 Problem setting . . . 8
2.2.2 Adversarial Learning-based Methods . . . 8
2.2.3 Generative model-based methods . . . 10
2.3 Partial domain adaptation . . . 10
2.3.1 Motivation and problem setting . . . 10
2.3.2 Partial domain adaptation methods . . . 11
2.4 Source-free domain adaptation . . . 11
2.4.1 Motivation and problem setting . . . 11
2.4.2 Source-free domain adaptation methods . . . 12
2.5 Domain generalization . . . 13
2.5.1 Motivation and problem setting . . . 13
2.5.2 Domain generalization methods . . . 13
2.6 Zero-shot domain adaptation . . . 15
2.6.1 Motivation and problem setting . . . 15
2.6.2 Zero-shot domain adaptation methods . . . 15
2.7 Summary and discussion . . . 16

3 Domain Generalization via Adversarially Learned Novel Domains **18**
3.1 Introduction . . . 18
3.2 Preliminary . . . 20
3.2.1 Cycle-Consistent Adversarial Networks . . . 20
3.2.2 Domain flow generator . . . 21
3.3 Proposal . . . 22
3.3.1 Domain flow generator with better cycle . . . 23
3.3.2 Representative data generating distribution by domain-code distribution . . . 24

3.3.3 Generative adversarial domain augmentation 25
3.4 Experiments 26
3.4.1 Experimental setting 26
3.4.2 Performance on Benchmark dataset 29
3.4.3 Visualization results 30
3.4.4 Ablation studies 32
3.4.5 Experiment with sufficient hyperparameters selection 33
3.5 Further discussion 34
3.5.1 How the number of source domains affects our proposal 35
3.5.2 Why our proposal improves the model's generalization ability 35
3.6 Conclusion 36
3.7 Limitation 36

4 Zero-shot domain adaptation based on dual-level mix and contrast 38
4.1 Introduction 38
4.2 Preliminary 40
4.2.1 Domain adversarial training 40
4.2.2 Mixup 41
4.3 Proposal 42
4.3.1 Overview 42
4.3.2 Dual Mixup for intermediate samples 43
4.3.3 Domain adversarial training with dual mixup 43
4.3.4 Dual contrastive learning for disentanglement 44
4.3.5 Adaptive temperature for contrastive learning 46
4.4 Experiments 46
4.4.1 Experimental results on two benchmarks 46
4.4.2 Visualizations 48
4.4.3 Ablation studies 50
4.5 Conclusion 51
4.6 Limitation 51

5 Conclusion 52
5.1 Future work 52

Acknowledgement 54

References 55

List of Figures

1.1 An illustration of a classification model's performance under domain shift . . 2

1.2 An illustration of domain generalization (DG) setting and zero-shot domain adaptation (ZSDA) setting. The blue box denotes the data that can be obtained during training. The red box denotes the data that can not be obtained during training. The goal of these two settings is to train a model that can predict data in the red box 4

2.1 A illustration of domain adversarial training, [1] 9

3.1 An illustration of the DLOW-B model and three domain codes. This model is trained on three different domains, so the domain code is a 3-dimensional probability vector. We visualize this model on a 2-dimensional simplex. Here, each point in the 2-dimensional simplex corresponds to a 3-dimensional domain code. By varying domain codes, the generated images exhibit different styles. 19

3.2 An illustration of the CycleGAN model and three domain codes. 21

3.3 An overview of GADA with the domain-code NN. The domain-code NN takes Gaussian noise as input and outputs a valid domain code. The pretrained DLOW takes both image x and domain code as input and synthesizes an image with the style specified by the domain code. The classification model is trained with synthesized images. We optimize the domain code-NN to maximize the classification loss so that it can represent the worst-case distribution. 23

3.4 Comparison of DLOW and DLOW-B 24

3.5 Example images from different DG datasets. 27

3.6 Visualization result of GADA-NN. The top represents how domain codes change during training, and the bottom represents how generated images change during training 31

3.7 Visualization result of GADA-D. The top represents how domain codes change during training, and the bottom represents how generated images change during training. 32

4.1 An example of zero-shot domain adaptation, samples in blue box are available during training, samples in red box are what we want to classify 39

4.2 An illustration of mixup 41

4.3 An illustration of our proposal DMCL. Here GRL refers to a gradient reversal layer. The red arrow indicates the adversarial learning procedure, the green arrow indicates a standard training procedure, and the pink arrow indicates the contrastive learning objectives. 42
4.4 An illustration of dual contrastive learning, here the dashed line indicates apply mixup on two samples. 45
4.5 Visualization results when domain shift is domain G to domain N. At the top, the ToI is MNIST, IrT is Fashion-MNIST. At the bottom, the ToI is MNIST, IrT is EMNIST. 49

List of Tables

2.1 A comparison among different settings 7

3.1 Experimental result on digits-DG with classification accuracy (%) averaged over 10 runs 30
3.2 Experimental result on PACS with classification accuracy (%) averaged over 10 runs 30
3.3 Experimental result on office-home with classification accuracy (%) averaged over 10 runs 31
3.4 Ablation studies on digits with classification accuracy (%) averaged over 10 runs 33
3.5 Ablation studies on PACS with classification accuracy (%) averaged over 10 runs 33
3.6 Experimental result of GADA with fine-tuned hyperparameters on digits-DG and PACS with classification accuracy (%) averaged over 10 runs 34
3.7 Experimental result of GADA with multiple DLOW-B models 35

4.1 Experimental result on X-NIST with classification accuracy (%) averaged over 10 runs 47
4.2 Experimental result on Office-Home with classification accuracy (%) averaged over 10 runs 48
4.3 Ablation studies on X-NIST with classification accuracy (%) averaged over 10 runs 50

Chapter 1

Introduction

In recent years, machine learning systems have achieved great success in different tasks. Image classification is one of the tasks that have been widely progressed. Machine learning models have gone from only being able to achieve good performance on simple digital classification tasks [2] to now being able to handle large-scale complex classification tasks accurately [3]. These successes largely depend on assuming that training and test data are independent and identically distributed. This assumption is reflected in the specific process of training and using machine learning models. A common process is first to collect data from an environment, later use that data to train a machine learning model, and eventually apply that model to the same environment.

However, in reality, machine learning models are often deployed in applications where test data and training data are drawn from different distributions. For example, a medical diagnostic model trained using data from hospital A may need to diagnose medical data from hospital B. In such a situation, the predictive performance of machine learning models trained may decrease significantly [4]. The performance drop will happen even in simple digital classification tasks [5]. As Figure 1.1 shows, when training data and test data are both grey-scale or colorful images, a classification model can achieve very good results on test data. However, when training data and test data are from different domains, the model's classification accuracy is only 52.25 %. This issue is commonly referred to as **domain shift** [6]. More specifically, a domain is defined as a joint distribution between samples and their label. The domain of training data is defined as the source domain, and the domain of target data is defined as the target domain. When the source domain and target domain are different but related, such as $P(y|x)$ can be commonly shared, we call domain shift happens.

1.1 Motivation

A large amount of research attempts to improve the model's performance when domain shifts occur [1, 7, 8, 9, 10]. A popular strategy is to collect unlabeled data from the test environment and use these data to improve the generalization ability of the model in the test environment. For example, unsupervised domain adaptation (UDA) [5] is one of the most common settings to alleviate domain shift with this strategy. UDA assumes that labeled data from the source domain and unlabeled data from the target domain are available. UDA aims to train a model that can generalize the target domain. A certain number of UDA

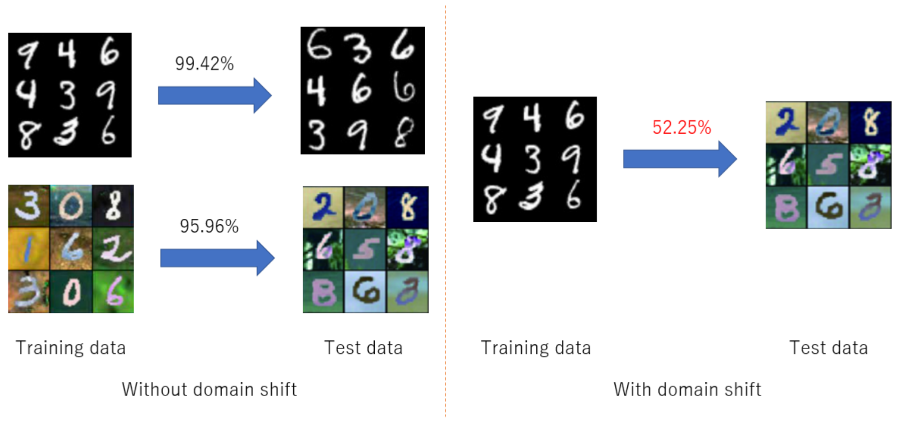


Figure 1.1: An illustration of a classification model's performance under domain shift

methods show outstanding performance on alleviating domain shifts [1, 7, 11].

UDA methods rely on the availability of unlabeled data from the target domain. However, it is often not easy to obtain data from the target domain in reality. Generally, we consider two situations where it is difficult to obtain data from the target domain:

1. **When the target domain is unpredictable, it is difficult to collect data from the target domain.** For example, self-driving car models need to handle various weather conditions. Due to the variation of the environment, it is difficult to specify the road conditions that will be faced in the future.

2. **Although it is possible to specify the target domain, it is hard to collect data we want to predict in that target domain.** For example, if we want to utilize unsupervised domain adaptation algorithms to adapt a diagnostic model trained on hospital A to hospital B, we need to collect unlabeled medical data from hospital B. However, since medical data are usually private, it is expensive or time-consuming to eliminate private information from these data.

In the above two situations, the usability of UDA methods will be greatly reduced. To tackle domain shifts under these situations, domain generalization (DG) [12, 8] and zero-shot domain adaptation (ZSDA) [9, 13, 14] settings are designed, respectively. The main difference between DG, ZSDA, and UDA is whether the task-related data can be obtained from the target domain.

When handling unseen target domains, there is no way to specify all possible target domains that we may encounter. However, it is possible to specify several possible target domains to encounter. For example, we can not specify all possible weather conditions when we train a self-driving car system. But, we can specify several possible weather conditions, such as rainy and cloudy. Then, we can collect data from these possible weather, and we then train a model with these data and expect this model to generalize to the unseen weather condition. DG is designed to realize the above procedure. DG assumes data from multiple domains are available, and the goal of DG is to train a model that can generalize to unseen domains. Since the test data is from an unknown domain, DG seeks to improve the generalization ability of the model in a general sense and expect such generalization ability can work on an unknown domain. An illustration of DG is shown in Figure 1.2.

On the other hand, in the second situation, we consider that the target domain can be specified, but task-relevant data is unobtainable in the target domain. Hence, what we need is a model that works well in a particular domain. Models that perform well for a particular domain are clearly easier to obtain than models that perform well in a general sense. Hence, DG would be an expensive setting to solve the second situation, as DG would seek to solve a much harder problem. We propose a zero-shot domain adaptation (ZSDA) method to solve the second situation. ZSDA involves two tasks. We call the classification task we want to solve the task-of-interest (ToI), we call the classification task that is irrelevant to the ToI irrelevant task (IrT). ZSDA assumes that we can obtain ToI data from the source domain and IrT data from both the source and target domains, and domain shift is shared with the ToI data and IrT data. For example, as Figure 1.2 shows, we want to train a model for classifying dogs, and we need to tackle the domain shift between the photo domain and the sketch domain. Here, the dog classification task is ToI. However, for some reason, we can not get dog images from the sketch domain. ZSDA assumes that we can obtain guitar images from both the photo and sketch domains. Here, the guitar classification task is IrT. In this case, the difference in the domain is the difference between photo style and sketch style, no matter it is a dog classification task or a guitar classification task. Hence, it is possible to utilize guitar images from both the photo and sketch domains to learn the style difference between the photo and sketch styles. Then, we can utilize learned style differences and dog images from the photo domain to train a model that can generalize to the sketch domain.

This thesis focuses on DG and ZSDA settings for alleviating domain shifts when task-relevant data from the target domain is unavailable. There are four differences between DG and ZSDA settings.

1. **Number of domains:** DG involves more than three domains, while ZSDA only involves two domains.
2. **Number of tasks:** In the DG setting, only one task is involved. We only need to handle differences caused by domains. In contrast, in the ZSDA setting, we need to deal not only with the difference between source and target domains but also with the difference between relevant and irrelevant tasks.
3. **Characteristics of target domain:** In the DG setting, it is impossible to obtain any information from the target domain since the target domain is not specified during the model training. However, in the ZSDA setting, the target domain is specified, we can utilize task-irrelevant data from the target domain to predict the characteristics of target domain.
4. **Goal:** DG aims to improve the model's domain generalization ability in a general sense, ZSDA aim to improve the model's generalization ability in a specific domain.

1.2 Approach

In this thesis, we mainly focus on two types of strategies to solve DG and ZSDA tasks:

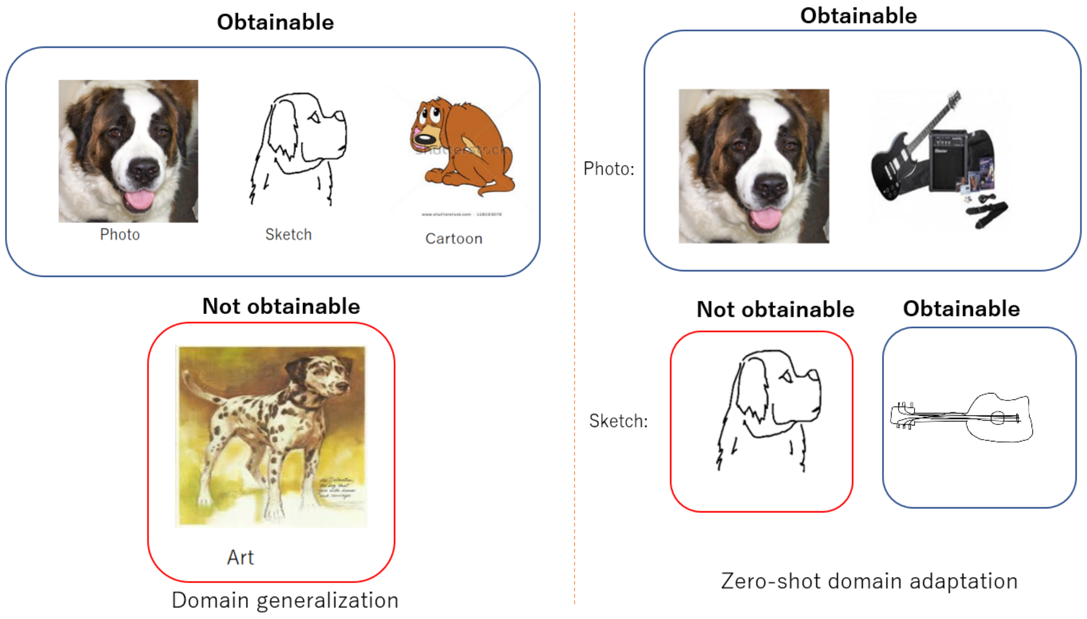


Figure 1.2: An illustration of domain generalization (DG) setting and zero-shot domain adaptation (ZSDA) setting. The blue box denotes the data that can be obtained during training. The red box denotes the data that can not be obtained during training. The goal of these two settings is to train a model that can predict data in the red box

1. **Data augmentation.** Data augmentation aims to increase the amount of training data by generating new data from existing training data. Data augmentation [9, 7] has been found to be a very effective strategy for alleviating domain shifts. With diverse training data, the machine learning model is forced to learn more invariant features. The invariant feature will lead to better domain generalization ability [15].

2. **Representation learning.** In general, we can divide a neural network into two parts, the last layer of the neural network is a linear classifier, and the rest of the neural network learns to provide representations to this classifier. Representation learning aims to design the representation, also called features, drawn from the data carefully for different purposes. In the context of alleviating domain shift, learning domain-invariant features is a common strategy. Domain-invariant features are not affected by changing of domains. if a classifier is based on such features to make predictions, such a classifier should be also not affected by changing of domains.

In Chapter 3, we propose a novel data augmentation technique for solving the DG task. In Chapter 4, we propose a framework that contains a novel representation learning objective for solving the ZSDA task.

1.3 Contribution

In this section, we list two contributions made in this thesis, which provide novel and effective solutions to DG and ZSDA.

1. **Generative Adversarial Domain Augmentation (GADA) (Chapter 3)**: The first contribution is a data augmentation-based method for DG task. GADA aims to

augment training data with diverse and difficult samples. We see three advantages of our proposal. First, we propose a flexible framework that utilizes the worst-case data-generating distribution represented by a generative model. More specifically, we propose utilizing domain codes to control the behavior of the generative model. The data-generating distribution is consequently adapted by optimizing the domain-code distribution so that it helps to improve the domain generalization ability. This framework can be easily extended to other tasks that require control of the data-generating distribution of a generative model. Second, our proposal serves as a data augmentation-based method. Due to the flexibility of the data augmentation methodology, our framework can be easily combined with other domain generalization algorithms. We remark that, to the best of our knowledge, our proposal is the first data augmentation-based method that exploits the worst-case data-generating distribution characterized by generative models that can represent semantically different domains. Third, experiments on three benchmarks indicate our proposal shows superior domain generalization ability.

2. **Dual Mixup Contrastive Learning (DMCL) (Chapter 4)**: The second contribution is a representation method for the ZSDA task. DMCL aims to adapt a classical representation learning method, domain adversarial training (DAT) [1], to the ZSDA setting for learning domain-invariant features. To achieve this, we extend the mixup [16] technique and combine it with DAT so that DAT can work with domains with different label spaces. Also, we design a dual-level contrastive learning objective to force feature disentanglement so that the classifier will rely on domain-invariant features more. With these designs, DMCL has the following merits. First, in the whole framework, unlike most other related work, we do not rely on generative models. Hence, DMCL requires fewer computational resources. Second, the model structure in DCML is similar to that of the classical DAT, so some further enhancement methods for this structure can be combined with our approach [17]. Third, we evaluate our proposal on two benchmarks. The experimental results suggest that our proposal successfully adapts DAT in the ZSDA setting and achieves good domain adaptation ability.

1.4 Outline

The rest of the thesis is organized as follows.

1. **Chapter 2** introduces existing work that aims to alleviate domain shifts. More specifically, we review existing work related to five settings designed to overcome domain shits, which are unsupervised domain adaptation (UDA), partial domain adaptation (PDA), source-free domain adaptation (SFDA), domain generalization (DG), and zero-shot domain adaptation (ZSDA).

2. **Chapter 3** presents a novel data augmentation method for domain generalization tasks. This method aims to augment training data with diverse and difficult images.

3. **Chapter 4** presents a representation method for zero-shot domain adaptation methods. This method aims to learn domain-invariant features when the source and target domain contain different label spaces.

4. **Chapter 5** summarizes this thesis and draws overall conclusions. We also discuss future research directions in this chapter.

1.5 List of publications

This thesis is based on the following articles. The first and second articles are for the domain generalization task (Chapter 3), which is a task designed for the first situation, and the third is for the zero-shot domain adaptation task (Chapter 4).

1. Yu Zhe, Kazuto Fukuchi, Youhei Akimoto, Jun Sakuma, "Domain Generalization Via Adversarially Learned Novel Domains," in 2022 IEEE International Conference on Multimedia and Expo (ICME), Taipei, Taiwan, 2022 pp. 1-6.
2. Yu Zhe, Kazuto Fukuchi, Youhei Akimoto, Jun Sakuma, "Domain Generalization via Adversarially Learned Novel Domains," in IEEE Access, vol. 10, pp. 101855-101868, 2022, doi: 10.1109/ACCESS.2022.3209815.
3. Yu Zhe, Kazuto Fukuchi, Jun Sakuma, Zero-shot domain adaptation based on dual-level mix and contrast. in Information-Based Induction Sciences and Machine Learning (IBISML) 2022-53.

Chapter 2

Literature Review

In this chapter, we introduce some related works that investigate how to alleviate domain shifts in the image classification task. In Section 2.1, we give a preliminary definition of the problem we aim to solve. In Section 2.2, we review unsupervised domain adaptation methods. In Section 2.3, we review partial domain adaptation methods. In Section 2.4, we review source-free domain adaptation methods. These works all rely on data from the target domain to alleviate domain shifts. In Section 2.5 and Section2.6, we discuss domain generalization methods and zero-shot domain adaptation methods. Section 2.5 and Section2.6 are directly related to our focus, which attempts to overcome domain shifts without obtaining task-relevant data from the target domain. Section 2.7 gives a discussion about introduced methods and settings. A comparison of characteristics of the above settings is shown in Table 2.1.

Table 2.1: A comparison among different settings

Topic	Can we obtain task-relevant data from the source domain	Can we obtain task-relevant data from the target domain	Can we specify target domain	Do we need to handle differnet label space	number of domains	number of tasks
UDA	Yes	Yes	Yes	No	2	1
PDA	Yes	Yes	Yes	Yes	2	2
SFDA	No	Yes	Yes	No	2	1
DG	Yes	No	No	No	More than 3	1
ZSDA	Yes	No	Yes	Yes	2	2

2.1 Preliminary

In this thesis, we focus on the classification task when domain shifts happen. In this section, we first provide the necessary notations and definitions used in this section. This thesis considers how to overcome domain shift under the supervised learning framework; hence we first introduce supervised learning.

Supervised learning refers to utilizing **labeled** datasets to train machine learning models that can generalize to the test data. Formally, let $\mathcal{X}$ be the input space and $\mathcal{Y}$ be the label space. Given a training dataset D_{train} that contain N labeled training examples, $D_{train} = \{(x_i, y_i)\}_{i=1}^{N}$, supervised learning algorithms aim to learn a machine learning model $f : \mathcal{X} \rightarrow \mathcal{Y}$. Then, the trained model f is used to classify samples from a test dataset

$D_{test} = \{(x_i)\}_{i=1}^{M}$. When the training data and test data are independent and identically distributed, we call the IID assumption holds. When the IID assumption holds, a model that performs well in the training dataset is expected to perform well in the test dataset. However, IID assumption may not hold in some cases, and we focus on a special case domain shift.

Let $\mathcal{X}$ be the input space and $\mathcal{Y}$ be the label space. A domain is defined as a joint distribution P_{XY} on $\mathcal{X} \times \mathcal{Y}$. For the sake of simplicity, we often use P to refer to P_{XY}. We denote the domain of the training data as source domain P^s and denote the domain of the test data as target domain P^t. When $P^s \neq P^t$, the IID assumption does not hold, and we call this domain shift happens.

Also, in this thesis, we use the label set to distinguish between different classification tasks. Given task A and task B with label sets $\mathcal{C}^a$ and $\mathcal{C}^b$ if $\mathcal{C}^a = \mathcal{C}^b$, then task A and task B are the same task. Otherwise, these two tasks are different tasks. In Section 2.3 and Section 2.6, we will introduce settings that involve two tasks.

2.2 Unsupervised Domain Adaptation

Unsupervised domain adaptation (UDA) aims to utilize labeled data from the source domain and unlabeled data from the target domain to train a model that can generalize to the target domain [1]. This setting is the most common and basic setting to alleviate domain shifts. The solution to the other settings is basically inspired by the UDA approach. Hence we first review classical strategies for solving UDA in this section.

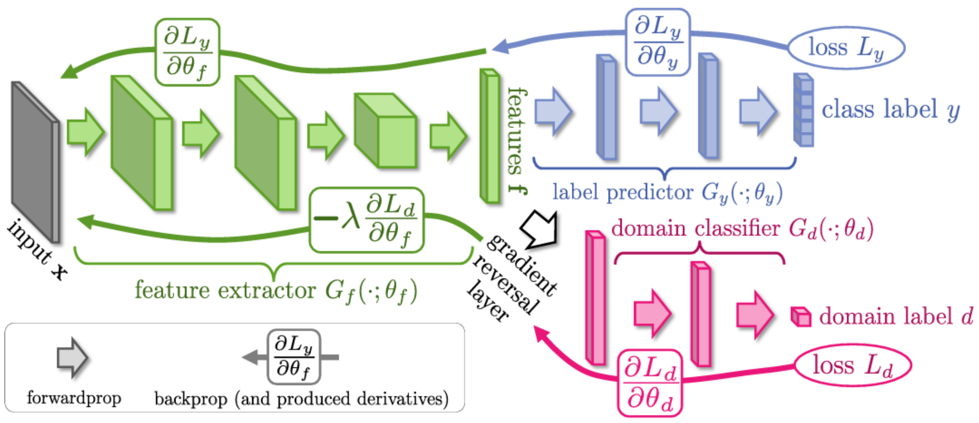
2.2.1 Problem setting

We first introduce a formal definition of UDA. UDA assumes that we have two domains, a source domain P^s_{XY} and a target domain P^t_{XY}. Also, UDA assumes that the source domain and target domain have the same label space. UDA assumes that we can have a labeled dataset that contains N data-label pairs drawn from domain P^s_{XY}, namely $D_s = \{(x^s_i, y^s_i)\}_{i=1}^{N}$ with $(x^s_i, y^s_i) \sim P^s_{XY}$. Also, we have a unlabeled dataset that contains M samples drawn from target domain, namely $D_t = \{(x^t_i)\}_{i=1}^{M}$. The goal of UDA is to train a model with a generalization ability to the target domain.

2.2.2 Adversarial Learning-based Methods

One of the most classical strategies to solve UDA is adversarial-based methods [1, 7, 18, 19]. These methods attempt to learn domain-invariant features to solve the UDA task. Domain-invariant features refer to features that are not affected by domain changes. If the classification model only relies on domain-invariant features, the model should not be affected by the domain shifts. In general, the adversarial-based method introduces a domain classifier that classifies whether a sample comes from the source or target domain. Then, by adversarially optimizing the feature extractor with the domain classifier, the features extracted from the source and target domains will become more and more domain invariant.

Adversarial-based strategy is first introduced to solve UDA tasks by [1]. They proposed a method termed domain adversarial training (DAT). As Figure 2.1 shows, DAT designs a model which contains a feature extractor (green in Figure 2.1), a label classifier (blue



$\frac{\partial L_y}{\partial \theta_f}$
$\frac{\partial L_y}{\partial \theta_y}$
loss L_y
class label y
features $\mathbf{f}$
label predictor $G_y(\cdot;\theta_y)$
$-\lambda\frac{\partial L_d}{\partial \theta_f}$
input $\mathbf{x}$
feature extractor $G_f(\cdot;\theta_f)$
gradient reversal layer
domain classifier $G_d(\cdot;\theta_d)$
domain label d
loss L_d
$\frac{\partial L_d}{\partial \theta_d}$
forwardprop
backprop (and produced derivatives)

Figure 2.1: A illustration of domain adversarial training, [1]

in Figure 2.1), and a domain classifier (red in Figure 2.1). During the backpropagation, a gradient reversal layer will reverse the gradient from the domain classifier, thus forcing the feature extractor to learn features that can not be used to classify the domain, i.e., domain-invariant features.

DAT was found to be a very effective method for solving UDA. Lots of studies follow the idea of DAT and try to improve this method. [20] argued that the performance of DAT relies heavily on the domain classifier. However, in the DAT method, the domain classifier is only trained by classifying the domain information of samples, which is a binary classification task. Hence such domain classifiers may not be able to explore the intrinsic structure of data distribution. To improve the performance of the domain classifier, [20] designed a data augmentation method, termed domain mixup, which augments training with intermediate samples between domains. With this data augmentation method, the domain classifier is required to not only classify the source and target domains but also to capture the intermediate status between the source and target domains. Hence, the resulting domain classifier is able to have a better discriminative ability to judge the domain information of samples. Adversarial learning with such domain classifier also show better domain adaptation ability.

[11] discussed that DAT only aims to learn domain-invariant features but ignores extracting task-specific features. In certain situations, the features learned by DAT will have limited discriminative ability in the target domain. To overcome this issue, [11] designed a novel model which contains one feature extractor and two classifiers. This method can be divided into three steps. In the first step, two classifiers and the feature extractor is trained by minimizing cross-entropy loss to predict the labels of samples correctly. In the second step, two classifiers are trained by maximizing the discrepancy between their outputs. In the third step, the feature extractor is learned by minimizing the discrepancy between outputs from outputs of two classifiers. The first step ensures the discriminative ability of learned features, and the second and third steps ensure the domain-invariant ability of learned features.

[21] pointed out that DAT may suffer from model collapse due to the separate design of the label classification task and domain classification task. To alleviate this issue, [21] proposed a novel method for increasing interaction between label and domain classifiers.

More specifically, their adversarial objective encourages mutually inhibitory relations between category and domain predictions. They showed that this novel adversarial objective alleviates the model collapse issues and achieves better domain adaptation ability than DAT.

2.2.3 Generative model-based methods

A certain number of works [22, 23, 24] attempt to utilize a generative model to solve UDA tasks. The generative model is often used for two purposes in the UDA setting. First, some methods [22] utilizes a generative model to perform data augmentation. They design a generative model for synthesizing novel data and augment training data with these synthesized data. Second, some methods [23, 24] introduce an additional task that performs data reconstruction of the source and target data by a generative model. These works show that such additional task is helpful in improving the model's domain adaptation ability.

[22] designed an image-to-image translation model, termed domain flow generation (DLOW), to represent data generating distribution of intermediate domains between the source and target domains. DLOW model takes an image and an additional probability vector as inputs and outputs an image with a style specified by the probability vector. By varying the probability vector, DLOW can synthesize diverse images. They then show that augmenting training data with data drawn from the diverse intermediate domains improves the classification model's domain adaptation ability.

[23] argued that explicitly learning and separating representations that are shared between domains and representations which is unique in each domain can improve the model's domain adaptation ability. Hence, they proposed a special model which contains encoders, decoders, and a classifier. The encoders and decoders are used to realize a data reconstruction task, and the encoders and a classifier are used to classify input samples. By training whole models with the classification and data reconstruction tasks, they showed that the performance on the classification task can be further improved.

[24] proposed a generative model, termed cycle-consistent adversarial domain adaptation (CyCADA), which attempts to overcome domain shifts at both the feature level and pixel level. At the pixel level, this model is trained with a data reconstruction task that attempts to convert data in the source domain into the target domain. Mapping source data into the target domain will alleviate domain shifts in the pixel level. At the feature level, this model utilizes domain adversarial training for learning domain-invariant features. With two-level adaptation, CyCADA shows superior domain adaptation performance than one level adaptation-based method.

2.3 Partial domain adaptation

2.3.1 Motivation and problem setting

With the development of big data evolution, more and more large-scale data sets have emerged, such as ImageNet [25]. Pre-training machine learning models on these large datasets first and then fine-tuning to specific tasks later becomes a popular strategy when we want to build a machine learning model. In the fine-tuning process, there are two issues. First, the specific task we are interested in usually has a smaller label set than big datasets' label sets. Second, we may need to handle the domain shift issue when we retrain the model. In this situation, UDA methods are hard solutions to alleviate domain shifts since UDA methods usually assume the label set in the source and target domains are the same. The partial domain adaptation (PDA) setting is introduced for alleviating domain shifts in this situation [26, 27].

PDA assumes that we have two domains, a source domain P^s_{XY} on $\mathcal{X}^s \times \mathcal{Y}^s$ and a target domain P^t_{XY} on $\mathcal{X}^t \times \mathcal{Y}^t$. The target domain label space is contained in the source domain label space, $\mathcal{Y}^t \subseteq \mathcal{Y}^s$. PDA assumes that we can have a labeled dataset that contains N data-label pairs drawn from domain P^s_{XY}, namely $D_s = \{(x^s_i, y^s_i)\}^N_{i=1}$ with $(x^s_i, y^s_i) \sim P^s_{XY}$. Also, we have a unlabeled dataset that contains M samples drawn from target domain, namely $D_t = \{(x^t_i)\}^M_{i=1}$. The goal of PDA is to train a model with a generalization ability to the target domain.

2.3.2 Partial domain adaptation methods

[26] first introduced the PDA problem, and they proposed Partial Adversarial Domain Adaptation (PADA) to solve the PDA problem. PADA is a variant of domain adversarial training, and the key idea is to learn domain-invariant features through adversarial training. The special feature of this method is to reduce the weight of data with classes that are not contained in the target domain label set during the adversarial training. With this strategy, the PDA setting will become more and more like a UDA setting during the training. Down-weighting the samples from the outlier source classes (classes that are only contained in the source domain) is the most common strategy for solving the PDA settings. A certain number of works follow this line [27, 28, 29].

[10] considered another strategy to solve the PDA problem. Unlike [26], they believe data with outlier source classes are also helpful in improving the model's domain adaptation ability. They augured that although outlier data are not helpful in the classification tasks we are interested in, these data may have meaningful features that can be used in aligning features in different domains. For example, considering we want to train a model for dog-cat classification, the fox images are not helpful for this classification task. However, we can learn features such as the tail and fur from both cat, dog, and fox images. When we map these features from both the source and target domains into the same space, we can obtain domain-invariant features. Models built on such domain-invariant features will have good domain adaptation ability. With this intuition, [10] proposed Implicit Semantic Response Alignment (ISRA) to discover disentangled features, then they utilized adversarial learning to map these features into the same space. During this process, the feature extractor can learn domain-invariant features. Experimental results suggest this method lead to outstanding domain adaptation ability in the PDA setting.

2.4 Source-free domain adaptation

2.4.1 Motivation and problem setting

When we utilize UDA methods to alleviate domain shifts, we need to collect data from both the source and target domains. However, this strategy is often difficult to use in certain situations. For example, the source and target domains refer to two different hospitals.

Applying UDA methods in this situation means two hospitals must share sensitive medical data, which is generally difficult to do in reality. Several works [30, 31] have made efforts to reduce private data sharing by introducing the source-free domain adaptation (SFDA) setting. In the SFDA setting, instead of collecting data from the source domain, a model trained using the source domain data is given.

Formally, SFDA assumes that we have two domains, a source domain P^s_{XY} on $\mathcal{X}^s \times \mathcal{Y}^s$ and a target domain P^t_{XY} on $\mathcal{X}^t \times \mathcal{Y}^t$. The target domain label space is the same as the source domain label space. SFDA assumes that we can have a pre-trained model f_s, which is trained on source domain data. Also, we have a unlabeled dataset that contains M samples drawn from target domain, namely $D_t = \{(x^t_i)\}^M_{i=1}$. SFDA aims to train a model with a generalization ability to the target domain.

2.4.2 Source-free domain adaptation methods

[30] first considered the SFDA setting in order to protect data privacy while overcoming domain shifts. Their key idea to solve SFDA is to utilize the pre-trained source model to assign labels to unlabeled target data. Then, train a target model with target data and their pseudo labels. Due to domain shifts, the source model is not able to assign correct labels to all target data. Hence, [30] hypothesized and verified that samples with low self-entropy measured by the pre-trained source model are more likely to have correct pseudo labels. Another issue is that only a few target data have reliable pseudo labels. To address this issue, they proposed a self-learning framework that progressively updates the target model. Surprisingly, the model trained by their method achieves outstanding domain adaptation ability without source data.

[32] pointed out that the pseudo-labeling-based method often results in a certain number of samples with the wrong label. Machine learning models trained on such data will have limited discriminative ability. On the other hand, they found although conditional generative adversarial networks (cGANs) are also trained on labeled data, the performance of cGANs is rarely affected by data with wrong labels. Hence, they proposed a variant of cGANs to synthesize samples with the target style. They proposed to jointly train the generative model and the classifier. The classifier will continuously improve the domain adaptation ability by training on target samples that are continuously generated by the generative model.

[33] argued that if the source model and target model can both learn a certain type of representation, termed domain generic representations, the SFDA task can be solved. The domain generic representations have two characteristics: first, the representations are invariant between source and target domains; second, the representations have a good discriminative ability. They then extended a data augmentation technique, termed mixup [16], to force the source model to learn domain generic representations. Then, when they learn target models, they also proposed to utilize this mixup technique. This technique can be combined with other SFDA methods since this method only perform data augmentation when they learn the source and target models. Although this method only makes a simple modification, it shows outstanding domain adaptation performance in the SFDA setting.

2.5 Domain generalization

2.5.1 Motivation and problem setting

Till now, we introduced UDA, PDA, and SFDA settings. A common assumption of these settings is the ability to specify the target domain and obtain data from the target domain. However, we often need to deal with the unseen target domain. If the target domain is an unseen domain, we can not obtain data from this domain. UDA, PDA, and SFDA methods may be hard solutions to alleviate domain shifts. Domain generalization (DG) setting is designed to improve the model's generalization ability on the unseen target domains [34]. To achieve this, DG assumes that data from multiple source domains are available and utilize multiple source domains to train a model that can generalize to unseen target domains.

Formally, assume we have a set of multiple datasets $\mathcal{D} = \{D_k\}_{k=1}^{K}$. Domain generalization assumes that each dataset D_k contains i.i.d. data-label pairs drawn from domain P_{XY}^{k}, namely $D_k = \{(x_i^k, y_i^k)\}_{i=1}^{N_k}$ with $(x_i^k, y_i^k) \sim P_{XY}^{k}$. We assume that all domains have the same label set. The DG task aims to train a classifier with a generalization ability to unseen domains by utilizing given datasets for training.

2.5.2 Domain generalization methods

In Chapter 3, we propose a data augmentation-based method for solving DG tasks. Hence, this section will mainly introduce data augmentation-based DG methods.

Data augmentation is one of the most common ways to solve the DG task. We divided these methods into three main categories: adversarial data augmentation methods [35], generative model-based data augmentation methods [36, 22], and domain-invariance-based methods [37, 38]. Also, there are some methods that do not fall into these categories [8, 12].

Adversarial data augmentation aims to augment training data with samples drawn from a fictitious domain that is "hard" for the model to classify, expecting that training with such augmented data will improve the classifier's domain generalization ability. [35] proposed a minimax optimization problem to find a distribution that is hard for the classifier to predict, i.e., the worst-case distribution, under the condition that the Wasserstein distance from the original data generation distribution to the synthesized distribution is within a certain distance. Their method, termed GUD, then trains the classifier with the worst-case distribution. They reported that the generalization ability to unseen domains is improved by training the model with the worst-case distribution.

Generative model-based data augmentation [36, 22] methods utilize the generative model to synthesize images with novel domains and augment training data with these synthesized images. Given training samples drawn from multiple domains that are mutually semantically different from each other, [36] proposed a generative model that can synthesize novel domains that are apart from every training domain in the sense of the Wasserstein distance while maintaining semantic information. Then, they augment the training data with samples generated by the proposed generative model. They report that their proposal, termed L2A-OT, can improve the model's generalization ability. [22] proposed a generative model named the domain flow generator, termed DLOW, which can synthesize intermediate domains among semantically different domains. They report that augmenting training data with samples uniformly drawn from the synthesized domains is beneficial to improving the classification model's domain generalization ability. These works train the generative model and the classification model independently. The data provided by the generative model is not directly designed for the classifier.

Some works attempt to design the generative model with the training of the classifier. [39, 40] proposed jointly training the generative model for data augmentation and the classification model for improving the classification model's domain generalization ability. During the training process, [39] (Wang2021) attempted to train the generator to maximize the mutual information between the synthesized images and the source images. [40] (yang2021) attempted to train the generator to maximize the divergence between features extracted from synthesized images and original training data. Although different strategies are proposed in these methods, they both aim to synthesize the augmented data that can result in diverse features for the classifier. The experimental results show that these methods achieve better domain adaptation ability than independently trained generative model-based methods.

A recent work [41] proposed a framework called MBDG. This work considered the same training objective as the adversarial data augmentation; they want to improve the classification model's performance over the worst-case domain. However, unlike [35], which attempted to solve a minimax optimization problem, [41] made several assumptions on the data generation process and the labeling mechanism to transform the minimax optimization problem into a relaxed problem that removes the inner maximization. Then, they proposed an algorithm to solve the relaxed problem by using a generative model that simulates the assumed data generation process. This work is related to both adversarial data augmentation and generative model-based augmentation. It utilizes samples generated by a generative model to improve the model's performance over the worst-case distribution. Empirically, they report that their algorithm improves the classification model's domain generalization ability on several datasets. However, their experimental results also show that MBDG only leads to good domain generalization ability in certain domains, such as the sketch domain in the PACS dataset, while in certain domains, this method does not perform well.

Domain-invariance-based methods attempt to augment training data without distinguishable domain features [37, 38]. [37] proposed CrossGrad, which augments training data with adversarial examples obtained from a special classifier, termed the domain classifier, that distinguishes training domains. The adversarial examples obtained from the domain classifier are expected to have few domain-level features. [38] followed this idea and proposed DDAIG; they utilized an additional neural network to generate the domain classifier's adversarial perturbation instead of the gradient-based perturbation. These studies aim to augment training data with samples that fool the domain classifier. By this approach, they can obtain samples with domain-invariance features.

In addition to those discussed above, [8] (JiGen) forced the model to solve additional jigsaw puzzles by augmenting training data with image patches and achieving outstanding performance on the DG task. [12] (Mix-style) proposed augmenting training data with features synthesized by linear combinations of features learned by the classifier.

Another common approach to tackling the DG problem is a special type of representation learning, domain-invariance feature learning [5, 18, 19, 42, 43]. The intuition of these works is that invariant features learned from different domains can be generalized to unseen domains. For instance, a popular technique in this line is domain adversarial learning [5] and its variant [18, 19], which attempts to learn a representation that cannot be distinguished among multiple domains. In addition, a method that has recently gained attention is invariant risk minimization (IRM) [42, 43], which aims to learn a representation such that the optimal linear predictor on top of this representation is identical over all source domains. Domain-invariance features and data augmentation are mutually complementary. A certain number of data augmentation methods can be combined with domain-invariance feature learning methods [36, 12].

2.6 Zero-shot domain adaptation

2.6.1 Motivation and problem setting

In some situations, it is possible to specify the target domain, but it is hard to collect task-relevant data from the target domain. For example, if task-relevant data are private, it is expensive or time-consuming to remove private information. To alleviate this issue, zero-shot domain adaptation (ZSDA) methods aim to utilize task-irrelevant data from both the source and target domain to learn domain shifts between source and target domains. Then, learned domain shift can be utilized to improve the model's generalization ability on the target task-relevant data [9].

In the zero-shot domain adaptation (ZSDA) task, we have two domains, a source domain D_s, and a target domain D_t. The data samples are also drawn from two tasks, a task of interest (ToI), which we call a relevant task, and an irrelevant task (IRT). ToI and IrT have different label sets $\mathcal{C}^r$ and $\mathcal{C}^{ir}$, that is $\mathcal{C}^r \cap \mathcal{C}^{ir} = \emptyset$. For the ToI, we have sample-label pairs from the source domain, namely we have $D_s^r = \left\{\left(x_{s_j}^r, y_{s_j}^r\right)\right\}_{j=1}^N$ where $y_s^r \in \mathcal{C}^r$. For the IRT, we have sample-label pairs from both source and target domains, namely we have $D_s^{ir} = \left\{\left(x_{s_j}^{ir}, y_{s_j}^{ir}\right)\right\}_{j=1}^N$ where $y_s^{ir} \in \mathcal{C}^{ir}$ and $D_t^{ir} = \left\{\left(x_{t_j}^{ir}, y_{t_j}^{ir}\right)\right\}_{j=1}^N$ where $y_t^{ir} \in \mathcal{C}^{ir}$, respectively. We also use a domain label to indicate the domain information of a sample. For x_s^r and x_s^{ir}, the corresponding domain label d_s is 0, for x_t^{ir} the corresponding domain label d_t is 1. Given $D_s^{ir}, D_t^{ir}, D_s^r$, the goal of the ZSDA task is to learn a model that can generalize to the distribution of target relevant data, namely distribution of $D_t^r = \left\{\left(x_{t_j}^r, y_{t_j}^r\right)\right\}_{j=1}^N$.

2.6.2 Zero-shot domain adaptation methods

There are mainly two types of methods to solve ZSDA tasks; the generative model-based methods and the domain-invariant feature learning-based methods. The generative model-based methods aim to train a generative model, such as generative adversarial networks (GAN) [44], for synthesizing virtual data that can represent the target ToI data [45, 13]. [9] first introduced the ZSDA setting. They proposed to utilize deep neural networks to learn the domain shifts, and then the learned domain shift is used to convert source data into the target domain. [45] extended conditional coupled generative adversarial networks (CoCoGAN) [46] to learn the joint distribution of data samples across domains and tasks. Such a generative model is able to synthesize task-relevant samples with target styles. With these synthesized samples, the domain shifts issue can be alleviated.

Another strategy for solving ZSDA tasks is the domain-invariant feature learning-based methods [47]. This work aims to force the classifier based on the domain-invariant features to make predictions. Domain-invariant features refer to features that do not change with the domain. If a model is only based on domain-invariant features to make predictions, then this model should generalize to different domains. [47] can be divided into two stages. In the first stage, this method extends domain adversarial training [5] to learn domain-invariant features. In the ZSDA setting, since we do not have samples for ToI in the target domain, the domain-invariant features are mainly learned with IrT data. Hence, in this stage, learned features might not generalize to the ToI approximately. To alleviate this issue, in the second stage, [47] design an attention-based [48] module to fine-tune the model learned in the first stage so that bias caused by the ZSDA setting can be alleviated. In this sense, [47] first learns biased features and then removes the bias in the second stage.

In addition to those discussed above, some works consider special zero-shot domain adaptation settings. Instead of obtaining task-irrelevant data from the target domain, they assume some special information can be obtained from the target domain. [14] assumes that the domain shift can be parameterized by attributes, and such attributes are known during model training. For example, if the rotation causes the domain shift, then the rotation angle in each domain is their corresponding attribute. They then utilize source and target domain attributes to improve the model's generalization ability without target data. [49] focus on a specific domain shift, which is the illumination shift. They propose a color invariant layer with this setting to improve the model's robustness against illumination changes.

2.7 Summary and discussion

In this Chapter, we have introduced five settings and detailed methods that aim to alleviate domain shifts.

1. In Section 2.2, we introduced unsupervised domain adaptation, which is the most basic setting to alleviate domain shift. We also introduced domain adversarial training (DAT), which is one of the most important methods to alleviate domain shift. DAT has inspired a lot of subsequent works. In Chapter 4, our proposal also follows this line.

2. Also, in Section 2.2, we have introduced how a generative model can be used to overcome domain shifts. A generative model will be used for data augmentation or solving an additional data reconstruction task. In Chapter 3, we also utilize a generative model for data augmentation.

3. In Section 2.3, we have introduced the partial domain adaptation (PDA) setting. The distinctive feature of the PDA is that the source and target domains have different label spaces. We also review strategies for solving the PDA problem. Although these methods work well when the target label space is a sub-space of source label space, these methods can not handle situations when the source and target label space have no intersection. In Chapter 4, we will consider how to alleviate domain shifts when the source and target label space have no intersection.

4. In Section 2.4, we introduced the source-free domain adaptation (SFDA) setting and algorithms. This setting can protect sensitive data in the source domain and overcome

domain shifts. In Chapter 4, we also investigate how to alleviate domain shifts without obtaining sensitive data, but we focus on protecting sensitive data in the target domain.

5. Section 2.5 introduced the domain generalization (DG) setting and algorithms. This section is directly related to Chapter 3. Chapter 3 also focuses on solving the DG problem. In Chapter 3, we will summarize the limitations of data augmentation methods presented in this section and propose a novel data augmentation method to overcome these limitations.

6. Section 2.6 introduced the zero-shot domain adaptation (ZSDA) setting and algorithms. These methods can be divided into two categories, the generative model-based method and the representation learning-based method. We will show the limitations of strategies shown in Section 2.6, and will provide a novel representation learning-based method in Chapter 4.

Chapter 3

Domain Generalization via Adversarially Learned Novel Domains

This chapter proposes a data augmentation method, generative adversarial domain augmentation (GADA), for domain generalization tasks. GADA aims to synthesize data with both semantically diverse styles and difficulty.

3.1 Introduction

In reality, machine learning models often need to handle unseen target domains. In this situation, it is difficult to collect data from the target domain. Hence, domain adaptation methods that assume unlabeled data from the target domain are obtainable are hard solutions. Domain generalization (DG) is designed to solve this problem. DG methods attempt to utilize multiple training domains to build a machine learning model that can generalize to the unseen target domain.

As we already discussed in Section 2.5, data augmentation is one of the most common ways to address domain generalization (DG) problems [36, 22, 35, 41]. By providing diverse data to the main classifier, the model will learn more domain-invariant features. Among various data augmentation methods, we focus on two types: adversarial data augmentation [35] and generative model-based data augmentation [22, 36]. Although these methods exhibit the good ability to solve DG problems, these methods have some limitations.

1. An adversarial data augmentation method [35] shows that augmenting difficult samples can improve the model's domain generalization ability. However, the augmented data considered in this method are bounded by the Wasserstein distance, resulting in samples semantically similar to the original training samples. This data augmentation method will achieve limited generalization ability when unseen domains are semantically different from the training distribution.

2. Generative model-based data augmentation methods [22, 36] utilize generative models to synthesize augmented samples. These works considered different strategies to train the generative model, they both aimed to improve the diversity of the generated images. However, these works have never considered data augmentation with the

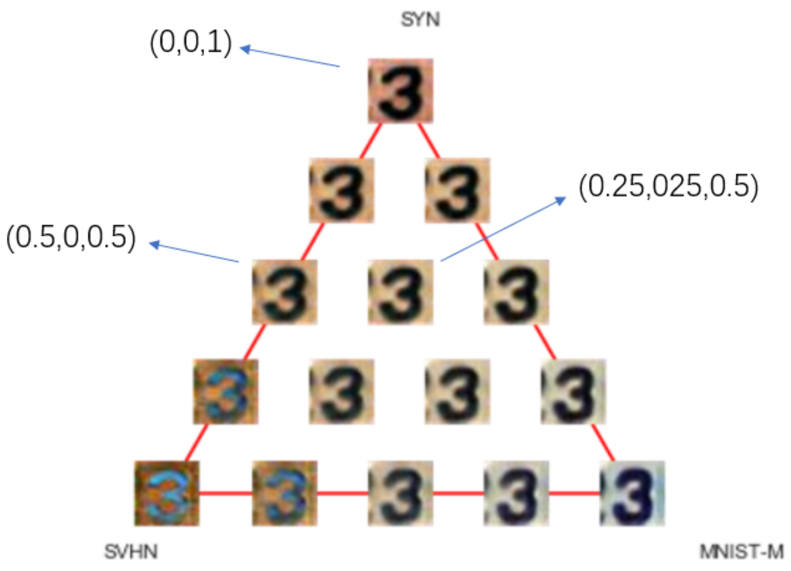


Figure 3.1: An illustration of the DLOW-B model and three domain codes. This model is trained on three different domains, so the domain code is a 3-dimensional probability vector. We visualize this model on a 2-dimensional simplex. Here, each point in the 2-dimensional simplex corresponds to a 3-dimensional domain code. By varying domain codes, the generated images exhibit different styles.

worst-case distribution using generative models.

We aim to incorporate the advantages of both adversarial data augmentation and generative model-based methods: it leverages the worst-case distribution with generative models. To attain these, we need a generative model that can draw samples from distributions representing semantically different domains. Additionally, we need to control the data-generating distribution to find the worst-case distribution over semantically different multiple domains. Considering these requirements, we propose a conditional image-to-image translation model, which allows us to generate images by specifying a mixture of semantically different styles over multiple training domains. Namely, we propose to use a special GAN (generative adversarial network) model named DLOW-B that allows controlling the style of generated images by a probability vector called domain code, which represents the mixture of multiple domains[1]. By varying the domain codes, the styles of the generated samples can be changed. Figure 3.1 shows an example of a DLOW-B model and domain codes.

With this generative model, we then need to find the worst-case data-generating distribution by controlling the domain code. Thus, we propose two plausible strategies to represent the domain-code distribution: the first is using a Dirichlet distribution with learnable parameters to represent the domain-code distribution; the second is using a two-layer neural network, termed domain-code NN, which takes Gaussian noise as input, and its output

[1]In different papers, domain code is named differently, such as domain label [50], and domainness [22], we collectively refer to as domain code. Notably, our framework works with any generative model that satisfies the two requirements denoted above, such as [51].

represents the domain-code distribution. Then, we optimize the parameters of the Dirichlet distribution or the parameters of the domain-code NN so that its output represents a distribution that gives the lowest predictive performance with the current classification model. Additionally, we train the classification model with training data augmented with samples drawn from the worst-case distribution represented by the learnable Dirichlet distribution or the domain-code NN. We denote our framework with a learnable Dirichlet distribution as GADA-D and another as GADA-NN.

Experimental results show that GADA-D and GADA-NN surpass the current data augmentation methods in three DG tasks with multiple training domains in most cases. Furthermore, further ablation studies and visualization results show that the proposed adversarial training strategy can learn a 'hard' domain-code distribution, which helps achieve good domain generalization ability.

The novelty of this study is summarized as follows.

1. We propose a flexible framework that utilizes the worst-case data-generating distribution represented by a generative model. More specifically, we propose utilizing domain codes to control the behavior of the generative model. The data-generating distribution is consequently adapted by optimizing the domain-code distribution so that it helps to improve the domain generalization ability.

2. Our proposal serves as a data augmentation-based method. Due to the flexibility of the data augmentation methodology, our framework can be easily combined with other domain generalization algorithms. We remark that, to the best of our knowledge, our proposal is the first data augmentation-based method that exploits the worst-case data-generating distribution characterized by generative models that can represent semantically different domains.

3. The experimental results suggest that the proposed method shows superior domain generalization ability.

3.2 Preliminary

3.2.1 Cycle-Consistent Adversarial Networks

In this work, we utilize a variant of cycle-consistent adversarial networks (CycleGAN) [52] to synthesize the augmented samples. We also modify a special module in the CycleGAN, cycle consistent loss, to enforce the generative model to synthesize images with diverse styles. Hence, we first introduce the CycleGAN model. As Figure 3.2 shows, CycleGAN contains two generators, G_{ST} and G_{TS}, and two discriminators, D_S and D_T. G_{ST} is designed to convert an image in the source domain to the target domain, and G_{TS} is designed to convert an image in the target domain to the source domain. D_S aims to judge whether an image is from the source domain, and D_T aims to judge whether an image is from the target domain.

Similar to GAN's design, the generators are trained to fool the discriminators. The training loss is shown in the following.

$$\mathcal{L}_{\text{GAN}}\left(G_{TS}, D_S\right) = \mathbb{E}_{x_s \sim p^s_{\mathcal{X}\mathcal{Y}}}\left[\log D_S(x_s)\right] + \mathbb{E}_{x_t \sim p^t_{\mathcal{X}\mathcal{Y}}}\left[\log\left(1 - D_S(G_{TS}(x_t))\right)\right]$$

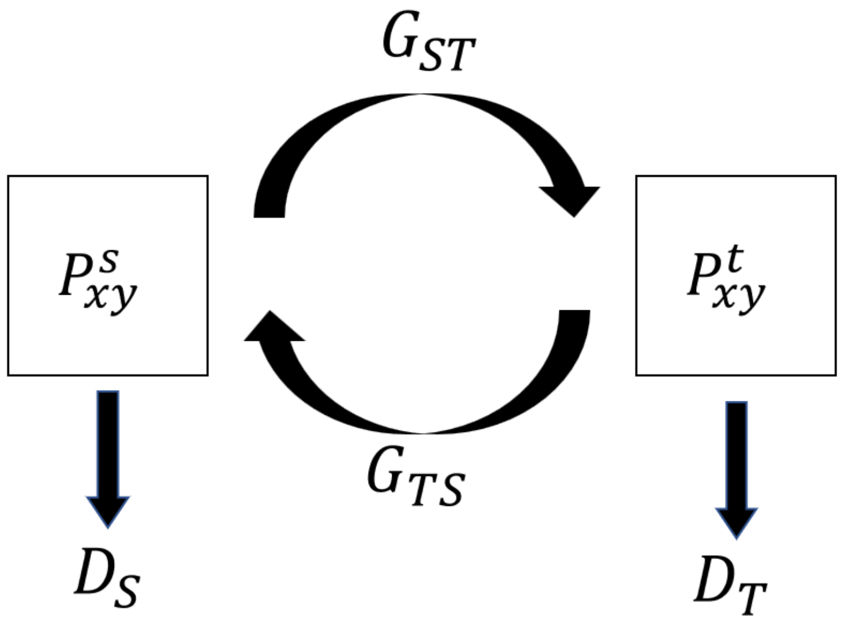


Figure 3.2: An illustration of the CycleGAN model and three domain codes.

$$\mathcal{L}_{\mathrm{GAN}}\left(G_{ST}, D_T\right)=\mathbb{E}_{x_t \sim p^t_{\mathcal{X}\mathcal{Y}}}\left[\log D_T(x_t)\right]+\mathbb{E}_{x_s \sim p^s_{\mathcal{X}\mathcal{Y}}}\left[\log \left(1-D_T(G_{ST}(x_s))\right)\right]$$

CycleGAN also proposed a cycle consistent loss to preserve content information in the input image. The key idea is that if the input image can be recovered from the synthesized image, then the content information should be preserved in the synthesized iamges. The cycle-consistent loss is shown in the following.

$$\mathcal{L}_{\mathrm{cyc}}\left(G_{ST}, G_{TS}\right)=\mathbb{E}_{x_s \sim p^s_{\mathcal{X}\mathcal{Y}}}\left[\|G_{TS}(G_{ST}(x_s))-x_s\|_1\right]$$

$$\mathcal{L}_{\mathrm{cyc}}\left(G_{TS}, G_{ST}\right)=\mathbb{E}_{x_t \sim p^t_{\mathcal{X}\mathcal{Y}}}\left[\|G_{ST}(G_{TS}(x_t))-x_t\|_1\right]$$

With the above training objectives, CycleGAN is able to convert images from the source domain to images with target styles.

3.2.2 Domain flow generator

In this work, we propose an image-to-image translation model to convert source domains into novel domains. The proposed model is built on DLOW [22], so we introduce the DLOW model in this section. DLOW is trained on multiple domains and takes one of the training domains as the source domain. It can transfer an image from the source domain into an image with an intermediate style among all training domains. Moreover, [22] considers a domain-code space that encodes a mixture of multiple training domains. The domain code z controls the style of the generated images. Assume we have K training domains $T_1, ..., T_K$. z is defined as a $K-$dim vector, $z=[z_1,\ldots,z_K]$ with $\sum_{k=1}^{K} z_k = 1, z_k \geq 0$. Each element z_k represents the degree of relevance to the target domain T_k. By specifying z at generation time, DLOW can convert an image in the source domain into an image with a style specified by z, which realizes the interpolation of all training domains. One illustration of the domain codes is shown in Figure 3.1.

The objective function of the DLOW model contains the adversarial loss term and cycle consistency loss, and DLOW results in two maps: G_{ST}, which learns a mapping from source domain S to target domains $T_1, ..., T_K$, and G_{TS}, which learns a mapping from $T_1, ..., T_K$ to S. G_{TS} is used to assist the training of G_{ST} for persevering categorical semantics, which is reflected in the cycle consistency loss. We only utilize G_{ST} in our proposal, so we take G_{ST} as an example to explain DLOW. For more details, please refer to [22].

For K target domains, the DLOW model introduces K discriminators, $D_{T_1}, ..., D_{T_K}$, that distinguish generated images and images from domain $T_1, ..., T_K$, respectively. Additionally, the discriminator D_S for source domain S distinguishes generated images and images from the source domain. Then, for each k, the adversarial loss between S and T_k can be formulated as:

$$\mathcal{L}_{adv}^{k}\left(G_{ST}, D_{T_k}\right) = \mathbb{E}_{x_{T_k} \sim p^{T_k}}\left[\log D_{T_k}(x_{T_k})\right] + \mathbb{E}_{x_s \sim p^S}\left[\log\left(1 - D_{T_k}(G_{ST}(x_s, z)\right)\right] \tag{3.1}$$

The full adversarial loss is defined as:

$$\mathcal{L}_{adv} = \sum_{k=1}^{K} z_k \cdot \mathcal{L}_{adv}^{k}\left(G_{ST}, D_{T_k}\right) \quad \text{s.t.} \quad \sum_{k}^{K} z_k = 1, \quad z_k \geq 0 \tag{3.2}$$

The cycle consistency loss between S and T_k is defined as:

$$\mathcal{L}_{\text{cyc}}\left(G_{ST}, G_{TS}\right) = \mathbb{E}_{x_s \sim p^S}\left\|G_{TS}\left(G_{ST}(x_s, z), z\right) - x_s\right\|_1 \tag{3.3}$$

Then the full objective L is defined as:

$$\mathcal{L} = \mathcal{L}_{adv} + \lambda \mathcal{L}_{cyc},$$

where λ is a hyperparameter used to balance the two losses in the training process, and similar losses are also defined for the opposite direction G_{TS}. During the training, in each epoch, we sample z from a fixed data distribution, and the generators are trained on minimizing the full objective while discriminators are trained on maximizing the full objective.

3.3 Proposal

In this section, we propose a generative model-based adversarial data augmentation method, termed GADA (**G**enerative **A**dversarial **D**omain **A**ugmentation), for the domain generalization task. GADA aims to train the main classifier with the worst-case distribution represented by the generative model. To achieve this goal,

1. We first propose a conditional generative model, domain flow with better cycle (DLOW-B), to synthesize augmented samples. This model takes an image and a probability vector, termed domain code, as inputs and outputs images with the style specified by the domain code. See Section 3.3.1 for more details.

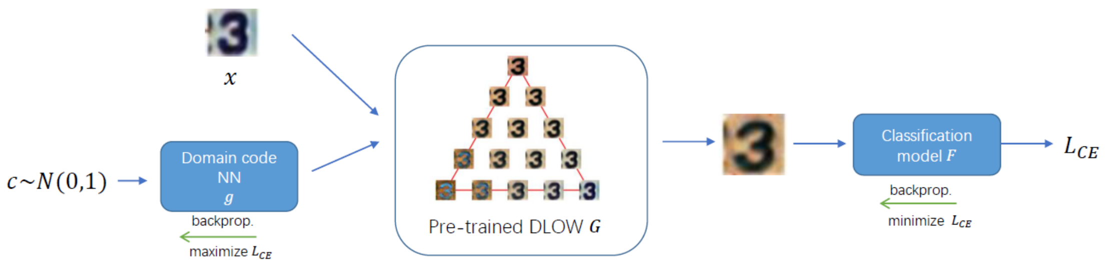


Figure 3.3: An overview of GADA with the domain-code NN. The domain-code NN takes Gaussian noise as input and outputs a valid domain code. The pretrained DLOW takes both image x and domain code as input and synthesizes an image with the style specified by the domain code. The classification model is trained with synthesized images. We optimize the domain code-NN to maximize the classification loss so that it can represent the worst-case distribution.

2. We then propose utilizing the distribution of domain codes to represent the data-generating distribution of the DLOW-B model. During this process, we consider two types of methods to represent the distribution of domain codes. One uses a Dirichlet distribution, the other uses a neural network, termed domain-code NN, to represent the distribution of domain codes. See Section 3.3.2 for more details.
3. Finally, we propose a min-max optimization problem to enforce the generative model to synthesize hard examples by finding the worst-case distribution of domain codes and, at the same time, train the classifier on the worst-case distribution. See Section 3.3.3 for more details.

The overview of GADA with domain-code NN is shown in Figure 3.3.

3.3.1 Domain flow generator with better cycle

As discussed above, we propose controlling the domain-code distribution to control the behavior of the generative model so that we can find the worst-case distribution. Hence, we need a generative model that is conditioned by domain codes. The DLOW model already satisfies our requirements. Empirically, however, we found that the synthesized images of the DLOW lacked diversity. The style of the generated images does not necessarily change with the domain code change.

[53] showed that the cycle consistency loss limits the diversity of the generated images. The cycle consistency loss forces generated images to be recovered to the original image, so the generated images usually look like the original images. To alleviate this issue, [53] proposed a weak version cycle consistency loss. Instead of forcing generated images can be recovered, they enforce that features extracted by the discriminator can be recovered. Inspired by the weak version of cycle consistency loss, we replace the cycle consistency loss of the original DLOW model with the weak cycle consistency loss. We call this new model DLOW-B (domain flow generator with better cycle). The weak cycle consistency loss is shown in the following:

$$\begin{aligned}\tilde{\mathcal{L}}_{\text{cyc}}\ (G_{ST}, G_{TS}, \gamma) = \mathbb{E}_{x_s \sim p^S}(\gamma \,\|f_{D_S}(G_{TS}(G_{ST}(x_s, z), z)) - f_{D_S}(x_s)\|_1 \\ + (1-\gamma)\|G_{TS}(G_{ST}(x_s, z), z) - x_s\|_1). \quad (3.4)\end{aligned}$$

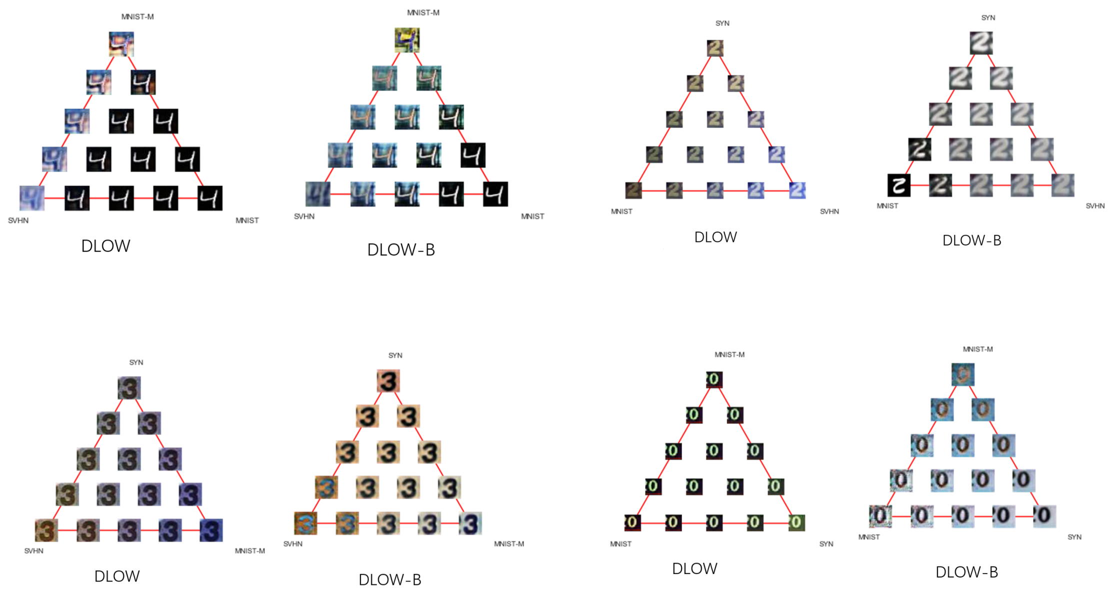


Figure 3.4: Comparison of DLOW and DLOW-B

Here, $f_{D_S()}$ is the feature extractor of the source domain's discriminator. γ is a hyperparameter used to balance the pixel-level cycle consistency loss and feature-level cycle consistency loss.

A comparison of the image quality of DLOW-B and DLOW is shown in Figure 3.4. These DLOW and DLOW-B models are trained on the same datasets, digits-DG. Digits-DG contains four digital dataset with different styles. Detailed introduction of digits-DG is shown in Section 3.4.1. Subjectively DLOW-B generates images with a wider variety of colors.

3.3.2 Representative data generating distribution by domain-code distribution

In this study, we aim to train the classifier with a min-max optimization objective, (3.5). More specifically, during the training process, we aim to optimize P_G, that is, the data-generating distribution of the generative model G so that the loss function is maximized. Additionally, we optimize the parameters of the classification model F so that the loss of the classification task is minimized. We can interpret this optimization as distributionally robust optimization [54]:

$$\min_{\theta_F} \sup_{P_G \in \mathcal{P}} \mathbb{E}_{P_G}[\ell(\theta_F; (X, Y))]. \tag{3.5}$$

Here, θ_F are the parameters of the classification model F. ℓ is the loss function, such as the cross-entropy loss. G represents a generative model, and P_G refers to a sample generating distribution of G, (X, Y) is sample-label pair drawn from P_G. We expect that the classification model can achieve good performance on P_G by optimizing the classification model through (3.5).

One difficulty in realizing optimization of (3.5) is how to control the generative model's behavior so that P_G becomes challenging for the classification model. To solve this difficulty, we propose to use a generative model, termed DLOW-B, as our generative model G. Then, by optimizing the distribution of domain codes, the data-generating distribution will be changed accordingly. We, therefore, seek a way to optimize the domain-code distribution.

It is natural to first consider using an elemental distribution to represent the domain-code distribution. Then, the domain-code distribution can be optimized by optimizing that elemental distribution. Since a domain code is a $K-$dimensional probability vector, we want a distribution on the simplex. The Dirichlet distribution is an elemental distribution whose random variable is defined on the simplex. It is parameterized by a vector of positive-valued parameters $\alpha = [\alpha_1, \dots, \alpha_K]$ with $\alpha_k \geq 0$. Hence, we can readily optimize the Dirichlet distribution by optimizing α. We propose utilizing the Dirichlet distribution to represent the distribution of domain codes as our first choice. We denote our framework with this strategy GADA-D. The advantage of GADA-D is that few parameters need to be optimized. Since the number of parameters of the Dirichlet distribution is the same as the number of dimensions of the domain code, which is generally less than 10.

Furthermore, considering that the Dirichlet distribution may not be able to represent a complex distribution over the simplex, we provide another strategy. The success of deep learning in recent years has shown that neural networks have great representation ability. In particular, GAN [44] showed that a neural network can learn a complex distribution from finite samples. Hence, we propose to use a neural network g, termed domain-code NN, to represent the distribution of domain codes. Inspired by [44], we introduce a Gaussian prior $c \sim \mathcal{N}(0, 1)$ to introduce randomness into the neural network. We design the last layer of the domain-code NN as a softmax layer so that it outputs a probability vector. Therefore, by choosing c randomly, $g(c)$ works as a distribution in the domain-code space. Although the number of parameters of the domain-code NN will be large, the domain-code NN have better representation ability than a Dirichlet distribution. When the multiple training domains are complex, this strategy can better represent the domain-code distribution. We call our framework with this strategy GADA-NN.

3.3.3 Generative adversarial domain augmentation

In this section, we propose an algorithm to realize the min-max optimization problem (3.5) with the DLOW-B model G and the idea of utilizing domain codes.

GADA-D: We denote a Dirichlet distribution as $\mathcal{B}()$, and its parameters as α. Here, α is defined as a $K-$dimensional vector, $\alpha = [\alpha_1, \dots, \alpha_K]$ with $\alpha_k \geq 0$. K is the number of given training domains. Then, by utilizing the distribution of domain codes to represent the data generating distribution, the objective of (3.5) becomes:

$$\min_{\theta_F} \sup_{\alpha} \mathbb{E}_{\substack{z \sim \mathcal{B}(\alpha), \\ (x,y) \sim D_{\text{source}}}} [\ell(\theta_F; (G(x, z), y))] \tag{3.6}$$

where (x, y) is an input-label pair drawn from source domain D_{source} (DLOW-B is designed to take samples from one training domain as input, and we denote that the training domain as the source domain which is randomly selected among all training domains), F is the main classifier, $c \sim \mathcal{N}(0, 1)$, and ℓ is the cross-entropy loss.

In the inner maximization of (3.6), we directly optimize the parameters of the Dirichlet distribution α to represent the worst-case distribution of domain codes. To optimize the parameters of the Dirichlet distribution, we utilize the implicit reparameterization technique

[55]. With outer minimization, F is trained to minimize the classification loss of the hardest samples.

GADA-NN: As discussed in Section 3.3.2, the domain-code NN g takes Gaussian noise c as input. By choosing c randomly, $g(c)$ works as a distribution in the domain-code space. With this setup, $G(x, g(c))$ works as a sample generator where two sources of randomness are involved; one is the choice of x, a sample in the source domain, and the other is c. Therefore, by utilizing the domain-code distribution to represent the data-generating distribution, the objective of (3.5) becomes:

$$\min_{\theta_F} \sup_{\theta_g} \mathbb{E}_{\substack{c \sim \mathcal{N}(0,1), \\ (x,y) \sim D_{\text{source}}}} [\ell(\theta_F; (G(x, g(c)), y))] \tag{3.7}$$

where c is $K-$dim Gaussian noise, $c \sim \mathcal{N}(0,1)$, ℓ is the cross-entropy loss, and g is the domain-code NN which takes c as input and outputs a domain code. Since the DLOW-B model requires the domain code to be a probability vector, we design g to have a softmax layer as the last layer.

In the inner maximization of (3.7), the domain-code NN g is trained to provide domain codes so that the classification loss of model F is maximized. In other words, g is trained so that it generates the hardest samples to classify under the current classification model F. With the outer minimization, F is trained to minimize the classification loss of the hardest samples.

To solve the above two bi-level optimization problems (3.6) and (3.7), we follow GAN's training strategy and alternatively train the domain-code NN g or the parameters of the Dirichlet distribution α and the classification model F. We first minimize the objective with regard to F, then maximize the objective with regard to g or α, and repeat this process.

Moreover, since the image quality of samples generated by the generative model is not as good as the original training data, we train the classification model on both given training datasets $\mathcal{D} = \{D_k\}_{k=1}^{K}$ and samples generated by G. Empirically, we find that this leads to better domain generalization ability, as in [35]. Therefore, the actual objective function is given by:

$$\min_{\theta_F} \sup_{\alpha} \mathbb{E}_{\substack{z \sim \mathcal{B}(\alpha), \\ (x,y) \sim D_{\text{source}}}} [\ell(\theta_F; (G(x, z), y))] + \lambda \sum_{k=1}^{K} \mathbb{E}_{(x^k, y^k) \sim D_k} (\ell(\theta_F; (x^k, y^k))). \tag{3.8}$$

or

$$\min_{\theta_F} \sup_{\theta_g} \mathbb{E}_{\substack{c \sim \mathcal{N}(0,1), \\ (x,y) \sim D_{\text{source}}}} [\ell(\theta_F; (G(x, g(c)), y))] + \lambda \sum_{k=1}^{K} \mathbb{E}_{(x^k, y^k) \sim D_k} (\ell(\theta_F; (x^k, y^k))). \tag{3.9}$$

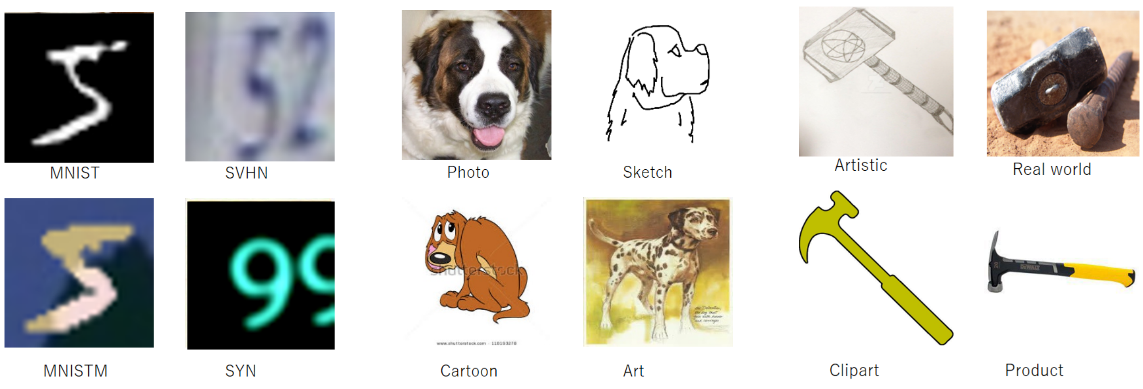
Here, λ is used to balance the weight of augmented data and the weight of given training data. The whole training process of GADA-NN is shown in Algorithm 1.

3.4 Experiments

3.4.1 Experimental setting

Dataset We evaluate our proposed method on three benchmarks. The first benchmark is digits-DG, which contains four digits datasets including MNIST [2], SVHN [56], MNIST-M [1], and SYN [1]. These datasets all contain digit images with labels from 0 to 9, but

```
Algorithm 1: Proposed method: GADA-NN
Input: Pretrained generative model $G$, given multiple training datasets $\mathcal{D} = \{D_k\}_{k=1}^{K}$,
       selected source dataset $D_{\text{source}}$, number of steps to apply to the classifier $t$,
       training iteration $n$, loss function $\ell$, weight to balance augmented samples and
       given training samples $\lambda$
Output: Parameters of classification model $\theta_F$
$i = 0$;
for $i < n$ do
    $j = 0$;
    for $j < t$ do
        Sample minibatch of $m$ Gaussian noise samples $\{c_1, \dots, c_m\}$;
        Sample minibatch of $m$ examples $\{(\boldsymbol{x}_1^s, \boldsymbol{y}_1^s), \dots, (\boldsymbol{x}_m^s, \boldsymbol{y}_m^s)\}$ from $D_{\text{source}}$;
        $k = 1$;
        for $k < K + 1$ do
            Sample minibatch of $m$ examples $\left\{(\boldsymbol{x}_1^k, \boldsymbol{y}_1^k), \dots, (\boldsymbol{x}_m^k, \boldsymbol{y}_m^k)\right\}$ from $D_k$;
            $k = k + 1$;
        end
        Update $\theta_F$ by descending its stochastic gradient:
         $\nabla_{\theta_f} \frac{1}{m} \sum_{i=1}^{m} [\ell\left(\theta_F; G\left(x_i^s, g\left(c_i\right)\right), y_i^s\right) + \lambda \sum_{k=1}^{K} (\ell(\theta_F; (x_i^k, y_i^k)))]$;
        $j += 1$;
    end
    Sample minibatch of $m$ Gaussian noise samples $\{c_1, \dots, c_m\}$;
    Update $g$ by ascending its stochastic gradient: $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \ell\left(\theta_F; G(x_i^s, g(c_i)), y_i^s\right)$;
    $i += 1$;
end
```



(a) digits-DG (b) PACS dataset (c) Office-Home dataset

Figure 3.5: Example images from different DG datasets.

differ in font style, background, and digit color. Some example images are shown in Figure 3.5a. The second benchmark is PACS [57] which contains object images in four domains, photo (P), art painting (A), cartoon (C), and sketch (S). Object images are labeled with seven categories: dog, elephant, giraffe, guitar, house, horse, and person. Some example images are shown in Figure 3.5b. The third benchmark is office-home [58]. It contains four domains: artistic, clipart, product and real world, and 65 classes. Some example images are shown in Figure 3.5c.

Implementation details For evaluation, we followed the leave-one-domain-out principle in [8], which chooses one domain as the test domain while the others are used as training domains. For a fair comparison, the architecture of classification models followed previous works [36]. We also followed [36]'s strategy to set the hyperparameters by assuming we only have limited computational resources. All general hyperparameters, such as the type of optimizer, the learning rate, and the total number of iterations, are consistent for all methods. For the only hyperparameter that is needed in our method, λ, we tuned it with limited trials (less than 5). The experiments in Table 3.1, 3.2, 3.3, 3.4 and 3.5 and Figure 3.6 and 3.7 followed this setting. For the digits-DG, we used a CNN model with four convolution layers. For PACS and office-home, we used ResNet-18 [59] pre-trained on ImageNet [25]. We designed the domain-code NN as a two-layer neural network with linear and softmax layers. We set each element in α as 1.0 in the beginning. We used Adam with a learning rate of 0.001 on the digits-DG task, and we set the batch size as 64 and the total number of iterations as 6,000. For the PACS and office-home tasks, we used Adam with a learning rate of 0.0001. We set the batch size as 16 and the total number of iterations as 10,000. The value of λ is set based on the performance of the training validation set, and the search space is $\lambda \in \{0.5, 1, 1.5\}$. Moreover, [8] showed that hyperparameter selection is important for domain generalization algorithms. We also show the performance of our algorithm with sufficient hyperparameters selection in Section 3.4.5.

Comparison methods We compare GADA with several types of works: (1) the most common baseline **ERM** which trains a model by empirical risk minimization (ERM) with all training datasets; (2) generative model-based methods, **L2A-OT** [36], **DLOW** [22], **MBDG** [41], **Yang2021** [40] and **Wang2021** [39]; (3) adversarial data augmentation, **GUD** [35], which utilizes adversarial data augmentation to solve domain generalization with a single training domain. For a fair comparison, we modify this method by treating a mixture of all training datasets as a single training dataset; and (4) other data augmentation methods, including **CrossGrad** [37], which augments training data with adversarial examples obtained by a domain classifier, **DDAIG** [38], which augments training data without distinguishable domain features, **JiGen** [8], which augments training data with shuffled image patches and **Mix-style** [12], which augments training data with mixed features. We compare classification accuracy on the test domain with these comparison methods, and experimental results are averaged over ten trials with different random seeds. [2]

[2]We get the results of CrossGrad, JiGen, Mix-style, L2A-OT, MBDG, Yang2021, Wang2021 from [12, 40, 41], JiGen, Mix-style, L2A-OT did not report standard deviation and MBDG, Yang2021, and Wang 2021 did not report results on all dataset

3.4.2 Performance on Benchmark dataset

Table 3.1, Table 3.2 and Table 3.3 show the prediction accuracy on the test domain on the digits-DG, PACS, and office-home benchmarks. Overall, the results show that the proposed method outperforms the baseline (ERM) and most competitive methods in most domains on three benchmarks. These results suggest that our method exhibits a more robust generalization capability than most comparison methods.

However, on average, our method performs worse than MBDG and Yang2021 in the PACS task and worse than Yang2021 in the Office-Home task (MBDG, Yang2021, and Wang2021 did not report results on all benchmarks). When we further observe the test accuracy on a specific test domain, we find that our method outperforms Yang2021 on the cartoon and sketch domains for the PACS task and on the clipart domain for the office-home task. Since our proposal is designed to minimize classification loss over the worst-case domain, we care more about the performance of our proposal over difficult domains. By comparing the performance of ERM on each domain, we can judge the difficulty of each domain. ERM should achieve lower accuracy on relatively difficult domains compared with relatively easy domains. Hence, we can think the sketch and clipart are more challenging domains compared with other domains in the PACS and office-home dataset. Therefore, the results in Table 3.2 and Table 3.3 suggest that compared with Yang2021, our proposal is better at handling difficult domains. This is reasonable since our method is designed to improve the model's performance in the most difficult domain, whereas Yang2021 did not have such characteristics. In contrast, our proposal only gives limited domain generalization ability on relatively easy domains, such as the photo domain in the PACS task and the real-world domain in the office-home task. In these relatively simple domains, the performance of our method is worse than most competitors. We speculate that this phenomenon is because we only focus on the most challenging domain. We ignore the model's performance in the relatively simple domain. If we can find a good balance between relatively easy domains and difficult domains, our proposal may lead to better domain generalization ability. We attempt to give a preliminary solution in Section 3.4.5.

Moreover, we can observe that MBDG achieves the highest accuracy in the sketch domain for the PACS task. MBDG also aims to improve the model's domain generalization ability over the worst-case domain. Hence it shows excellent performance in the difficult domain. However, their algorithm made a strong assumption about the classifier. They require that the main classifier can give a consistent prediction for samples in different training domains. However, it is difficult to train such a classifier, especially when training domains have significant differences. In the PACS dataset, the sketch domain has a very different visual appearance from other domains. When MBDG takes the sketch domain as one of the training domains, i.e., when the test domains are cartoon, photo, and art domains, we can observe that MBDG is not as powerful as solving the sketch domain. In contrast, we did not make substantial requirements on the target domain; hence compared with MBDG, our method works on more test domains, and our method outperforms MBDG in the cartoon and art domains. Moreover, we would like to note that the experimental results of MBDG in Table 3.2 cannot compare fairly with other results since we obtained the result of MBDG in [41]. They spent more computational resources to select their hyperparameters. We will make a fair comparison between our proposal and MBDG in Section 3.4.5.

Table 3.1: Experimental result on digits-DG with classification accuracy (%) averaged over 10 runs

	MNIST	SVHN	MNIST-M	SYN	Avg
ERM	96.0 ± 0.2	62.7 ± 0.6	59.6 ± 0.5	78.6 ± 0.5	74.2
CrossGrad [37]	96.7	65.3	61.1	80.2	75.8
JiGen [8]	96.5	63.7	61.4	74.0	73.9
L2A-OT [36]	96.7	68.6	63.9	83.2	78.1
Mix-style [12]	96.5 ± 0.3	64.7 ± 0.7	63.5 ± 0.8	81.2 ± 0.8	76.5
GUD[35]	97.1 ± 0.2	66.7 ± 0.3	63.9 ± 0.3	81.5 ± 0.2	77.3
DDAIG [38]	96.6 ± 0.2	68.6 ± 0.6	64.1 ± 0.4	80.2 ± 0.2	77.6
DLOW [22]	97.6 ± 0.1	68.3 ± 0.5	65.1 ± 0.6	82.2 ± 0.3	78.3
GADA-D	98.1 ± 0.3	**72.2** ± 0.5	**67.1** ± 0.5	**84.9** ± 0.3	**80.6**
GADA-NN	**98.2** ± 0.5	70.2 ± 0.4	66.5 ± 0.5	84.5 ± 0.6	79.9

Table 3.2: Experimental result on PACS with classification accuracy (%) averaged over 10 runs

	Art	Cartoon	Photo	Sketch	Avg
ERM	77.6 ± 0.5	76.4 ± 0.3	96.3 ± 0.1	69.8 ± 0.6	80.0
CrossGrad	79.8	76.8	93.6	66.8	79.3
Jigen	79.4	75.3	96.0	71.6	80.6
L2A-OT	83.3	78.2	96.2	73.6	82.8
Mix-style	84.1 ± 0.4	78.8 ± 0.4	96.3±0.3	75.9 ± 0.9	83.8
GUD	79.5 ± 0.4	77.2 ± 0.5	94.9 ± 0.2	71.1 ± 0.3	80.7
DDAIG	84.2 ± 0.3	78.1 ± 0.6	95.3 ± 0.4	74.7 ± 0.8	83.1
DLOW	80.7 ± 0.6	76.1 ± 0.5	94.1 ± 0.3	76.7 ± 0.8	81.9
Wang2021[39]	81.4	79.6	95.5	80.6	84.3
Yang2021[40]	**85.8** ±0.6	80.7 ± 0.5	**97.3**±0.3	77.3 ± 0.5	85.3
MBDG [41]	80.6 ± 1.1	79.3 ± 0.2	97.0 ± 0.4	**85.2**±0.2	**85.6**
GADA-D	84.3 ± 0.7	**82.7** ± 1.1	96.2 ± 0.4	76.8 ± 0.8	85.1
GADA-NN	84.3 ± 1.2	81.4 ± 0.8	96.0 ± 0.8	78.6 ± 0.6	85.1

When we further compare GADA-D and GADA-NN, we find that GADA-D performs better than GADA-NN on the digits task, and these two methods achieve close performance in the PACS and office-home tasks. Empirically, we find that GADA-D is easier to optimize since the number of parameters of the Dirichlet distribution is much less than the neural network parameters. We speculate that this is why GADA-D achieves better performance than GADA-NN in some cases. We provide more evidence in Section **??**. GADA-NN also demonstrated clear advantages over GADA-D in certain domains, such as the sketch domain in the PACS task. As we discussed in Section 3.3.3, a neural network can represent a more complex distribution than a Dirichlet distribution. Considering that the sketch domain is the most challenging in the PACS task and that images in this domain show a clear difference from the other three domains, we expect the sketch domain to be the most complex in the PACS task. In such a situation, we expect better presentation ability of the neural network to lead to better domain generalization ability than a Dirichlet distribution. Therefore, there is a tradeoff between GADA-D and GADA-NN. When computational resources are limited, we should use GADA-D; otherwise, we should use GADA-NN.

3.4.3 Visualization results

We take an example from the digits-DG task to observe how GADA works. In this example, the model is trained on MNIST, SVHN, and MNIST-M. Since we have three training

Table 3.3: Experimental result on office-home with classification accuracy (%) averaged over 10 runs

	Artistic	Clipart	Product	Real World	Avg
ERM	58.9 ± 0.3	49.4 ± 0.1	74.3 ± 0.1	76.2 ± 0.2	64.7
CrossGrad	58.4	49.4	73.9	75.8	64.4
Jigen	53.0	47.5	71.5	72.8	61.2
L2A-OT	60.6	50.1	**74.8**	**77.0**	65.6
Mix-style	58.7 ± 0.3	53.4 ± 0.2	74.2 ± 0.1	75.9 ± 0.1	65.5
GUD	58.7 ± 0.2	51.6 ± 0.2	74.2 ± 0.3	74.9 ± 0.3	64.9
DDAIG	59.2 ± 0.1	52.3 ± 0.3	74.6 ± 0.3	76.0 ± 0.1	65.5
DLOW	59.2 ± 0.4	51.7 ± 0.4	73.2 ± 0.2	74.4 ± 0.2	64.6
Yang2021	**60.7** $\pm$ 0.8	52.9 ± 0.3	**75.8** $\pm$ 0.1	77.2 ± 0.2	**66.7**
GADA-D	60.1 ± 0.2	**53.7** $\pm$ 0.3	73.3 ± 0.4	76.2 ± 0.4	65.8
GADA-NN	59.7 ± 0.3	53.1 ± 0.3	73.6 ± 0.7	76.7 ± 0.6	65.7

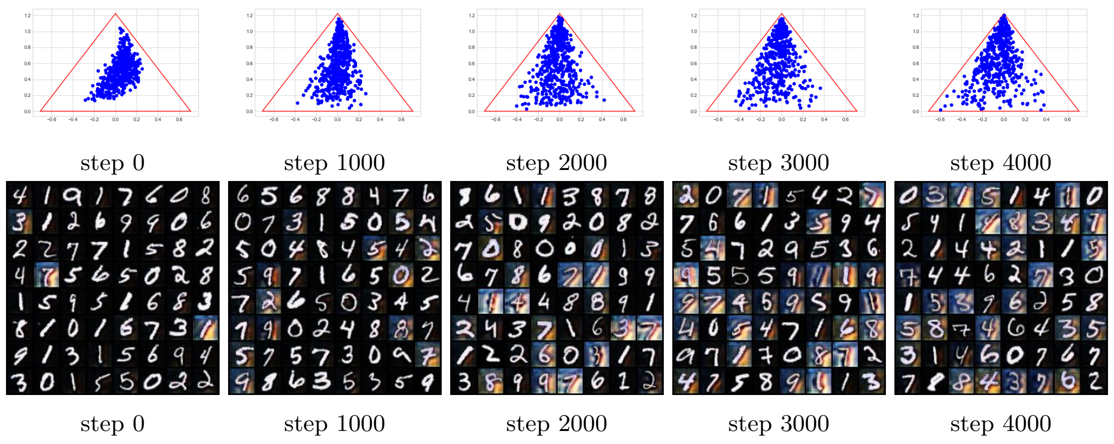


Figure 3.6: Visualization result of GADA-NN. The top represents how domain codes change during training, and the bottom represents how generated images change during training

domains, the domain code is a three-dimensional probability vector; we visualize domain codes on the two-dimensional simplex. We record and visualize 500 generated domain codes on the two-dimensional simplex every 1,000 steps. Each point refers to one domain code in Figure 3.6 and Figure 3.7. We also show synthesized images every 1,000 steps in Figures 3.6 and Figures 3.7. For GADA-NN, we can see that generated domain codes concentrate around the center of the simplex in the beginning. As the learning progresses, the distribution of domain codes gradually shifts away from the lower right corner (MNIST domain). This tendency is also reflected in the generated images; we find that the generated images become increasingly colorful. Intuitively, considering that the MNIST domain is the easiest domain to classify, the observation that the style of the generated samples is less and less like MNIST indicates that our adversarial search of worst-case distribution works successfully. For GADA-D, since we set the initial parameters of the Dirichlet distribution as (1.0,1.0,1.0), the distribution of domain codes is equivalent to a uniform distribution over the two-dimensional simplex. We can see that the generated domain codes shift away from the right corner during the training, and the generated images become more colorful. Here, the parameters of the Dirichlet distribution for each 1,000 steps are (1.0,1.0,1.0), (0.89,1.12,1.05), (0.69,1.19,1.09), (0.41,1.26,1.15) and (0.01,1.45,1.33). This tendency is the

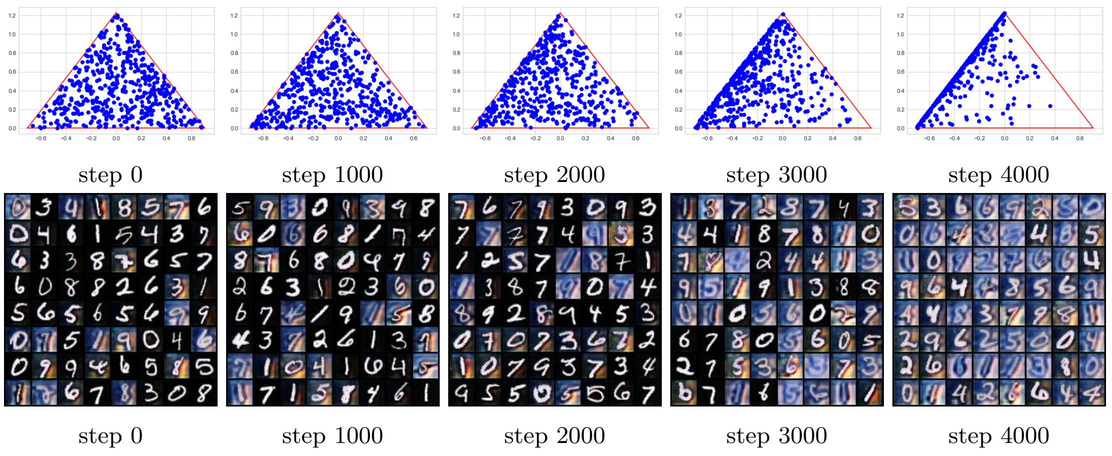


Figure 3.7: Visualization result of GADA-D. The top represents how domain codes change during training, and the bottom represents how generated images change during training.

same as that of GADA-NN. Moreover, we can observe that in steps 3,000 and 4,000, the proportion of color digit images in Figure 3.7 is significantly higher than that in Figure 3.6. This indicates that GADA-D is easier to optimize than GADA-NN.

3.4.4 Ablation studies

To check the importance of each module in GADA, we perform the following ablation studies on the digits-DG and PACS tasks.

Importance of the generative model (vs. pixel-level adversarial attack) We proposed utilizing a generative model to synthesize novel data. Using a generative model allows us to generate semantically different samples from the original training samples. We also have other methods for generating augmented data. For example, [35] utilizes a pixel-level adversarial attack to generate augmented samples. We compare our proposal with this method and the results shown in the third row in Table 3.4 and Table 3.5. The results confirm that using a generative model can achieve better accuracy than using a pixel-level adversarial attack on digits and PACS tasks. These results indicate that augmenting semantically different samples can improve domain generalization ability more than pixel-level adversarial examples, and the use of the generative model is necessary.

Importance of adversarially learned domains (vs. nonadversarial) We propose to train the classification model with adversarially learned domains. To show the importance of training with the worst-case domain, we replace the worst-case distribution of domain codes with a nonadversarial distribution of domain codes. Since the domain code is a probability vector, we compare our proposal with two fixed distributions on the simplex: uniform distribution on the simplex and logit-normal distribution. Experimental results on the PACS (the fourth and fifth rows in Table 3.5) and digits datasets (the fourth and fifth rows in Table 3.4) show that augmenting adversarial learned domains achieves better domain generalization ability than nonadversarial domains.

Importance of domain-code NN (vs. simple combination) Adversarially perturbing the domain code may be considered instead of learning the worst-case distribution of

Table 3.4: Ablation studies on digits with classification accuracy (%) averaged over 10 runs

	MNIST	SVHN	MNISTM	SYN	Avg
GADA-D	98.1 ± 0.3	**72.2** $\pm$ 0.5	**67.1** $\pm$ 0.5	**84.9** $\pm$ 0.3	**80.6**
GADA-NN	**98.2** $\pm$ 0.5	70.2 ± 0.4	66.5 ± 0.5	84.5 ± 0.6	79.9
Adversarial attack	97.1 ± 0.2	66.7 ± 0.3	63.9 ± 0.3	81.5 ± 0.8	77.3
Logistic normal	97.9 ± 0.2	70.5$\pm$0.5	65.7 ± 0.3	82.7 ± 0.2	79.2
Uniform	98.0 ± 0.1	68.3 ± 0.3	66.7 ± 0.5	83.2 ± 0.4	79.1
Non-distributional	97.8 ± 0.3	68.8 ± 0.4	65.2 ± 0.4	83.6 ± 0.6	78.9

Table 3.5: Ablation studies on PACS with classification accuracy (%) averaged over 10 runs

	Art	Cartoon	Photo	Sketch	Avg
GADA-D	**84.3**$\pm$ 0.7	**82.7** $\pm$ 1.1	**96.3** $\pm$0.4	76.8 ± 0.8	**85.1**
GADA-NN	**84.3**$\pm$ 1.2	81.4 ± 0.8	96.0 ± 0.8	**78.6** $\pm$ 0.6	**85.1**
Adversarial attack	79.5 ± 0.4	77.2 ± 0.5	94.9 ± 0.2	71.1 ± 0.3	80.7
Logistic normal	80.8 ± 0.3	78.4 ± 0.7	93.9 ± 0.7	76.4 ± 0.5	82.3
Uniform	81.7 ± 0.6	78.1 ± 0.5	94.5 ± 0.3	76.6 ± 0.8	82.7
Non-distributional	82.2 ± 0.7	79.4 ± 0.9	95.2 ± 0.7	75.8 ± 0.6	83.2

domain codes. Then the training objective becomes:

$$\min_{\theta_F} \sup_{z} \mathbb{E}_{(x,y)\sim D_{\text{source}}} \ell(\theta_F; (G(x,z),y)) + \lambda \sum_{k=1}^{K} \mathbb{E}_{(x^k,y^k)\sim D_k}(\ell(\theta_F; (x^k,y^k))).s.t. \sum_{k=1}^{K} z_k = 1, z_k \geq 0 \quad (3.10)$$

We refer to this objective as a nondistributional method. The nondistributional method can be considered a simple combination of adversarial data augmentation methods and generative model-based data augmentation methods. Compared with the nondistributional method, our method samples domain codes from a learned distribution, which improves the randomness of the domain codes. Hence, the diversity of augmented samples also increases. Experimental results on the PACS (the sixth row in Table 3.5) and digits datasets (the sixth row in Table 3.4) show that using a domain-code NN or learnable Dirichlet distribution to represent the distribution of domain codes performs better than the nondistributional method.

3.4.5 Experiment with sufficient hyperparameters selection

The experimental results in Table 3.1, Table 3.2 and 3.3 suggest that our method is not good at handling relatively easy domains. We speculate that this is because of insufficient hyperparameter tuning. In our training objective, (3.9) and (3.8), we only use one hyperparameter λ to balance the tradeoff between the worst-case domain and the given training domains, and this hyperparameter is selected from a small space, $\lambda \in \{0.5, 1, 1.5\}$, hence, it is possible that in previous experiments, the value of λ is not properly set. Moreover, considering that different training domains have different characteristics, giving different training domains different weights is more appropriate. Hence, we modify our training objective, and (3.9) and (3.8) become:

$$\min_{\theta_F} \sup_{\theta_g} \mathbb{E}_{\substack{c\sim\mathcal{N}(0,1),\\ (x,y)\sim D_{\text{source}}}} [\ell(\theta_F; (G(x,g(c)),y))] + \sum_{k=1}^{K} \lambda_k \mathbb{E}_{(x^k,y^k)\sim D_k}(\ell(\theta_F; (x^k,y^k))). \quad (3.11)$$

Table 3.6: Experimental result of GADA with fine-tuned hyperparameters on digits-DG and PACS with classification accuracy (%) averaged over 10 runs

		MBDG	GADA-D	GADA-NN	GADA-D-F	GADA-NN-F
Digits-DG	MNIST	-	98.1 ±0.3	98.2±0.5	98.4±0.1	**98.5**±0.1
	SVHN	-	72.2 ±0.5	70.2±0.4	**74.1**±0.4	73.9 ± 0.3
	MNISTM	-	67.1 ±0.5	66.5 ±0.5	**67.8**±0.3	67.4±0.4
	SYN	-	84.9±0.3	84.5±0.6	**87.9**±0.4	87.6±0.5
	Mean	-	80.6	79.9	**82.1**	81.8
PACS	Photo	97.0 ± 0.4	96.2 ±0.4	96.0 ±0.8	97.0 ±0.2	**97.2**±0.1
	Art	80.6 ± 1.1	84.3 ±0.7	84.3 ±1.2	84.7 ±0.8	**85.5**±0.5
	Cartoon	79.3 ±0.2	82.7 ±1.1	81.4 ± 0.8	**82.9** ±0.5	82.4 ±0.8
	Sketch	**82.7** ± 1.1	76.8 ± 0.8	78.6 ±0.6	77.6±0.3	81.2 ± 1.1
	Mean	85.6	85.1	85.1	85.5	**86.5**

or

$$\min_{\theta_F} \sup_{\alpha} \mathbb{E}_{\substack{z\sim\mathcal{B}(\alpha),\\ (x,y)\sim D_{\text{source}}}} [\ell(\theta_F; (G(x,g(c)),y))] + \sum_{k=1}^{K} \lambda_k \mathbb{E}_{(x^k,y^k)\sim D_k}(\ell(\theta_F; (x^k,y^k))). \tag{3.12}$$

In (3.11) and (3.12), we introduce more hyperparameters for finding a better balance among all training domains. To find the appropriate value for these hyperparameters, we follow [60]'s strategy. We first randomly set the value of all hyperparameters from a uniform distribution, and then we train a model by our proposal. Finally, we repeat these processes 20 times, evaluate these 20 models on the training evaluation set, and choose hyperparameters that can achieve the best validation accuracy as our final hyperparameters. We denote modified GADA-D and GADA-NN as GADA-D-F and GADA-NN-F (F refers to fine-tuned). We would like to note that MBDG also followed this strategy to tune their hyperparameters.

Table 3.6 reports the performance of GADA-NN-F and GADA-D-F on digits-DG and PACS tasks. The overall result shows that fine-tuning weights for different domains can further improve the classification model's domain generalization ability on both relatively easy domains and relatively hard domains. Moreover, GADA-NN-F achieves higher average accuracy than MBDG on the PACS task, which suggests that under a fair setting, our method exhibits better domain generalization ability than MBDG. We would like to note that since we spent more computational resources on hyperparameters tuning than in previous experiments, the experimental results in Table 3.6 are not comparable to most results in Table 3.1 and Table 3.2 beside MBDG. The current solution requires sufficient computing resources for hyperparameter selection; hence, we think this is only a preliminary solution. How to automatically find a good balance between all training domains is an important question. We leave this as a future direction.

3.5 Further discussion

In this section, we further discuss our proposal in the following aspects:

1. How the number of source domains affects our proposal. More specifically, how will our approach be affected when we have to use fewer source domains to train the DLOW-B

Table 3.7: Experimental result of GADA with multiple DLOW-B models

	MNIST	SVHN	MNIST-M	SVHN
Single DLOW-B	98.2	70.2	66.5	84.5
Multiple DLOW-B	93.5	57.6	55.7	69.5

model?

2. Why our proposal improves the model's generalization ability. Especially when the test domain cannot be obtained by interpolation of the source domain.

3.5.1 How the number of source domains affects our proposal

In the domain generalization setting, we can obtain multiple domains. In our proposal, we utilize these domains to train a generative model to represent the intermediate domains of these training domains. When the number of source domains is large, it is often hard to train a good generative model that can cover all training domains. Hence, in this situation, we have to train the generative models with part of the source domains. To cover all source domains, we attempt to train multiple generative models while each generative model is trained on the part of the source domains. For example, if we have nine domains, then we train three generative models, and each generative model covers three source domains. Then, we can replace the single DLOW-B model in the original proposal with multiple DLOW-B models. We conduct experiments on digital-DG datasets. We train three DLOW-B models on any two source domains, then we can obtain three DLOW-B models. We also modify our proposal with three domain-code NNs, each domain-code NN corresponds to a generative model. We compare this method with our original proposal. The results are shown in Table 3.7.

We can observe that using one DLOW-B model that can cover three domains achieves much better results than using three DLOW-B models trained on two source domains. We believe this result is reasonable. When we train a DLOW-B model on all three training domains, this DLOW-B model can synthesize samples with intermediate styles between all three source domains. However, when we train three DLOW-B models on any two source domains, the synthesized style can only be an interpolation of two source styles. Hence, if we use multiple DLOW-B models trained on part of source domains, the diversity of synthesized images is decreased. Therefore, the performance of the whole proposal drops. However, if the given source domains are too large, it is very difficult to train a DLOW-B model that can cover all source domains. currently, we may only use multiple DLOW-B models to solve such DG problems.

3.5.2 Why our proposal improves the model's generalization ability

We will consider two cases of why our approach works.

1. The target domain can be represented by a combination of all source domains.

2. The target domain can not be represented by a combination of all source domains.

When the target domain can be represented by a combination of all source domains, the DLOW-B model can approximately cover the data-generating distribution of the target domain. In this situation, it is natural that our proposal can improve the model's performance on the target domain.

On the other hand, when the target domain can not be represented by a combination of all source domains, the DLOW-B model can not cover the data-generating distribution of the target domain, and the classification model still needs to handle unseen samples. In this situation, we also observe that our proposal improves the model's domain generalization ability, as shown in Table 3.1, Table 3.2 and Table 3.3. We speculate that the improvement of the domain generalization ability is because our proposals force the model to learn the domain-invariant features.

As shown in [15, 8], the model will learn better domain-invariant features when more data are provided even if the training algorithm is ERM. In our proposal, we continuously feed the model with **difficult** and **diverse** data. The model can not obtain good prediction accuracy on these data due to the difficulty, which means the current features learned by our model are not domain-invariant enough. Augmenting these data will effectively force the model to learn better domain-invariant features. On the other hand, diversity ensures the learned features can keep invariant in more diverse domains. With the above two expectations, our proposal can train a model that can extract good domain-invariant features. Ideally, good domain-invariant features should be shared by all domains. For example, in the digital classification task, the shape is a good domain-invariant feature shared by all possible domains. Hence, this may be the reason why our proposal can improve the model's domain generalization ability even if the target domain can not be represented by a combination of all source domains.

3.6 Conclusion

In this work, we propose GADA, which utilizes a generative model to perform adversarial data augmentation. Using a generative model allows generated samples to be different from original training samples in image style. Moreover, we propose using a domain-code NN or a trainable Dirichlet distribution to represent the distribution of domain codes. During the training, we adversarially optimize the domain-code NN or parameters of the Dirichlet distribution with training the main classifier so that the generative model can generate more difficult samples. In the evaluation, GADA shows superior performance on three benchmark tasks. Further ablation studies and visualization show that each module of GADA is necessary. Moreover, we show that finding a better tradeoff between training domains and the worst-case domain can further improve the model's domain generalization ability.

3.7 Limitation

Our proposal has the following limitations:

1. In the section 3.4.2, we show that our proposal is good at handling difficult test domains, while when the test domain is relatively domain, the performance of our

proposal may be worse than other competitors. This is because our proposal aims to train the model with the data drawn from the worst distribution. In this process, we ignore the model's performance on the relatively simple domains. Although we provide a preliminary solution in Section 3.4.5, this preliminary solution requires large computational resources. We should try to investigate how to find a good balance between difficult and simple domains automatically.

2. Our proposal train a generative model to synthesize augmented data. However, training a good generative model requires relatively large computational resources and carefully hyper-parameter tuning. Hence, if the computational resources are limited, our proposal is a hard solution. Fortunately, most of the domain generalization settings do not limit training resources.

Chapter 4

Zero-shot domain adaptation based on dual-level mix and contrast

This chapter proposes a framework, termed dual mix contrastive learning (DMCL), for zero-shot domain adaptation (ZSDA) tasks. DMCL is a representation learning method that aims to learn domain-invariant features between the source and target domains.

4.1 Introduction

When task-relevant is hard to collect in the target domain, unsupervised domain adaptation methods are hard solutions to alleviate domain shifts. Zero-shot domain adaptation (ZSDA) is designed for this situation. This setting assumes that domain shift is shared with two different tasks, task-of-interest (ToI) and irrelevant task (IrT). The goal of ZSDA is to use task-irrelevant data from both source and target domains to learn the domain shift and then transfer the learned domain shift to the ToI. For example, as Figure 4.1 shows, we assume that digital classification is the task we want to solve, and alphabetical classification is an irrelevant task. We also assume that digital data are hard to obtain and alphabetical data are easy to obtain. We want to utilize digital data from the source domain to build a model that can generalize to the target domain. The ZSDA methods aim to utilize alphabetical data from both source and target domains and digital data from the source domain to build a digital classification model that can generalize to the target domain.

As already discussed in Section 2.6, previous ZSDA works attempt to utilize two types of strategies to solve the ZSDA problem. The generative model-based method and domain-invariant feature learning-based methods. We believe these methods have limitations in the following aspects:

1. In the generative model-based method, we need to train a generative model. Training a good generative model requires significant computational resources and careful tuning of the hyperparameters. If the trained generative model is not good enough, such methods' performance will decrease significantly. Moreover, when data from more domains or tasks are available, the generative model-based methods are hardly a solution since it is difficult for the generative model to handle data from many domains.

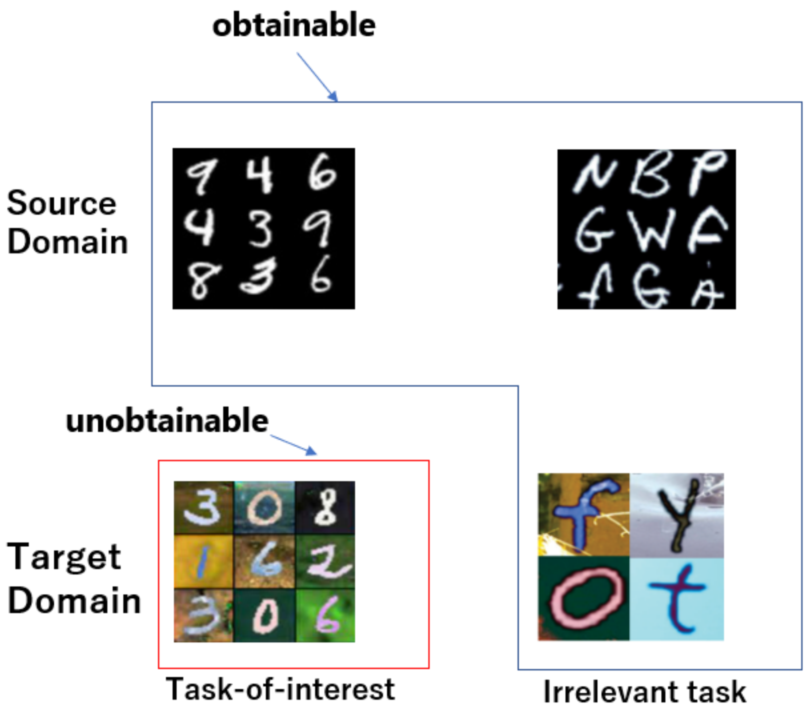


Figure 4.1: An example of zero-shot domain adaptation, samples in blue box are available during training, samples in red box are what we want to classify

2. The domain-invariant feature learning-based methods [47] can be divided into two stages. In the first stage, this method extends domain adversarial training [5] to learn domain-invariant features. In the ZSDA setting, since we do not have samples for ToI in the target domain, the domain-invariant features are mainly learned with IrT data. Hence, in this stage, learned features might not generalize to the ToI approximately. To alleviate this issue, in the second stage, [47] design an attention-based [48] module to fine-tune the model learned in the first stage so that bias caused by the ZSDA setting can be alleviated. In this sense, [47] first learns biased features and then removes the bias in the second stage. Considering that not all bias can be completely eliminated in the second stage, it is important to learn as little bias as possible in the first stage.

In this paper, we present a novel domain-invariant representation learning method, termed **d**ual **m**ixup **c**ontrastive **l**earning (DMCL). Our method aims to learn unbiased domain-invariant features to solve the ZSDA task without relying on the generative model.

First, although our method relies on virtual data to act as task-relevant data from the target domain, these virtual data are not generated by a generative model. We extend mixup technique [16] to a dual-level mixup, which applies mixup on both category-level and domain-level, to synthesize virtual data to act as the role of target task-relevant data. Therefore, our method makes the virtual data we rely on easily accessible since mixup technique does not require heavy computational resources and careful hyperparameter tuning. Compared with previous mixup-based data adaptation methods [61, 20], we combine dual-level mixup with contrastive learning-based objectives for extending mixup to the

ZSDA setting. Contrastive learning objectives allow us to carefully design the behavior of feature extractors with data from multiple domains and tasks. We show that contrastive learning-based objectives lead to better domain adaptation ability in the ZSDA setting than distance-based objectives.

Moreover, our proposal is designed to learn unbiased domain-level and task-level features directly by extending domain adversarial training to the ZSDA setting. Hence, our method can be considered as an improvement of [47]'s first stage, and our method can be combined with the second stage of [47]. We have made the following two efforts to extend the domain adversarial training to the ZSDA setting. First, the domain adversarial training largely depends on a domain classifier which is used for distinguishing the domain of a sample. In the ZSDA setting, it is hard to obtain ToI data from the target domain, hence trained domain classifier can not generalize to ToI. To overcome this difficulty, we aim to provide virtual data to act as task-relevant data from the target domain. Then, we train the domain classifier by augmenting these virtual data so that the resulting domain classifier can generalize to ToI. Second, we propose two contrastive learning objectives for encouraging feature disentanglement between domain-level features and task-level features. Then, domain-level features will be used in domain adversarial training so that the main feature extractor will learn domain-invariant features. Task-level features will be used in classification tasks without the influence of the domain.

Our main contribution can be summarized as follows:

1. We propose to use a dual mixup, which generates samples by randomly interpolating two domains and two tasks. Dual mixup allows to generate various samples belonging to intermediate tasks and domains without the training of generative models.

2. We propose an extension of domain adversarial training to obtain domain-invariant features that can generalize over ToI by further forcing the model to distinguish the dual mixup samples.

3. Samples generated with dual mixup have intermediate class labels and domain labels that interpolate the two tasks and domains. To exploit the diversity of dual mixup samples to enhance domain invariance and reduce task biasedness of the features, we introduce a novel dual-level contrastive learning method that contrasts pairs of samples at two levels: task and domain.

4. We experimentally demonstrate that our proposal achieves good performance among several competitors with several datasets. Also, additional experiments verify our proposal is able to learn domain-invariant features for ToI data.

4.2 Preliminary

4.2.1 Domain adversarial training

This work attempt to extend domain adversarial training in the ZSDA setting. Hence, we first introduce the domain adversarial training [5].

Domain adversarial training is a classical unsupervised domain adaptation algorithm for learning domain-invariant features. Let $\mathcal{X}_s$ and $\mathcal{Y}_s$ be the input space and label space of the

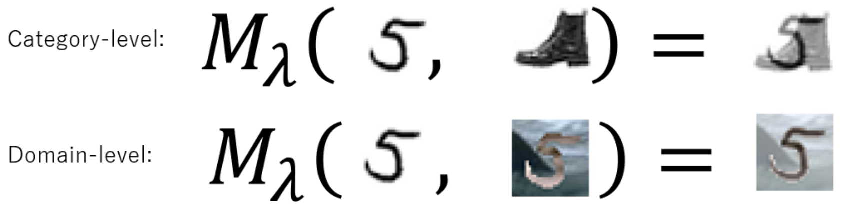


Figure 4.2: An illustration of mixup

source domain, and $\mathcal{X}_t$ be the input space of target domain. Let $G : \mathcal{X} \mapsto \mathbb{R}^m$ be the feature extractor, $C : \mathbb{R}^m \mapsto \mathcal{Y}$ be the category classifiers. Domain adversarial training introduces an additional domain classifier $D : \mathbb{R}^m \mapsto [0, 1]$ for distinguishing domain information of input samples (samples from source domain with domain label 0, samples from target domain with domain label 1). The domain adversarial training can be formulated as follows:

$$\min_{G,C} \max_{D} \mathcal{L}_c(G, C) + \lambda \mathcal{L}_d(G, D)$$

$$\mathcal{L}_c(G, C) = \mathbb{E}_{(\mathbf{x}^s, y^s) \sim D^s} \ell \left(C \left(G \left(\mathbf{x}^s \right) \right), y^s \right)$$

$$\mathcal{L}_d(G, D) = \mathbb{E}_{\mathbf{x}^s \sim D^s} \log D \left(G \left(\mathbf{x}^s \right) \right) + \mathbb{E}_{\mathbf{x}^t \sim D^t} \log \left(1 - D \left(G \left(\mathbf{x}^t \right) \right) \right) \tag{4.1}$$

where ℓ is the cross-entropy loss, and λ is a trade-off hyper-parameter. By optimizing parameters of D, features obtained with G are expected to be domain invariant. L_c is a classification loss for label classifier, L_d is used to train a domain classifier.

4.2.2 Mixup

Mixup is an important building block in our framework. Hence we introduce mixup in this section. Category-level mixup [16] performs data augmentation by constructing virtual samples with convex combinations of pair of sample-label pairs $(\mathbf{x}_i, y_i)$ and $(\mathbf{x}_j, y_j)$. Category-level mixup can be formulated as follows:

$$\begin{aligned} \widetilde{\mathbf{x}} &= \mathcal{M}_\lambda \left(\mathbf{x}_i, \mathbf{x}_j \right) = \lambda \mathbf{x}_i + (1 - \lambda) \mathbf{x}_j \\ \widetilde{y} &= \mathcal{M}_\lambda \left(y_i, y_j \right) = \lambda y_i + (1 - \lambda) y_j \end{aligned}$$

Where $\lambda \sim \text{Beta}(\alpha, \alpha)$, for $\alpha \in (0, \infty)$.

Inspired by [16], [20] proposed domain-level mixup for training a good domain classifier. They assume the corresponding domain information also be mixed in equal proportions. Then, the domain label of the synthesized samples is also a convex combination of the corresponding domain label of $\mathbf{x}_i$ and $\mathbf{x}_j$.

$$\widetilde{d} = \mathcal{M}_\lambda \left(d_i, d_j \right) = \lambda d_i + (1 - \lambda) d_j$$

An example of category-level mixup and domain-level mixup are shown in Figure 4.2.

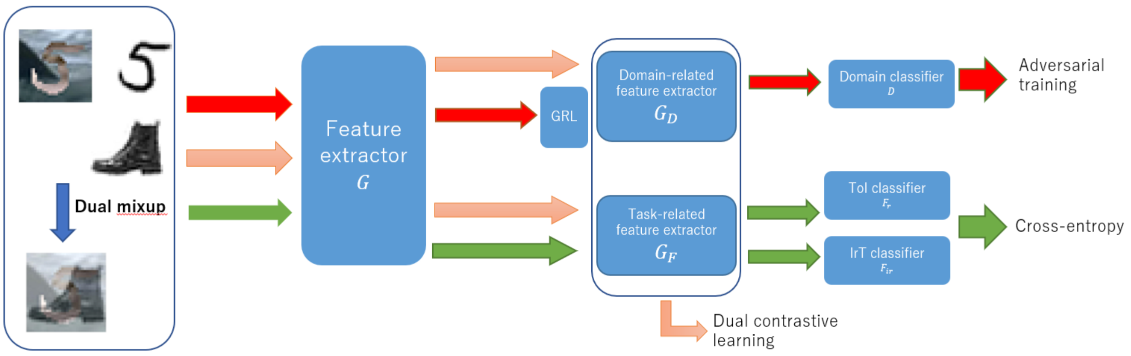


Figure 4.3: An illustration of our proposal DMCL. Here GRL refers to a gradient reversal layer. The red arrow indicates the adversarial learning procedure, the green arrow indicates a standard training procedure, and the pink arrow indicates the contrastive learning objectives.

4.3 Proposal

4.3.1 Overview

In this section, we propose a learning algorithm, termed **d**ual **m**ixup **c**ontrastive **l**earning (DMCL), for the zero-shot domain adaptation. As discussed above, to solve the ZSDA task, DMCL aims to learn domain-invariant features that generalize over ToI task. An overview of our proposal is shown in Figure 4.3.

First, we extend mixup to synthesize intermediate samples between domains and tasks to fill the absence of target ToI data. Then, we extend domain adversarial training with intermediate samples to learn domain-invariant features. As shown in Figure 4.3, our model structure is an extension of domain adversarial neural network with a domain classifier $D \circ G_D$, and two task classifiers, $F_r \circ G_F$ for ToI, $F_{ir} \circ G_F$ for IrT. With adversarial learning (red arrow in Figure 4.3), the feature extractor G is expected to extract domain-invariant features with maintaining discriminative capability in both ToI and IrT. Here, the domain invariance of the features is induced by the gradient reversal layer (GRL) which reverses the gradient by multiplying a negative value during the backpropagation. Also, the discriminative capability of the features is due to training by cross-entropy loss (green arrow in Figure 4.3).

As we demonstrate with ablation studies later, domain-invariant features learned by domain adversarial training are insufficient to generalize to ToI. Since we can only have IrT data from both domains, the resulting features are biased toward the IrT task. To deal with this task bias, we design dual-level contrastive learning objectives (pink arrow in Figure 4.3). With the first contrastive learning objective, G_F is encouraged to be more sensitive to differences in tasks and class labels while caring less about differences in domains. This further enhances the domain invariance of the features input to F_r and F_{ir}. Similarly, with the second contrastive learning objective, G_D focuses on domain-related information more while ignoring task-related information. Hence, the performance of the domain classifier will be further improved.

Moreover, since G_D only learns domain-related features, when G is adversarially trained with gradient information from G_D, only domain-related features are eliminated, and the task-related features which are helpful in classification tasks are unaffected. On the other hand, with adversarial training, G tends to learn domain-invariant features, i.e., the input of G_F becomes domain-invariant. This leads to a better domain invariance of outputs of G_F. Therefore, by training features with the contrastive learning objective and the domain adversarial training objective alternately, domain-invariance and task-unbiasedness can enhance each other.

4.3.2 Dual Mixup for intermediate samples

In this work, we provide intermediate data from different domains and tasks to act as target ToI data with low computation costs. This strategy has third merits. First, these intermediate data contain information from both source and target domains, also from both ToI and IrT. A model trained on these data should be able to learn features that work on different domains and tasks. Second, we can generate intermediate data without using target ToI data since intermediate data between source ToI data and target IrT already contain information from two domains and tasks. Third, we synthesize intermediate samples by mixup. Mixup only requires applying a convex combination on different samples; hence this method requires low computation resources.

To synthesize intermediate data between both tasks and domains, we extend the mixup [16] technique. Formally, for x_i in D_s^r and x_j in $D^{ir} = D_s^{ir} \cup D_t^{ir}$, we synthesize virtual data by $\widetilde{x} = \mathcal{M}_\lambda\left(x_i, x_j\right)$.

Then, unlike single-level mixup, we mix both the category label and the domain label for $\widetilde{x}$ by:

$$\widetilde{y} = \mathcal{M}_\lambda\left(y_i, y_j\right), \quad \widetilde{d} = \mathcal{M}_\lambda\left(d_i, d_j\right)$$

where $\lambda \sim \mathrm{Beta}(\alpha, \alpha)$, for $\alpha \in (0, \infty)$.

4.3.3 Domain adversarial training with dual mixup

Using data augmentation with intermediate data synthesized by dual mixup, we extend the domain adversarial training method for learning domain-invariant features. During this procedure, we force the domain classifier and label classifier to distinguish samples generated by mixing samples in different domains and tasks with random proportions. Through this training, the feature extractor is trained to be able to handle data from different domains and tasks.

Specifically, we define $C_d = D \circ G_D \circ G$, which means $C_d(x) = D(G_D(G(x)))$. Similarly, we define $C_r = F_r \circ G_F \circ G$ and $C_{ir} = F_{ir} \circ G_F \circ G$. Then, the domain adversarial training with dual mixup samples is shown in the following:

$$\min_{\substack{G, G_F \\ F_r, F_{ir}}} \max_{G_D, D} \mathcal{L}_{adv} = \mathcal{L}_d(C_d) + \mathcal{L}_{md}(C_d) + \mathcal{L}_f(C_r, C_{ir}) + \mathcal{L}_{mf}(C_r, C_{ir}). \quad (4.2)$$

where

$$\mathcal{L}_d(C_d) = \mathbb{E}_{\mathbf{x}_s \sim D_s} \log\left(1 - C_d(\mathbf{x}_s)\right) + \mathbb{E}_{\mathbf{x}_t \sim D_t^{ir}} \log C_d(\mathbf{x}_t),$$

$$\mathcal{L}_{md}(C_d) = \mathbb{E}_{\substack{\mathbf{x}_i \sim D_s \\ \mathbf{x}_j \sim D_t^{ir} \\ \lambda \sim \mathrm{Beta}(\alpha,\alpha)}} \lambda \log(1 - C_d(\widetilde{x})) + (1 - \lambda) \log C_d(\widetilde{x}).$$

$$\mathcal{L}_f(C_r, C_{ir}) = \mathbb{E}_{(\mathbf{x},y)\sim D_s^r}\ell(C_r(\mathbf{x}), y) + \mathbb{E}_{(\mathbf{x},y)\sim D^{ir}}\ell(C_{ir}(\mathbf{x}), y)$$

$$\mathcal{L}_{mf}(C_r, C_{ir}) = \mathbb{E}_{\substack{\lambda\sim\text{Beta}(\alpha,\alpha)\\ \mathbf{x}_i\sim D_s^r, \mathbf{x}_j\sim D^{ir}}} \lambda\ell(C_r(\widetilde{x}), y_i) + (1-\lambda)\ell(C_{ir}(\widetilde{x}), y_{ir}).$$

Here, $D_s = D_s^{ir} \cup D_s^r$, $D^{ir} = D_s^{ir} \cup D_t^{ir}$, $\widetilde{x} = \mathcal{M}_\lambda(x_i, x_j)$, ℓ is the cross-entropy loss, and α is hyper-parameter.

In the above training objective, $\mathcal{L}_d$ and $\mathcal{L}_{md}$ force G to learn domain-invariant features and force D and G_D to give high domain classification accuracy. $\mathcal{L}_f$ and $\mathcal{L}_{mf}$ are the classification error of ToI, IrT data, and of their mixup, respectively; this forces C_r and C_{ir} become to be able to classify ToI and IrT data, respectively.

4.3.4 Dual contrastive learning for disentanglement

We then design two contrastive learning objectives to enhance domain invariance and reduce task biasedness in the features. We assume that the features extracted from an image can be divided into two types: domain-related features and task-related features. We expect the domain-related features to contain information that allows identifying the domain while it is insensitive to changes in the task and category. In contrast, we expect the task-related features to contain information that allows identifying category labels while it is insensitive to changes in the domain. As discussed in section 4.1, disentangling task-related features from domain-related features helps the model in classifying target ToI data.

We introduce two feature extractors, G_F and G_D, to realize the feature disentanglement. G_F is used to extract task-related features, G_D is used to extract domain-related features. To enforce feature disentanglement, our key intuition is that when two samples from different domains but with the same category are fed into G_F, their corresponding outputs should be very similar since these two samples are only different at the domain level. Likewise, when two samples from different categories but with the same domain are fed into G_D, their corresponding outputs should be the same.

Considering that we have no way of knowing how domain-related and task-related features are mixed together in target ToI data, we need our model to be able to have the generalized feature disentangled capability. This means our model should be able to achieve feature disentanglement for data containing different mixes of domain-related and task-related features. On the other hand, by varying λ, mixup is able to generate intermediate data containing different proportions of domain information and task information. Hence, we utilize intermediate data to realize our intuition for obtaining generalized feature disentangled ability. Figure 4.4 explains a high-level concept of the proposed mixup procedure.

Formally, let us consider three mini-batches of K samples $X_s^r, X_s^{ir}, X_t^{ir}$ from D_s^r, D_s^{ir}, D_t^{ir}, We first apply mixup on any pair of min-batches of samples from the three mini-batches with the same λ for obtaining intermediate samples.

$$\mathbf{A} = \left\{\mathcal{M}_\lambda\left(\mathbf{X}_s^{r_i}, \mathbf{X}_s^{ir_i}\right)\right\}_{i=1}^K, \mathbf{B} = \left\{\mathcal{M}_\lambda\left(\mathbf{X}_s^{r_i}, \mathbf{X}_t^{ir_i}\right)\right\}_{i=1}^K$$

$$\mathbf{C} = \left\{\mathcal{M}_\lambda\left(\mathbf{X}_s^{ir_i}, \mathbf{X}_t^{ir_i}\right)\right\}_{i=1}^K$$

Then, we get three mini-batches of mixup samples $\mathbf{A}, \mathbf{B}, \mathbf{C}$. We assume that after applying mixup, the corresponding category information and the domain information will also

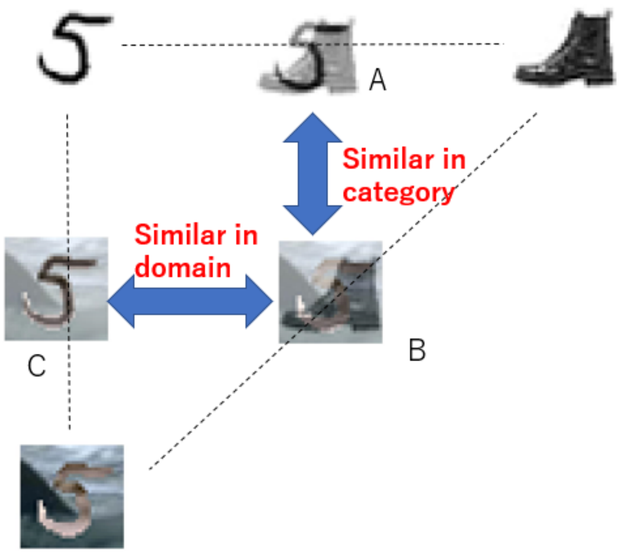


Figure 4.4: An illustration of dual contrastive learning, here the dashed line indicates apply mixup on two samples.

be mixed in equal proportions. Then, $\mathbf{A}_i$ and $\mathbf{B}_i$ contain the same category information but different domain information, and $\mathbf{B}_i$ and $\mathbf{C}_i$ contain the same domain information but different category information.

Considering that the core idea of our intuition is similar to the goal of contrastive learning, our intuition can be naturally realized by contrastive learning. We take G_D as an example to explain our detailed approach. For G_D, our key idea suggests that $G_D(G(\mathbf{B}_i))$ and $G_D(G(\mathbf{C}_i))$ should become as similar as possible, which means $(\mathbf{B}_i, \mathbf{C}_i)$ should be treated as a positive pair in contrastive learning; we treat the other $2(K-1)$ augmented samples within a minibatch and $\mathbf{B}_i$ as $2(K-1)$ negative pairs, like did in [62].

Then, given positive and negative pairs, we utilize the normalized temperature-scaled cross entropy (NT-Xent) loss $\mathcal{L}_{con_d}$ to train G_D:

$$\mathcal{L}_{con_d}(G_D) = -\log \frac{\exp\left(\operatorname{sim}\left(\mathbf{zd}_{\mathbf{B}}^{i}, \mathbf{zd}_{\mathbf{C}}^{i}\right)/\tau\right)}{\sum_{\substack{j=1, j\neq i \\ T\in\{\mathbf{B},\mathbf{C}\}}}^{K} \exp\left(\operatorname{sim}\left(\mathbf{zd}_{\mathbf{B}}^{i}, \mathbf{zd}_{T}^{j}\right)/\tau\right)} \tag{4.3}$$

where, $\operatorname{sim}(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^{\top}\boldsymbol{v}/\|\boldsymbol{u}\|\|\boldsymbol{v}\|$ denote the dot product between l_2 normalized $\boldsymbol{u}$ and $\boldsymbol{v}$. $\mathbf{zd}_{\mathbf{B}}^{i} = G_D(G(\mathbf{B}_i))$, $\mathbf{zd}_{\mathbf{C}}^{i} = G_D(G(\mathbf{C}_i))$, τ is the temperature parameter.

Likewise, G_F treats $(\mathbf{A}_i, \mathbf{B}_i)$ as positive pair. G_F is trained by NT-Xent loss L_{con_f}:

$$\mathcal{L}_{con_f}(G_F) = -\log \frac{\exp\left(\operatorname{sim}\left(\mathbf{zf}_{\mathbf{A}}^{i}, \mathbf{zf}_{\mathbf{B}}^{i}\right)/\tau\right)}{\sum_{\substack{j=1, j\neq i \\ T\in\{\mathbf{A},\mathbf{B}\}}}^{K} \exp\left(\operatorname{sim}\left(\mathbf{zf}_{\mathbf{A}}^{i}, \mathbf{zf}_{T}^{j}\right)/\tau\right)} \tag{4.4}$$

where, $\mathbf{zf}_{\mathbf{A}}^{i} = G_F(G(\mathbf{A}_i))$, $\mathbf{zf}_{\mathbf{B}}^{i} = G_F(G(\mathbf{B}_i))$.

The entire training procedure is performed as follows. We first optimize equation (4.2), then optimize equation (4.3) and (4.4) with shared intermediate samples. We repeat the above two steps alternately.

4.3.5 Adaptive temperature for contrastive learning

There is a limitation with the above contrastive learning objectives. We take L_{con_d} as an example to explain this limitation. In this loss, we sample λ from a beta distribution. When λ is close to 0, $\mathbf{B}_i$ and $\mathbf{C}_i$ are all similar to $\mathbf{X}_t^{ir_i}$. In this situation, $\mathbf{zd}_\mathbf{B}^i$ and $\mathbf{zd}_\mathbf{C}^i$ will be naturally close to each other. Hence, focusing on the positive pair is meaningless. In contrast, when λ is close to 1, the difference between $\mathbf{B}_i$ and $\mathbf{C}_i$ is large. In this situation, it is difficult to make $\mathbf{zd}_\mathbf{B}^i$ and $\mathbf{zd}_\mathbf{C}^i$ close to each other. We should pay more attention to the positive pair. However, the equation 4.3 equally treats the above two situations. The training objective does not take into account the change in focus.

To alleviate this limitation, we propose introducing an adaptive temperature to control the focus of the training loss. In the original NT-Xent loss, the temperature is usually a constant hyper-parameter. [63] found when the temperature τ becomes smaller, the contrastive loss tends to make the positive pairs closer to each other. This means τ should decrease when we want to focus on positive pair. Inspired by this, we propose an adaptive temperature to change the focus of NT-Xent loss during the training automatically. Based on the above discussion, the adaptive temperature should decrease as λ increases. Hence, we use a function $k(\lambda)$ to replace τ. $k(\lambda) = max(\eta, a\lambda + b)$, where η is a small positive value for preventing temperature smaller than 0, a and b are two hyper-parameters, and $a < 0$. With this adaptive temperature function, we modify the equation 4.3 as the following. In our proposal, we set $a = -10$, $b = 10$, and $\eta = 0.1$.

$$\mathcal{L}_{ada_d}(G_D) = -\log \frac{\exp\left(\text{sim}\left(\mathbf{zd}_\mathbf{B}^i, \mathbf{zd}_\mathbf{C}^i\right)/k(\lambda)\right)}{\sum_{\substack{j=1, j\neq i \\ T\in\{\mathbf{B},\mathbf{C}\}}}^{K} \exp\left(\text{sim}\left(\mathbf{zd}_\mathbf{B}^i, \mathbf{zd}_T^j\right)/k(\lambda)\right)} \tag{4.5}$$

Empirically, this new contrastive learning objective can improve the model's domain adaptation ability with our proposal. Hence, we use the adaptive temperature in our proposal.

4.4 Experiments

4.4.1 Experimental results on two benchmarks

Datasets We evaluate our proposal on two benchmarks. The first is X-NIST, which consists of four domains and four classification tasks. X-NIST is based on four datasets, including MNIST (task M) [64], Fashion-MNIST (task F) [65], EMNIST (task E) [66], NIST (task N) [67]. Images in these datasets are all in the gray-scale domain (domain G). To test the domain adaptation performance, we create the color (domain C), edge (domain E), and negative domains (domain N). The color domain is synthesized by using [5]'s method, blending the samples with randomly selected patches from the BSDS500 dataset [68]. The edge domain is created by applying the canny edge detector, and the negative domain is obtained by subtracting the original pixel value from 255. When conducting experiments, we choose two of the four tasks as ToI and IrT, but we do not consider the NIST and EMNIST combinations since their label spaces are not completely different.

The second benchmark is Office-Home datasets. It consists of images from four different domains: Artistic images (Ar), Clip images (Cl), Product images (Pr), and Real-world images (Rw). This dataset contains images from 65 object categories for each domain.

Table 4.1: Experimental result on X-NIST with classification accuracy (%) averaged over 10 runs

Domains	Methods	ToI	MNIST(D_M)			FashionMNIST(D_F)			NIST(D_N)		EMNIST(D_E)		Average
		IrT	D_F	D_N	D_E	D_M	D_N	D_E	D_M	D_F	D_M	D_F	
G→ C	ZDDA		73.2	92.0	94.8	51.6	43.9	65.3	34.3	21.9	71.2	47.0	59.5
	CoCoGAN		78.1	92.4	**95.6**	56.8	56.7	**66.8**	41.0	44.9	75.0	54.8	66.2
	Wang2020		**81.2**	**93.3**	95.0	**57.4**	58.7	62.0	44.6	45.5	72.4	58.9	**66.9**
	DF-ZSDA		68.7	77.0	86.3	40.2	42.3	42.4	45.7	31.3	**83.0**	**66.8**	58.4
	Ours		76.6	89.2	80.9	45.3	**59.5**	57.0	**50.2**	**59.2**	78.1	60.9	65.7
G→ E	ZDDA		72.5	91.5	93.2	54.1	54.0	65.8	42.3	28.4	73.6	50.7	62.6
	CoCoGAN		79.6	94.9	95.4	61.5	57.5	71.0	48.0	36.3	77.9	58.6	68.1
	Wang2020		81.4	93.5	**96.3**	**63.2**	58.7	**72.4**	49.9	38.6	78.2	61.1	69.3
	DF-ZSDA		79.5	**95.5**	93.5	33.4	30.7	35.8	**53.4**	**47.0**	**85.5**	**74.4**	62.9
	Ours		**87.0**	91.5	93.1	59.9	**63.8**	64.7	48.3	44.2	78.5	71.5	**70.3**
G→ N	ZDDA		77.9	82.4	90.5	61.4	47.4	62.7	37.8	38.7	76.2	53.4	62.8
	CoCoGAN		80.3	87.5	93.1	66.0	52.2	69.3	45.7	53.8	81.1	56.5	68.6
	Wang2020		-	-	-	-	-	-	-	-	-	-	-
	DF-ZSDA		59.7	81.0	90.6	68.7	64.3	77.6	58.7	59.0	77.7	64.0	70.1
	Ours		**94.6**	**94.2**	**97.6**	**69.8**	**68.7**	**78.9**	**62.7**	**64.9**	**86.2**	**86.4**	**80.4**
C → G	ZDDA		67.4	85.7	87.6	55.1	49.2	59.5	39.6	23.7	75.5	52.0	59.5
	CoCoGAN		73.2	89.6	94.7	61.1	50.7	70.2	47.5	57.7	80.2	67.4	69.2
	Wang2020		73.7	91.0	93.4	62.4	53.5	71.5	50.6	58.1	83.5	70.9	70.9
	DF-ZSDA		**98.1**	**99.1**	**99.1**	**88.0**	**89.1**	**89.5**	**69.0**	**69.1**	**91.3**	**92.1**	**88.4**
	Ours		92.1	90.3	92.8	86.2	76.2	74.9	65.9	62.8	89.4	75.9	80.7
N→ G	ZDDA		78.5	90.7	87.6	56.6	57.1	67.1	34.1	39.5	67.7	45.5	62.4
	CoCoGAN		80.1	92.8	93.6	63.4	61.0	72.8	47.0	43.9	78.8	58.4	69.2
	Wang2020		82.6	**94.6**	95.8	67.0	68.2	77.9	51.1	44.2	79.7	62.2	72.3
	DF-ZSDA		64.1	68.7	89.5	58.7	57.2	30.3	58.4	51.0	73.4	56.6	60.8
	Ours		**95.8**	92.4	**97.9**	**75.0**	**73.9**	**78.1**	**64.6**	**57.2**	**88.3**	**87.5**	**81.1**

When conducting experiments, we used 10 random categories from 65 categories as ToI and the rest as the IrT.

Implementation deatails In all experiments, the classifier F_r, F_{ir}, and D were implemented with one fully connected layer. In the X-NIST benchmark, two feature extractors, G_D and G_F, were implemented with three convolutional layers. G was implemented with three convolutional layers. We set the batch size as 64 and the total number of iterations as 7,000. In the Office-Home dataset, we utilized ResNet-50 pre-trained on ImageNet. G was implemented with stages 0 to 3 of ResNet-50. Stage 4 of ResNet-50 was copied into two parts: G_D and G_F. We set the batch size as 32 and the total number of iterations as 15,000. For all tasks, we used Adam with a learning rate of 0.0002. In half of the full training iterations, the learning rate was decayed by 0.1. More details are shown in the supplementary.

Comparison methods We compare our proposal with two types of works: (1) the generative model-based methods, ZDDA [9], CoCoGAN [45], and Wang2020 [13]. (2) Feature disentanglement-based method, DF-ZSDA [47] [1]. For all competitors, the model was trained on source ToI, source IrT, and target IrT. Then the model was tested on the target ToI data.

Results Table 4.1 shows the prediction accuracy of the target ToI data for over 10 task combinations on the X-NIST. Overall, these results show that our proposal exhibits a good domain adaptation ability in the ZSDA setting; our proposal achieves the best average

[1]We used the official code provided by the author of DF-ZSDA to reproduce DF-ZSDA's experimental results on X-NIST. The reproduced results are worse than the results reported in their paper. The experimental results on office-home datasets are drawn from their paper because the provided code is not prepared for office-home.

Table 4.2: Experimental result on Office-Home with classification accuracy (%) averaged over 10 runs

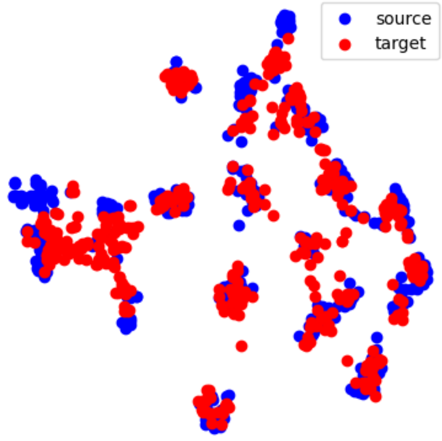
Source	Pr			Rw			Ar			Cl		
Target	Ar	Cl	Rw	Ar	Cl	Pr	Cl	Pr	Rw	Ar	Pr	Rw
CoCoGAN	57.6	53.4	71.7	69.2	51.3	65.8	62.3	69.5	74.5	66.7	74.0	66.4
Wang2020	**70.3**	60.8	74.8	72.2	61.4	72.2	62.7	71.9	76.3	**72.6**	**75.1**	73.9
DF-ZSDA	64.4	**69.2**	**82.0**	**77.9**	**76.2**	**88.5**	71.0	76.5	**85.1**	62.1	68.7	**75.1**
Ours	67.5	65.1	78.9	74.3	69.0	75.8	**72.1**	**76.7**	83.8	69.8	73.0	71.5

accuracy on X-NIST. In particular, our proposal achieves the best performance when domain shifts happen between the gray-scale domain and the negative domain (the third and fifth columns in Table 4.1). This is because our proposal relies on the mixup technique, which assumes that intermediate domains can be obtained by linear interpolating source and target domains. Gray-scale and negative domains satisfy this assumption very well, so our proposal performs particularly well. When domain shifts are $C \rightarrow G$, our proposal achieves the second-best result, only worse than DF-ZSDA. When domain shifts are $G \rightarrow C$ and $G \rightarrow E$ (the first and second columns in Table 4.1), the performance gap between our proposal and the best method, Wang2020, is within four percent. This gap is acceptable when considering that our proposal does not require heavy computation resources to train a generative model.

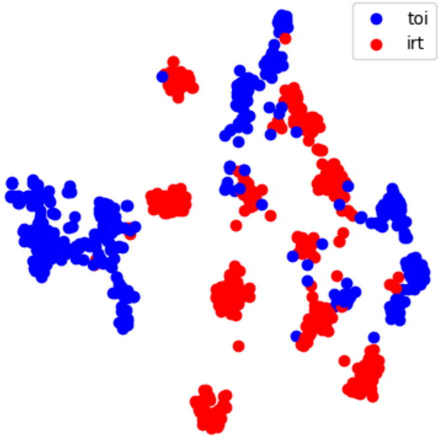
Table 4.2 shows the prediction accuracy of the target ToI data on the office-home. Although the domain shift in this benchmark is more complex than X-NIST, our proposal still shows good performance to overcome domain shift. Compared with generative model-based methods, our proposal achieves competitive or better results with fewer training resources. Compared with DF-ZSDA, our method achieves close results in certain situations. However, when the source domain is Rw and the target domains are Cl or Pr (the sixth and seventh columns in Table 4.2), there is a clear gap between our approach and DF-ZSDA. We suspect that the reason for not performing well is that the gap between domains is complex. This complexity weakens the effect of mixup and therefore affects the performance of our proposal.

4.4.2 Visualizations

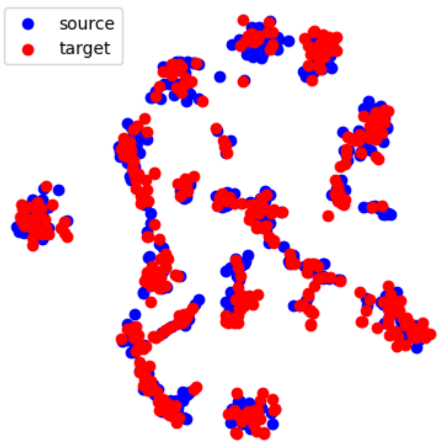
In this work, we are ultimately concerned with the features used to classify ToI data, which are outputs of G_F. We expect G_F can learn domain-invariant features and is able to separate features belonging to different tasks. To verify whether our proposal works as expected, we visualize the feature space of G_F on X-NIST. More specifically, we randomly select 200 samples from source IrT, source ToI, target IrT, and target ToI datasets, respectively. Then, we utilize t-SNE to visualize features extracted by G_F on a two-dimensional space. In Figure 4.5, we use two ways to colorize feature points. First, we colorize points by their domain label. We can find the feature distributions of source and target domains are very similar. This indicates G_F can extract domain-invariant features. Then, we colorized points by their task. We can find that the features of ToI and IrT can be separated, which indicates our contrastive learning objective works well. With the above observation, we concluded that our proposal works as expected. Visualizations on more datasets are shown in the supplementary due to space limits.



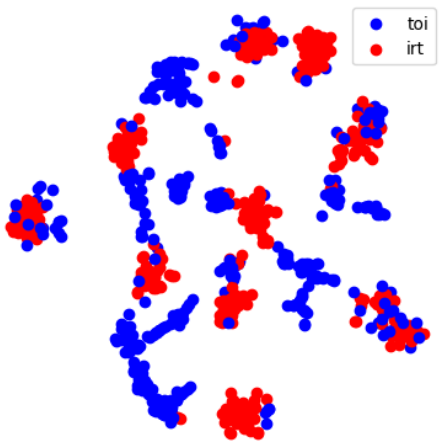
(a) Illustration of domain-invariance



(b) Illustration of contrasting task



(c) Illustration of domain-invariance



(d) Illustration of contrasting task

Figure 4.5: Visualization results when domain shift is domain G to domain N. At the top, the ToI is MNIST, IrT is Fashion-MNIST. At the bottom, the ToI is MNIST, IrT is EMNIST.

Table 4.3: Ablation studies on X-NIST with classification accuracy (%) averaged over 10 runs

Domain shift	G→ C	G→ E	G→ N	C→ G	N→ G
w/o dual mixup	59.9	63.1	65.2	74.2	76.0
w/o dual contrastive	56.4	66.2	48.4	75.0	61.5
w/o adaptive temperature	61.39	68.59	77.91	76.83	81.73
triplet loss	55.17	44.19	38.35	59.46	56.77
Ours	**65.7**	**70.3**	**80.4**	**80.7**	**81.1**

4.4.3 Ablation studies

To check the importance of each module in DMCL, we perform the following ablation studies on the X-NIST.

Importance of the dual mixup We extend domain adversarial training with mixup samples. To verify this module is necessary, we remove the mixup samples used in domain adversarial training. The second row in Table 4.3 shows the averaged classification accuracy over 10 task combinations of the target ToI data when we remove dual mixup module. Overall, these results show that the dual mixup module is important. When this module is removed, the difference in performance can be up to 17%.

Importance of the dual-level contrastive learning We utilize dual-level contrastive learning to force feature disentanglement between domain-related features and task-related features. To verify this module is necessary, we remove two contrastive learning objectives. The third row in Table 4.3 shows that dual-level contrastive learning is important, and only domain adversarial training is not enough to solve ZSDA. For all domain shifts, removing contrastive learning objectives leads to worse accuracy on the target domain. This fits our expectation. Exploring the intrinsic relationships with contrastive objectives is helpful in solving ZSDA tasks. More ablation studies are shown in the supplementary.

Importance of adaptive temperature We utilize an adaptive temperature to adjust the focus of contrastive learning objectives automatically. To verify the effectiveness of this module, we replace the adaptive temperature with a constant temperature with a value of 0.5. The fourth row in Table 4.3 shows that the adaptive temperature can provide a minor improvement of domain adaptation ability. This module may not be decisive in solving the ZSDA problem, but the introduction of this module can bring a steady improvement in all situations on X-NIST.

Another contrastive learning objective In our proposal, we utilize NT-Xent loss to realize the contrastive learning objective. However, there are some other options to realize our intuition. For example, triplet loss can also be used to realize contrastive learning objectives. We take $\mathcal{L}_{con_d}$ as an example. For $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ defined in the section 4.3.4, if we assume $\mathbf{B}_i$ as an anchor sample, then $\mathbf{C}_i$ is the positive sample, and $\mathbf{A}_i$ is the negative sample. Then, triplet loss is formalized as:

$$\begin{aligned}\mathcal{L}_{trip_d}(G_D) = \max(&\|G_D(G(B_i)) - G_D(G(C_i))\|^2 - \\ &\|G_D(G(B_i)) - G_D(G(A_i))\|^2 + \alpha, 0)\end{aligned} \quad (4.6)$$

where α is a margin between positive and negative pairs.

This triplet loss can replace $\mathcal{L}_{con_d}$ in our proposal. Similarly, a similar loss shown in the following can replace $\mathcal{L}_{con_f}$.

$$\mathcal{L}_{trip_f}(G_F) = \max(\|G_F(G(B_i)) - G_F(G(A_i))\|^2 - \|G_F(G(B_i)) - G_F(G(C_i))\|^2 + \alpha, 0) \quad (4.7)$$

The fifth row in Table 4.3 shows the classification accuracy on the target domain when we use the above two triplet losses to replace dual-level contrastive losses in our proposal. We can find that there is a large gap between our original proposal and our proposal with triplet losses. This may be due to the fact that the triplet loss is difficult to optimize. In contrast, the NT-Xent loss does not have such an issue and leads to good domain adaptation ability.

4.5 Conclusion

In this paper, we propose DMCL to learn domain-invariant features that are not affected by the difference between tasks. Specifically, we design a dual mixup to synthesize intermediate samples between tasks and domains. These data bridge the gap of target ToI data. Then we design two contrastive learning objectives to explore the relationship among data from two aspects: domain and task. With contrastive learning, our model is able to separate domain-related features and task-related features. Finally, by applying adversarial learning, the resulting domain-invariant features are not affected by differences in tasks. In the evaluation, DMCL shows good performance in solving zero-shot domain adaptation problems. Moreover, we visualize our proposal on feature space and confirm our proposal works as expected.

4.6 Limitation

In this section, we discuss the limitation of our proposal. First, our proposal assumes that applying the convex combination of two samples at the pixel level will result in samples with the intermediate domain. This assumption does not always hold. For example, complex domain shifts such as style transfer will break this assumption. The experimental result on the Office-Home datasets also suggests that our method does not show superior performance on the complex domains. Hence, we need to consider other solutions to synthesize samples in the intermediate domains. Although the generative model [22] seems the best way to represent the data-generating distribution of intermediate domains, we do not want to significantly increase computational resources in our proposal. Instead, we may utilize some extension of mixup, such as CutMix [69], to relax our assumption.

Second, the motivation of this work is to avoid the sharing of private data. By introducing the zero-shot domain adaptation setting, now, we are able to avoid the sharing of target private data. However, we still need to utilize private data in the source domain. A simple solution to this issue would be combining the source-free domain adaptation setting and the ZSDA setting. Then, we can get rid of the need for private data both in the source and target domains.

Chapter 5

Conclusion

In this thesis, we aim to provide solutions to overcome domain shifts when task-relevant data from the target domain is unobtainable.

In Chapter 3, we propose a data augmentation method, termed GADA, for the domain generalization setting. This data augmentation method aims to augment training data with semantically diverse and difficult samples. Empirically, GADA shows superior domain generalization ability on three benchmarks. Further visualization and ablation studies confirm our proposal can provide hard samples, and each module in our proposal is important.

In Chapter 4, we propose a representation method, termed DMCL, for the zero-shot domain adaptation setting. DMCL aims to learn domain-invariant features in the ZSDA setting. To achieve this, we design dual-level contrastive learning objectives to force feature disentanglement between domain-level and category-level features. Then, the classifier is built upon the category-level features. Hence, the classifier will not be affected by domain shifts. The experimental results suggest this method can lead to good domain adaptation ability when label space is different in the source and target domains.

5.1 Future work

In the future, we would like to investigate our proposal in the following directions.

Domain adaptation on non-image data First, our two proposals, GADA and DMCL, aim to alleviate domain shift in the image classification task. However, the domain shift issue is not only present in image tasks but also in non-image tasks. For example, [70] argued that neural machine translation models are usually trained with fixed vocabulary but need to handle open vocabulary. In this situation, the training and test domains are different, and the domain shift happens. Compared to image tasks, how to mitigate domain shift in non-image tasks is currently not well studied. It would be exciting and meaningful to investigate how to extend our proposal in the non-image tasks.

Generalized domain adaptation setting Second, we investigate domain generalization and zero-shot domain adaptation settings in this thesis. These settings both limit the number of domains or tasks that can exist. DG assumes that the source domains and target domains are both related to the same task, and ZSDA assumes that only two domains and two tasks exist. However, in reality, we often need to handle a more general setting due to the large amount of data collected from different media. That is, we have multiple source domains and multiple target domains, and these domains may be related to multiple tasks.

In the future, we would like to investigate how to handle this general setting.

Application on realistic datasets Third, we evaluate our proposals on artificial benchmarks. Although these benchmarks contain realistic images, the domain shifts that existed in these domains are mainly caused by style transfer. In reality, the domain shift we encounter is not only due to a style shift but also illumination change, rotation, etc. Hence, it is meaningful to test our proposal on more realistic domain shifts.

Acknowledgement

I would like to express my deepest gratitude to my advisor Prof. Jun Sakuma for his support of my PhD study. Through his guidance, I gradually learned how to design experiments, how to write papers, how to make presentations, etc. If not his dedication, motivation, and energy, I will never grow up to be a qualified PhD,

I would like to sincerely thank Prof. Youhei Akimoto, and Prof. Kazuto Fukuchi. They all gave me a lot of valuable comments on my research.

I would like to extend my sincere thanks to my parents. They always tried their best to support me to do what I want. Without their support, I would never have had a chance to come to study in Japan.

Finally, I want to thank my labmates, for their help in my life in Japan and their advice on my research. I am very honored to spend four years with you.

References

[1] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pp. 1180–1189. PMLR, 2015.

[2] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, Vol. 86, No. 11, pp. 2278–2324, 1998.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

[4] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

[5] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, Vol. 17, No. 1, pp. 2096–2030, 2016.

[6] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang Lu, Yiqiang Chen, Wenjun Zeng, and Philip Yu. Generalizing to unseen domains: A survey on domain generalization. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[7] Yuan Wu, Diana Inkpen, and Ahmed El-Roby. Dual mixup regularized learning for adversarial domain adaptation. In *European Conference on Computer Vision*, pp. 540–555. Springer, 2020.

[8] Fabio M Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain generalization by solving jigsaw puzzles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2229–2238, 2019.

[9] Kuan-Chuan Peng, Ziyan Wu, and Jan Ernst. Zero-shot deep domain adaptation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 764–781, 2018.

[10] Wenxiao Xiao, Zhengming Ding, and Hongfu Liu. Implicit semantic response alignment for partial domain adaptation. *Advances in Neural Information Processing Systems*, Vol. 34, pp. 13820–13833, 2021.

[11] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3723–3732, 2018.

[12] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain generalization with mixstyle. *arXiv preprint arXiv:2104.02008*, 2021.

[13] Jinghua Wang and Jianmin Jiang. Adversarial learning for zero-shot domain adaptation. In *European Conference on Computer Vision*, pp. 329–344. Springer, 2020.

[14] Masato Ishii, Takashi Takenouchi, and Masashi Sugiyama. Zero-shot domain adaptation based on attribute information. In *Asian Conference on Machine Learning*, pp. 473–488. PMLR, 2019.

[15] Rowel Atienza. Improving model generalization by agreement of learned representations from data augmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 372–381, 2022.

[16] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *CoRR*, Vol. abs/1710.09412, , 2017.

[17] Won Young Jhoo and Jae-Pil Heo. Collaborative learning with disentangled features for zero-shot domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8896–8905, 2021.

[18] Ya Li, Xinmei Tian, Mingming Gong, Yajing Liu, Tongliang Liu, Kun Zhang, and Dacheng Tao. Deep domain generalization via conditional invariant adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 624–639, 2018.

[19] Rui Shao, Xiangyuan Lan, Jiawei Li, and Pong C Yuen. Multi-adversarial discriminative deep domain generalization for face presentation attack detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10023–10031, 2019.

[20] Minghao Xu, Jian Zhang, Bingbing Ni, Teng Li, Chengjie Wang, Qi Tian, and Wenjun Zhang. Adversarial domain adaptation with domain mixup. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34, pp. 6502–6509, 2020.

[21] Hui Tang and Kui Jia. Discriminative adversarial domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34, pp. 5940–5947, 2020.

[22] Rui Gong, Wen Li, Yuhua Chen, and Luc Van Gool. Dlow: Domain flow for adaptation and generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2477–2486, 2019.

[23] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. *Advances in neural information processing systems*, Vol. 29, , 2016.

[24] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning*, pp. 1989–1998. Pmlr, 2018.

[25] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, Vol. 115, No. 3, pp. 211–252, 2015.

[26] Zhangjie Cao, Lijia Ma, Mingsheng Long, and Jianmin Wang. Partial adversarial domain adaptation. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 135–150, 2018.

[27] Jing Zhang, Zewei Ding, Wanqing Li, and Philip Ogunbona. Importance weighted adversarial nets for partial domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8156–8164, 2018.

[28] Jian Hu, Hongya Tuo, Chao Wang, Lingfeng Qiao, Haowen Zhong, and Zhongliang Jing. Multi-weight partial domain adaptation. In *BMVC*, p. 5, 2019.

[29] Shuang Li, Chi Harold Liu, Qiuxia Lin, Qi Wen, Limin Su, Gao Huang, and Zhengming Ding. Deep residual correction network for partial domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, Vol. 43, No. 7, pp. 2329–2344, 2020.

[30] Youngeun Kim, Donghyeon Cho, Kyeongtak Han, Priyadarshini Panda, and Sungeun Hong. Domain adaptation without source data. *IEEE Transactions on Artificial Intelligence*, Vol. 2, No. 6, pp. 508–518, 2021.

[31] Ning Ding, Yixing Xu, Yehui Tang, Chao Xu, Yunhe Wang, and Dacheng Tao. Source-free domain adaptation via distribution estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7212–7222, 2022.

[32] Pietro Morerio, Riccardo Volpi, Ruggero Ragonesi, and Vittorio Murino. Generative pseudo-label refinement for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 3130–3139, 2020.

[33] Jogendra Nath Kundu, Akshay R Kulkarni, Suvaansh Bhambri, Deepesh Mehta, Shreyas Anand Kulkarni, Varun Jampani, and Venkatesh Babu Radhakrishnan. Balancing discriminability and transferability for source-free domain adaptation. In *International Conference on Machine Learning*, pp. 11710–11728. PMLR, 2022.

[34] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant feature representation. In *International Conference on Machine Learning*, pp. 10–18. PMLR, 2013.

[35] Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation. *arXiv preprint arXiv:1805.12018*, 2018.

[36] Kaiyang Zhou, Yongxin Yang, Timothy Hospedales, and Tao Xiang. Learning to generate novel domains for domain generalization. In *European Conference on Computer Vision*, pp. 561–578. Springer, 2020.

[37] Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi. Generalizing across domains via cross-gradient training. *arXiv preprint arXiv:1804.10745*, 2018.

[38] Kaiyang Zhou, Yongxin Yang, Timothy Hospedales, and Tao Xiang. Deep domain-adversarial image generation for domain generalisation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34, pp. 13025–13032, 2020.

[39] Zijian Wang, Yadan Luo, Ruihong Qiu, Zi Huang, and Mahsa Baktashmotlagh. Learning to diversify for single domain generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 834–843, 2021.

[40] Fu-En Yang, Yuan-Chia Cheng, Zu-Yun Shiau, and Yu-Chiang Frank Wang. Adversarial teacher-student representation learning for domain generalization. *Advances in Neural Information Processing Systems*, Vol. 34, , 2021.

[41] Alexander Robey, George Pappas, and Hamed Hassani. Model-based domain generalization. *Advances in Neural Information Processing Systems*, Vol. 34, , 2021.

[42] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

[43] Kartik Ahuja, Karthikeyan Shanmugam, Kush Varshney, and Amit Dhurandhar. Invariant risk minimization games. In *International Conference on Machine Learning*, pp. 145–155. PMLR, 2020.

[44] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, Vol. 35, No. 1, pp. 53–65, 2018.

[45] Jinghua Wang and Jianmin Jiang. Conditional coupled generative adversarial networks for zero-shot domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3375–3384, 2019.

[46] Chieh Hubert Lin, Chia-Che Chang, Yu-Sheng Chen, Da-Cheng Juan, Wei Wei, and Hwann-Tzong Chen. Coco-gan: Generation by parts via conditional coordinating. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4512–4521, 2019.

[47] Won Young Jhoo and Jae-Pil Heo. Collaborative learning with disentangled features for zero-shot domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8896–8905, 2021.

[48] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[49] Attila Lengyel, Sourav Garg, Michael Milford, and Jan C van Gemert. Zero-shot day-night domain adaptation with a physics prior. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4399–4409, 2021.

[50] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8789–8797, 2018.

[51] Alexander H Liu, Yen-Cheng Liu, Yu-Ying Yeh, and Yu-Chiang Frank Wang. A unified feature disentangler for multi-domain image translation and manipulation. *arXiv preprint arXiv:1809.01361*, 2018.

[52] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

[53] Tongzhou Wang and Yihan Lin. Cyclegan with better cycles. 2018.

[54] Christina Heinze-Deml and Nicolai Meinshausen. Conditional variance penalties and domain shift robustness, 2019.

[55] Mikhail Figurnov, Shakir Mohamed, and Andriy Mnih. Implicit reparameterization gradients. *Advances in Neural Information Processing Systems*, Vol. 31, , 2018.

[56] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[57] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, pp. 5542–5550, 2017.

[58] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5018–5027, 2017.

[59] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[60] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. *arXiv preprint arXiv:2007.01434*, 2020.

[61] Yuan Wu, Diana Inkpen, and Ahmed El-Roby. Dual mixup regularized learning for adversarial domain adaptation. In *European Conference on Computer Vision*, pp. 540–555. Springer, 2020.

[62] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. *CoRR*, Vol. abs/2002.05709, , 2020.

[63] Feng Wang and Huaping Liu. Understanding the behaviour of contrastive loss. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2495–2504, 2021.

[64] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, Vol. 29, No. 6, pp. 141–142, 2012.

[65] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, Vol. abs/1708.07747, , 2017.

[66] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. EMNIST: an extension of MNIST to handwritten letters. *CoRR*, Vol. abs/1702.05373, , 2017.

[67] Patrick J Grother. Nist special database 19. *Handprinted forms and characters database, National Institute of Standards and Technology*, Vol. 10, , 1995.

[68] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, Vol. 33, No. 5, pp. 898–916, 2010.

[69] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6023–6032, 2019.

[70] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.