Domain Adaptation Algorithms with Scarce Data
（不十分なデータに基づくドメイン適応手法）

by

Masato Ishii

石井 雅人

A Doctor Thesis

博士論文

Submitted to

the Graduate School of the University of Tokyo

on December 4, 2020

in Partial Fulfillment of the Requirements

for the Degree of Doctor of Information Science and Technology

in Computer Science

Thesis Supervisor: Masashi Sugiyama　杉山 将

Professor of Computer Science

ABSTRACT

Machine learning has become one of the most important technologies to provide intelligent products and systems that can enrich people's lives. The latest machine learning algorithms have achieved human-level performance in many applications, for example, face recognition, speech recognition, and multi-lingual translation. One of the most important factors of such success is the availability of a large-scale training dataset. While it is absolutely essential, obtaining such a large-scale dataset requires an expensive cost, which will be a major barrier to apply machine learning technologies to a wide variety of applications.

In this dissertation, we tackle several types of problem settings of domain adaptation with scarce data. The domain adaptation is a technology to reuse existing datasets, and it can substantially reduce the cost of obtaining the training dataset for a new domain. It basically aims to match the distribution of source data (the data we want to reuse) to that of target data (the data on which the trained model should perform well). By properly conducting the domain adaptation, the model trained with the adapted source data can perform well in the target domain. For effective adaptation, we need a sufficient amount of source and target data. However, due to the motivation of the domain adaptation, the amount of target data is often supposed to be small. Additionally, we cannot access source data in some practical cases, for example, due to data privacy issues. Therefore, how to effectively handle such data scarceness is quite important to make it possible to apply this technology to real-world problems. In this dissertation, we present three contributions to domain adaptation with scarce data.

In the first part of this dissertation, we consider the case in which we only have incomplete target data. In this setting, a certain subset of classes is missing in unlabeled target data, while all classes appear in labeled source data, and the goal is to discriminate all classes in the target domain. We call this problem setting partially zero-shot domain adaptation. To solve this problem, we utilize an adversarial training scheme and adopt instance weighting to estimate the loss related to unavailable target data in the missing classes. The instance weight is computed based on the prediction of deep neural networks, implying which instance would be similar to unseen data and having useful information for the loss estimation. This estimation makes it possible to explicitly consider all classes during the domain adaptation training even in the partially zero-shot setting, which leads to accurate adaptation between domains. Experimental results with several benchmark datasets validate the advantage of our method.

In the second part of this dissertation, we consider the most extreme case called zero-shot domain adaptation in which we do not have any target data for domain adaptation. If we do not have any information about the target data, the adaptation may be impossible. Therefore, we first clarify a possible scenario where we can assume availability of some knowledge instead of data, which enables us to effectively conduct the domain adaptation. We consider the situation where domain shift is caused by a prior change of a specific factor and assume that we know how the prior changes between the source and target domains. We call this factor an attribute, and reformulate the domain adaptation problem to utilize the attribute prior instead of target data. In our method, source data are reweighted with the sample-wise weight estimated by the attribute prior and the data itself so that they are useful in the target domain. We theoretically reveal that our method provides more accurate estimation of sample-wise transferability than a straightforward attribute-based reweighting approach. Experimental results with both toy datasets and benchmark datasets show that our method can perform well, though it does not use any target data.

In the third part of this dissertation, we consider another extreme case of domain adaptation called source-free domain adaptation in which we cannot access any source data during adaptation. In this setting, a model pretrained with source data is given instead of source data, and we aim to fine-tune this model with unlabeled target data. For distributional alignment between domains, we propose utilizing batch normalization statistics stored in the pretrained model to approximate the distribution of unobserved source data. This makes it possible to explicitly evaluate the distributional discrepancy between domains, and we conduct domain adaptation by minimizing this discrepancy. Experimental results with several benchmark datasets show that our method achieves competitive performance with state-of-the-art domain adaptation methods even though it does not require access to source data during adaptation.

In summary, this dissertation was devoted to increasing the applicability of domain adaptation. We have studied three kinds of problem settings for domain adaptation with scarce data. Our proposed algorithms are designed with a common perspective of matching data distributions between domains, and the experiments with several benchmark datasets have validated their advantages. Therefore, we conclude that we have succeeded to make it possible to apply domain adaptation to more diverse situations that is conceivable in real-world problems.

論文要旨

機械学習は、人々の生活を豊かにするインテリジェントな製品・システムを提供するために必要な、最も重要な技術の一つである。最新の機械学習技術は、顔認識、音声認識、多言語翻訳などを始めとする様々な応用において、人間に匹敵する性能を達成している。この成功を可能にした大きな要因の一つが、大量の学習データの活用である。学習データは機械学習において必要不可欠である一方、大量の学習データを取得するコストは非常に高い。したがって、今後、機械学習技術を更に様々な実応用に適用する上で、学習データを構築するコストは大きな障壁になると考えられる。

本学位論文では、不十分なデータに基づくドメイン適応に取り組んだ。ドメイン適応とは、学習データを再利用する技術であり、新規タスクのための学習データを取得するコストを大幅に低減することができる。多くのドメイン適応手法では、ソースデータ（再利用するデータ）の分布をターゲットデータ（モデルを学習したい対象のデータ）の分布に合わせることを目指す。これにより、適応後のソースデータで学習したモデルは、ターゲットデータにおいて高い性能を達成することができるようになる。効果的な適応を行うためには十分な量のソースデータとターゲットデータが必要となるが、ドメイン適応が必要となる多くの場面では、ターゲットデータが少ないと考えられる。また、データの権利的・倫理的な問題によって、ソースデータにアクセスできない場面も実応用上は想定される。したがって、不十分なデータをいかに効果的に活用するかが、ドメイン適応を実問題へ適用可能にするために重要である。本学位論文では、不十分なデータに基づくドメイン適応について三つの手法を提案する。

第一の手法では、不完全なターゲットデータしか得られない場合を考える。ここでは、いくつかのクラスがターゲットデータに含まれていない一方、ソースデータには全てのクラスが含まれているという状況において、ターゲットドメインで全てのクラスを高精度に識別可能なモデルを学習することを目指す。部分的ゼロショットドメイン適応と名づけたこの問題設定に対し、新しいドメイン適応手法を提案する。提案手法では、ターゲットデータに含まれていないクラスの未知データに対する損失を推定するため、従来の敵対的学習に基づくアプローチに対してサンプル重みづけを併用する。サンプル重みは、どのサンプルが未知のデータに類似し、その損失の推定に有用かという情報を反映するように、モデルの予測結果に基づいて推定する。この推定により、部分的ゼロショットドメイン適応の設定においても、全てのクラスのデータを考慮した正確なドメイン適応を行うことができる。複数のベンチマークデータを用いた実験により、提案手法の有効性を示す。

第二の手法では、ターゲットデータが全く得られないゼロショットドメイン適応と呼ばれる最も極端な場合に取り組む。もし、ターゲットデータに関する情報が全く得られなければ、適応は不可能である。したがって、まずは、効果的なドメイン適応が可能となるためには、どのような知識がデータの代わりに得られていればよいかという点を明らかにする。ここでは、特定の要因の事前分布の違いに起因してドメイン間のデータ

分布がずれている状況を考え、その事前分布の変化が知識として得られることを仮定する。この特定の要因を属性と呼び、データの代わりに属性の事前分布を用いるようにドメイン適応を再定式化した。提案手法では、ソースデータと属性の事前分布からサンプルごとの重みを推定し、重みづけしたソースデータをターゲットデータと見なせるようにする。理論的な解析により、提案手法は、属性のみを用いた重みづけを行う従来手法よりも、サンプルごとの重みを正確に推定できることを示す。また、人工データとベンチマークデータを用いた実験により、ターゲットデータを全く用いなくても、提案手法によって適切なドメイン適応が実現できることを示す。

　第三の手法では、別の極端な場合として、ソースデータが全く得られない状況におけるドメイン適応に取り組む。この問題設定では、ソースデータを用いて事前学習されたモデルがソースデータの代わりに与えられており、このモデルをラベルなしターゲットデータのみを用いてターゲットドメインに適応することを目指す。ドメイン間でデータの分布を合わせることを可能にするため、モデル内に保存されたバッチ統計量を利用し、観測されていないソースデータの分布を近似することを提案する。これにより、ドメイン間の分布のずれを明示的に評価することが可能となり、この分布のずれを最小化することでドメイン適応を実現する。複数のベンチマークデータを用いた実験により、提案手法は適応時にソースデータへのアクセスを必要としないにもかかわらず、最新のドメイン適応手法と同等の性能を達成していることを示す。

　本学位論文では、ドメイン適応が適用可能な問題設定の範囲を拡大するため、不十分なデータに基づくドメイン適応について３つの問題設定に取り組んだ。提案手法はいずれもドメイン間でデータ分布を合わせるという共通の観点で設計されており、実験によってその有効性が示された。以上より、本学位論文によって、実問題で直面し得る多様な状況にもドメイン適応を適用可能にすることができたと結論付ける。

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we first describe the background and motivation of our work. Then, we give an overview of our contributions and the organization of this dissertation.

1.1 The age of machine learning

The age of machine learning has arrived. It has become an indispensable technology to provide intelligent products and systems that can enrich people's lives. If you look around, you will see an abundance of smart services made possible by machine learning, for example, face recognition by digital cameras [Kortli et al., 2020], speech recognition by smart speakers [Malik et al., 2020], and automatic multi-lingual translation in online meeting software [Dabre et al., 2020], just to name a few. We have greatly benefited from machine learning in our daily lives and even more in the future.

In general, given a particular task and performance metric, machine learning aims to automatically construct a model that achieves good performance at the task by learning from experience [Mitchell, 2006]. Here, we mainly focus on a supervised-learning setting shown in Fig. 1.1. In this setting, a model is trained to accurately predict a label of a test input sample by learning with a training dataset that contains pairs of input examples and corresponding correct labels. To evaluate how the performance of the trained model generalize to unseen data, we often validate the model's performance with test data that are not used for training [Hastie et al., 2009]. In the long history of machine learning, many kinds of models have been proposed [Bishop, 2006], but, recently, deep neural networks are often adopted due to their promising performance in various applications [Hatcher and Yu, 2018].

One of the most important factors of recent success by machine learning is the availability of a large-scale training dataset. For example, in the field of computer vision, there are several million-scale datasets that are publicly available and are commonly used in the literature [Russakovsky et al., 2015, Kemelmacher-Shlizerman et al., 2016, Zhou et al., 2017, Kuznetsova et al., 2020]. The most influential dataset among them is ImageNet [Russakovsky et al., 2015], which is an image database organized according to the WordNet [Miller, 1998] hierarchy. Based on this dataset, Im-

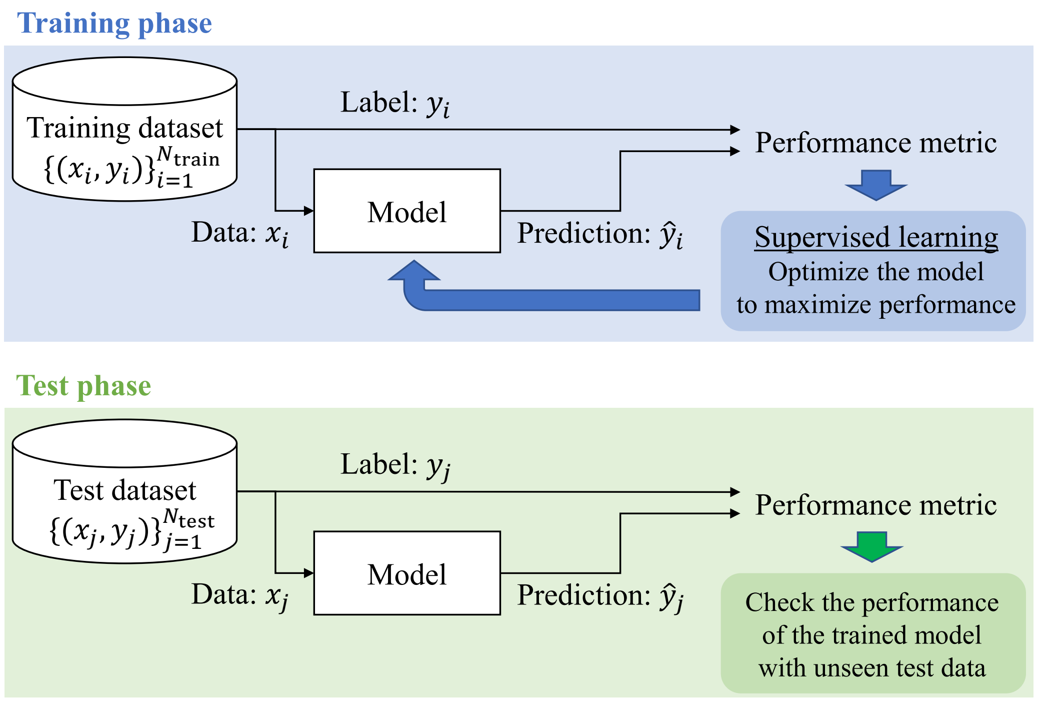


Figure 1.1: A setup of supervised learning.

ageNet Large Scale Visual Recognition Challenge (ILSVRC) has been annually held since 2010 to 2017. While this dataset contains several sub-datasets for different tasks, "ImageNet" often indicates the object classification dataset used in ILSVRC2012, which contains about 1 million images with 1,000 object classes. Through ILSVRC, a lot of innovative techniques on visual recognition have been developed [Krizhevsky et al., 2012, He et al., 2016, Hu et al., 2018a]. The best accuracy on ImageNet was 71.8 % at ILSVRC2010, but it reached 97 % at ILSVRC2017, which exceeds human-level performance [Hu et al., 2018a]. Large-scale datasets enable us to train highly complex models, which makes it possible to effectively tackle complicated tasks that often appear in real-world applications. The standardization and construction of such datasets has substantially accelerated the research and development of machine learning technology.

1.2 Issues in applying machine learning technology to real-world problems

As described in the previous section, the huge success of machine learning has been mainly achieved in standardized tasks where a sufficiently large amount of training data are available. On the other hand, when we consider to apply machine learning technology to more diverse real-world problems, we cannot always expect a large-scale training dataset, because the cost to obtain such a dataset for each task is prohibitively expensive in many practical situations [Pan and Yang, 2010]. This limitation leads to two major issues that have intensely been tackled for decades in the field of machine learning: data scarcity [Roh et al., 2019] and domain shift [Quionero-Candela et al., 2009].

1.2.1 Data scarcity

When we use a small-scale dataset for training, a model tends to be overly fitted to the training data during training, if it has a sufficient capability to learn from training data. Although the performance of the trained model would be extremely good in the training data, it should be poor in unseen test data. This phenomenon is called overfitting [Hastie et al., 2009]. To detect overfitting, we often conduct model validation after training [Hastie et al., 2009]. In validation, we evaluate the performance of the trained model with validation data that are split from the original training data and are not used for training. If the performance on validation data is substantially degraded, we can consider that overfitting occurs. However, when training dataset is quite small, this validation process does not work well, because computed performance should have a large variance. Even when we can detect overfitting through validation, how to avoid overfitting for obtaining better performance still remains as a major problem.

Since a small-scale dataset contains less information on data distribution, it is essentially hard to extract much information for generalization in a statistically stable manner [Hastie et al., 2009]. Therefore, additional assumptions or conditions are required to effectively avoid overfitting. In this dissertation, we focus on a kind of transfer learning setting where we additionally have another dataset that is constructed in a similar domain to our target [Pan and Yang, 2010]. For example, in case of the object recognition task, even when we only have a small number of real images that capture the target object, it may be possible to obtain many synthesized images using computer graphics [Peng et al., 2017]. In the most straightforward way, we can simply use the additional dataset to train a model. However, this simple method would result in poor performance due to another issue, called domain shift [Quionero-Candela et al., 2009].

1.2.2 Domain shift

A domain shift is a change in data distributions between the training dataset and the test dataset [Quionero-Candela et al., 2009]. It naturally occurs in the setting of transfer learning, because the additional training data, called source data, are obtained in a similar but different domain from the original training data, called target data [Wilson and Cook, 2020]. Since many supervised-learning algorithms assume that training data stem from the same distribution as test data [Hastie et al., 2009], a model trained with source data does not necessarily work well at target data as visualized in the left-hand side of Fig. 1.2 [Quionero-Candela et al., 2009]. To overcome this problem, how to fill the distributional discrepancy between source and target data has been in-

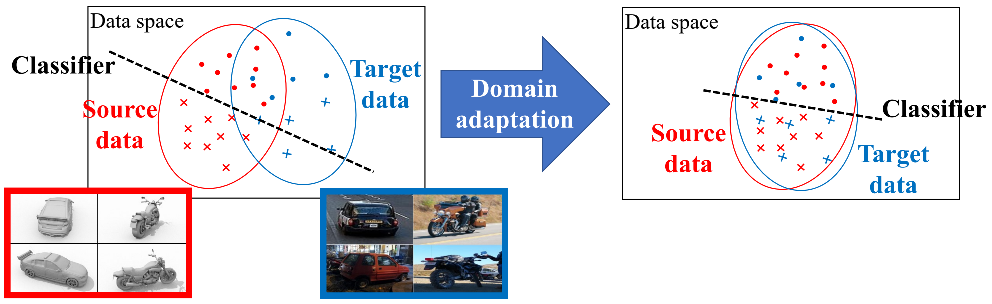


Figure 1.2: A concept of domain shift and domain adaptation.

tensely studied and is called domain adaptation. By reducing the discrepancy, a model trained with the adapted source data works well in the target domain as shown in the right-hand side of Fig. 1.2 [Wilson and Cook, 2020].

To effectively reduce the distributional discrepancy, we need a sufficient amount of source and target data. However, due to the motivation of the domain adaptation, the amount of target data is often small, which makes it quite hard to extract precise information on target distribution that is used to estimate the discrepancy [Motiian et al., 2017]. Additionally, we cannot access source data in some practical cases [Kundu et al., 2020, Li et al., 2020]. For example, when we use machine learning to optimize some personalized services, we can construct a large-scale source dataset by collecting personal data from all users, but it should be problematic to use it for domain adaptation in each user's environment due to data privacy issues [Voigt and Bussche, 2017]. Therefore, domain adaptation algorithms with scarce data are highly demanded to make it possible to apply domain adaptation to a wide range of real-world applications.

1.3 Domain adaptation and transfer learning

As described in the previous section, we focus on the transfer learning setting, particularly domain adaptation, in this dissertation. Transfer learning aims to help improve the training of a model in the target domain using the knowledge in the source domain [Pan and Yang, 2010]. Meanwhile, in the literature of transfer learning [Pan and Yang, 2010, Csurka, 2017], domain adaptation was treated as a particular case of transfer learning that leverages labeled source data to learn a model for unseen or unlabeled data in a target domain within the same task. However, since recent studies on domain adaptation do not necessarily assume the availability of labeled source data [Li et al., 2020, Liang et al., 2020a] nor a common task between domains [You et al., 2019, Kundu et al., 2020], we can consider that domain adaptation has come to cover a broad range of transfer learning settings, and there is actually almost no distinction between domain adaptation and transfer learning in these days.

Naturally, domain adaptation is closely related to other transfer learning techniques [Csurka, 2017]. Sample selection bias correction [Cortes et al., 2008] has a similar motivation to domain adaptation; it also aims to make a trained model work well under a different distribution from the original training dataset. However, it assumes a more specific situation than domain adaptation in which distributional discrepancy between training and test data is caused by non-random sample selection when constructing the training dataset. Due to this assumption, it is applicable only when both source and target data are constructed for a common task and share the common underlying data distribution.

Multi-task learning [Caruana, 1997] aims to train a model by jointly using both source and target datasets that are constructed for different tasks. It can achieve better performance than learning source and target tasks independently, when source and target tasks are sufficiently related to each other [Evgeniou and Pontil, 2004]. For joint training, multi-task learning requires labeled target data, while domain adaptation is often tackled under an unsupervised setting in which only unlabeled target data are available. Additionally, it also needs to simultaneously access both source and target data, which is different from the setting of another cross-task transfer learning scheme, continual learning [Parisi et al., 2019].

Continual learning [Parisi et al., 2019], which is also known as lifelong machine learning [Chen and Liu, 2018], aims to continuously train a model from a sequence of multiple tasks. In this setting, a target task to learn is dynamically switched to another task over time, and training data for previous tasks cannot be directly accessed during learning of a current task differently from multi-task learning. Therefore, one major problem in continual learning is how to prevent a model from forgetting the knowledge obtained from previous tasks, so-called catastrophic forgetting [McCloskey and Cohen, 1989].

1.4 Contributions of this dissertation

This dissertation contributes to developing domain adaptation algorithms with scarce data. As noted in the previous section, when only scarce data are available during adaptation, it is quite hard to precisely estimate the distributional discrepancy between domains. We have tackled this challenging problem and have developed three algorithms that can work effectively for each specific type of data scarcity.

In the first contribution, we introduce a new problem setting, called partially zero-shot domain adaptation, which has not been explored in the literature but can appear in many practical situations. In this setting, a certain subset of classes is missing in target data, while all classes are to be discriminated after adaptation. We present a novel method for partially zero-shot domain adaptation, in which a new instance-weighting strategy is adopted to effectively match distributions between domains while considering unobservable missing-class target data. We validate its advantage in the experiments with several benchmark datasets.

In the second contribution, we consider the most extreme case called zero-shot domain adaptation in which no target data are available for adaptation. Here, we assume that some knowledge is available instead of data to conduct domain adaptation. We present that a certain kind of information, which is a prior change of an attribute, enables us to effectively align the source distribution to the target one. The attribute is expected to represent a factor that causes the difference in distributions between domains, and we theoretically clarify requirements for the attribute to be useful in adaptation. Our method conducts domain adaptation with such attribute information, and its advantage is shown in both theoretical analysis and empirical evaluations.

In the third contribution, we consider another extreme case called source-free domain adaptation in which no source data are available during adaptation. In this setting, a model pretrained with source data is given instead of source data, and our goal is to fine-tune this model with unlabeled target data. Here, we assume that the pretrained model includes a batch normalization module, which is widely adopted in modern neural networks. By using batch statistics stored in the module, we estimate the source distribution and consequently the distributional discrepancy between domains. Then, we minimize this discrepancy to conduct domain adaptation without access to source data. Experimental results with several benchmark datasets show that our method achieves comparable performance with state-of-the-art standard domain adaptation methods though it does not use any source data for adaptation.

1.5 Organization of this dissertation

This dissertation consists of six chapters. In this first chapter, we have given the introduction of this dissertation. We have described the motivation of domain adaptation and have clarified our focus of this dissertation, which is domain adaptation with scarce data. The remaining chapters are organized as follows.

Chapter 2 gives a literature review on domain adaptation. After describing the simplest case of domain adaptation, we introduce several variants of domain adaptation methods. These variants are designed to tackle harder situations, which are related to our work.

Chapter 3 gives our first contribution to domain adaptation with scarce data. We first define our new problem setting called partially zero-shot domain adaptation and reveal the problem of standard domain adaptation methods under this setting. Then, we describe a novel method that utilizes both adversarial training and instance weighting to tackle this problem. We present a setup of experiments and their results to show the advantage of our method.

Chapter 4 gives our second contribution. We first formulate a baseline method for standard domain adaptation and introduce existing works on zero-shot domain adaptation. Then, we present our approach and method, and theoretical analysis is also provided to clarify characteristics of our method. Experimental results with toy datasets and benchmark datasets are also shown to validate the advantage of our method.

Chapter 5 gives our third contribution. We start by introducing prior works on domain adaptation and batch normalization to motivate our main idea, which is the usage of batch normalization statistics for source-free domain adaptation. Then, we present our domain adaptation method for the source-free setting and its empirical evaluation with several benchmark datasets.

Chapter 6 gives the conclusion of this dissertation and discuss future directions to make domain adaptation techniques more useful in real-world applications.

Chapter 2

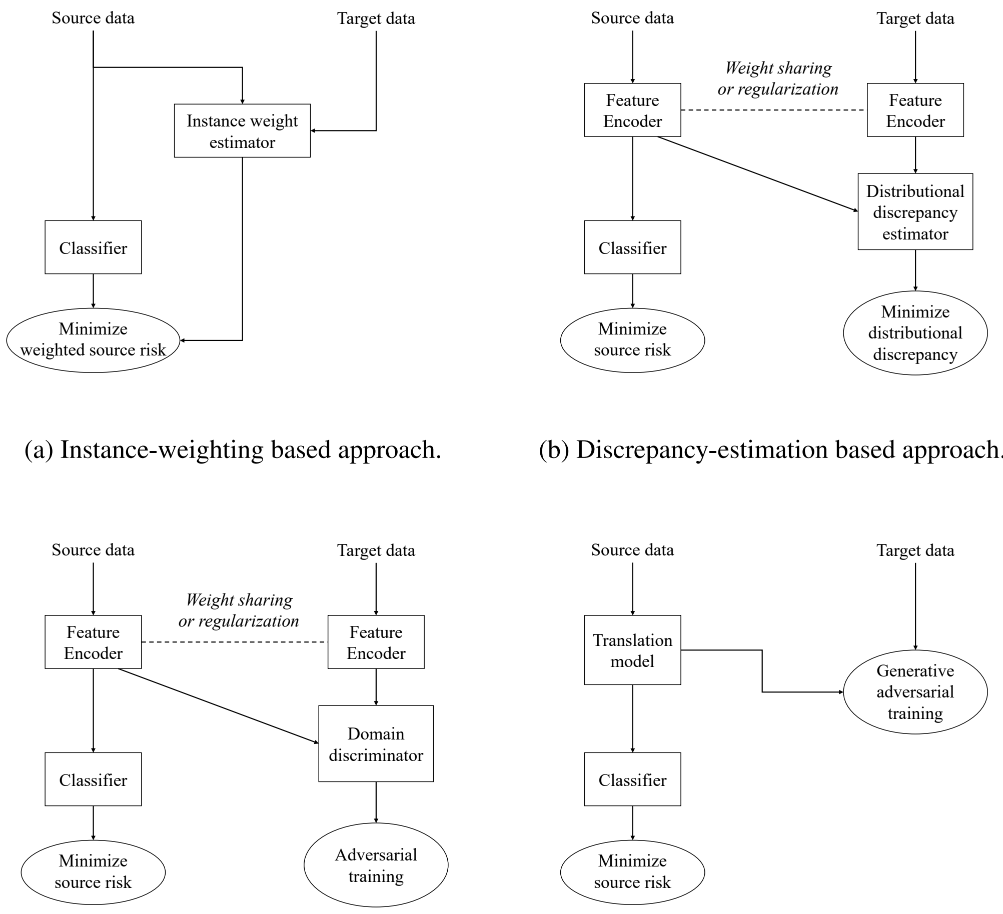
Literature review on domain adaptation

This chapter gives a literature review on domain adaptation. After describing the motivation of domain adaptation, we formulate the simplest case of domain adaptation, called closed-set domain adaptation, and review existing studies on this problem via categorizing them into four approaches. Then, we introduce several variants of domain adaptation methods that are designed to tackle harder situations than closed-set domain adaptation. Finally, we close this chapter by briefly introducing our contributions on domain adaptation with scarce data.

2.1 Motivation of domain adaptation

Domain adaptation has been intensely studied in a long history of machine learning researches [Csurka, 2017, Wilson and Cook, 2020] to overcome the domain shift problem introduced in Section 1.2.2. In domain adaptation, we consider two domains, called the source domain and target domain, which have different data distributions from each other. In the source domain, we can obtain annotated data to train a model, and the obtained source data are expected to be sufficiently large-scaled in general. On the other hand, in the target domain, we want to make a model work well after training, and available target data are often assumed to be small-scaled. Additionally, in unsupervised domain adaptation, which is a popular setting in the literature [Csurka, 2017, Wilson and Cook, 2020], the labels of target data are not available, and we cannot directly train a model with target data. Due to distributional discrepancy between domains, simply training a model with source data would result in poor performance of the trained model at the target domain [Quionero-Candela et al., 2009]. Note that, in this dissertation, we basically consider this unsupervised setting, unless otherwise noted. Under such a situation, domain adaptation aims to train a model that works well in the target domain by utilizing given source and target data.

Domain adaptation algorithms generally adopt a certain kind of "adaptation" techniques to tackle the domain shift. In many cases, they adapt labeled source data to the target domain based on given target data so that a model trained with the adapted data achieves substantially good performance in the target domain. From this perspective,



(a) Instance-weighting based approach. (b) Discrepancy-estimation based approach.

(c) Adversarial-training based approach. (d) Generative approach.

Figure 2.1: General setups in representative approaches for closed-set domain adaptation.

domain adaptation can be viewed as a special case of transfer learning [Wilson and Cook, 2020]. Domain adaptation also shares a similar motivation to sample selection bias correction [Cortes et al., 2008].

2.2 The simplest case: closed-set domain adaptation

In this section, we introduce the simplest case of domain adaptation, which is called closed-set domain adaptation [Wilson and Cook, 2020]. After formulating closed-set domain adaptation, we present four representative approaches to tackle this problem shown in Fig. 2.1.

2.2.1 Formulation

We first formulate closed-set unsupervised domain adaptation for a classification task, which is the most basic setting in the literature [Wilson and Cook, 2020]. Let $x \in$

$\mathbb{R}^m$, $y \in \mathcal{C}$, and $d \in \{\mathrm{S}, \mathrm{T}\}$ denote input data, labels, and domains, respectively. Here, m is the dimensionality of input data, $\mathcal{C}$ is a set of class candidates, and $\{\mathrm{S}, \mathrm{T}\}$ represent the source and target domains, respectively. Let us consider the situation where we are given labeled source data $\mathcal{D}_{\mathrm{S}} = \{(x_i^{\mathrm{S}}, y_i^{\mathrm{S}})\} \sim p_{\mathrm{S}}(x, y)$ and unlabeled target data $\mathcal{D}_{\mathrm{T}} = \{x_i^{\mathrm{T}}\} \sim p_{\mathrm{T}}(x)$. The goal of domain adaptation is to train a model that can accurately predict labels of input data in the target domain. More specifically, supposing a classification model $f : \mathbb{R}^d \to \mathcal{C}$ is parameterized by θ, the goal is to obtain the optimal θ that minimizes the target risk defined as

$$R_{\mathrm{T}}(\theta) = \sum_{y \in \mathcal{C}} \int l(x, y, \theta) p_{\mathrm{T}}(x, y) \mathrm{d}x, \tag{2.1}$$

where $l(x, y, \theta)$ is a loss when y is predicted by f with θ at x. If labeled target data were given, the target risk could be empirically approximated as

$$\hat{R}_{\mathrm{T}}(\theta) = \frac{1}{|\mathcal{D}_{\mathrm{T}}|} \sum_i l(x_i^{\mathrm{T}}, y_i^{\mathrm{T}}, \theta). \tag{2.2}$$

However, in the unsupervised domain adaptation setting, labels of target data are not available, which makes it impossible to calculate this empirical target risk directly.

Since given source data stem from a different distribution than that of target data, those data cannot be directly used in empirical approximation of Eq. (2.1). To solve this issue, domain adaptation methods aim to minimize the target risk by utilizing both labeled source data and unlabeled target data in another way. In closed-set domain adaptation, the class candidates are common between domains. This condition enables us to estimate the target risk with given source data effectively as shown in the following subsections.

Before presenting representative approaches for closed-set domain adaptation, we introduce an important condition, called covariate shift [Shimodaira, 2000], which is often assumed explicitly or implicitly in the existing studies. Under covariate shift, the class-posterior probability $p(y|x)$ is common in both the source and target domains, though data distributions $p(x)$ are different between domains:

$$p_{\mathrm{S}}(y|x) = p_{\mathrm{T}}(y|x) \text{ and } p_{\mathrm{S}}(x) \neq p_{\mathrm{T}}(x). \tag{2.3}$$

This condition indicates that the class of a certain input sample can be determined independently of its domain, and it is intuitively reasonable in many pattern recognition tasks.

2.2.2 Instance-weighting based approach

In the instance-weighting based approach [Shimodaira, 2000, Huang et al., 2007, Sugiyama et al., 2008, Kanamori et al., 2009, Wen et al., 2015, Xia et al., 2018], the target risk is estimated by calculating the source risk with instance weights. Specifically, the target risk in Eq. (2.1) can be rewritten as

$$\begin{aligned} R_{\mathrm{T}}(\theta) &= \sum_{y \in \mathcal{C}} \int w(x,y) l(x,y,\theta) p_{\mathrm{S}}(x,y) \mathrm{d}x, && (2.4) \\ w(x,y) &= \frac{p_{\mathrm{T}}(x,y)}{p_{\mathrm{S}}(x,y)}, && (2.5) \end{aligned}$$

where $w(x, y)$ is an instance weight for the corresponding data (x, y). Intuitively, the weight is large if the corresponding sample often appears in the target domain but rarely appears in the source domain, while it is small in the opposite case. Since the right-hand side of Eq. (2.4) is the expectation over the distribution of source data, it can be empirically approximated with source data in a similar manner to that in Eq. (2.2), while the weight of each source sample is to be estimated. Consequently, how to estimate the instance weight is the major problem tackled in this approach. Once estimated, it is used to calculate the weighted source risk, and a model is trained by minimizing the weighted risk.

By assuming the covariate shift, the instance weight can be simplified as follows:

$$\frac{p_{\mathrm{T}}(x,y)}{p_{\mathrm{S}}(x,y)} = \frac{p_{\mathrm{T}}(y|x)p_{\mathrm{T}}(x)}{p_{\mathrm{S}}(y|x)p_{\mathrm{S}}(x)} = \frac{p_{\mathrm{T}}(x)}{p_{\mathrm{S}}(x)} = w(x). \tag{2.6}$$

Note that this simplified weight does not depend on label information, which means that it can be estimated by using both source and target data without labels. Since the weight is formulated as the ratio between p_{S} and p_{T}, estimation of the weight can be considered as density ratio estimation, which has been intensely studied in the field of machine learning [Sugiyama et al., 2012]. For instance, Huang et al. [2007] proposed Kernel Mean Matching (KMM), which estimates the weight by matching the mean of weighted source data with that of target data in a kernel feature space. Sugiyama et al. [2008] adopted the Kullback-Leibler divergence instead of difference of the means in a feature space, while a simple but effective least-square criterion is used in Kanamori et al. [2009]. These are general methods for density ratio estimation, but some of them have recently been extended for the purpose of domain adaptation in terms of stability [Xia et al., 2018] or computational efficiency [Wen et al., 2015].

2.2.3 Discrepancy-estimation based approach

In the previous approach, the distribution of target data is approximated by that of weighted source data. Although it is quite simple and is theoretically validated [Shimodaira, 2000], it would show unstable results when the target distribution is far from the source distribution [Cortes et al., 2010], because such a situation induces extremely large weights to appear in weighted source data, which leads to overfitting of the trained model to a small subset of source data. To avoid this issue, several recent studies [Csurka, 2017] tried to obtain domain-invariant feature representation by adopting a trainable feature-transform so that the source risk sufficiently approximates the target risk. Once such feature representation is obtained, the target risk can be minimized by simply minimizing the empirical source risk in the feature space without instance weights. Note that, obviously, the feature representation should also be also discriminative to train a classifier as well as domain-invariant [Ben-David et al., 2010].

In a discrepancy-estimation based approach [Fernando et al., 2013, Gretton et al., 2012, Long et al., 2015, Courty et al., 2017, Long et al., 2017, Sun et al., 2017, Yan et al., 2017, Damodaran et al., 2018, Rozantsev et al., 2018], the distributional discrepancy between source and target features is explicitly evaluated, and it is minimized to obtain good feature representation by which the distribution of source features matches that of target features. Therefore, how to compute the distributional discrepancy is the most important research topic in this approach. Additionally, since only minimizing distributional discrepancy leads to a trivial solution, which is to map any data into the same feature, how to avoid this trivial solution is also important. The most common condition adopted to avoid the trivial solution is that the feature representation should also be discriminative as well as domain-invariant to train a good classifier. To satisfy this condition, the empirical source risk is often minimized jointly in addition to the distributional discrepancy as shown in Fig. 2.1(b).

Maximum Mean Discrepancy (MMD) [Gretton et al., 2012] has often been adopted in this approach to evaluate the distributional discrepancy. MMD between p_{S} and p_{T} is defined as the following equation:

$$\mathrm{MMD}(p_{\mathrm{S}}(z), p_{\mathrm{T}}(z)) = \left\| E_{z_{\mathrm{S}} \sim p_{\mathrm{S}}(z)} \phi(z_{\mathrm{S}}) - E_{z_{\mathrm{T}} \sim p_{\mathrm{T}}(z)} \phi(z_{\mathrm{T}}) \right\|_{\mathcal{H}_k}, \tag{2.7}$$

where $z \in \mathbb{R}^{m'}$ is a feature representation of x, $\mathcal{H}_k$ denotes a Reproducing Kernel Hilbert Space (RKHS) endowed with a characteristic kernel k, and ϕ is a mapping function to $\mathcal{H}_k$. Importantly, MMD becomes zero if and only if $p_{\mathrm{S}} = p_{\mathrm{T}}$. Therefore, minimizing MMD by training a feature transform leads to obtaining domain-invariant feature representation where the source distribution matches the target distribution [Rozantsev et al., 2018]. For more precise matching, Long et al. [2015] extended MMD to use multiple kernels, while it is also extended to handle multiple feature representations in Long et al. [2017]. Yan et al. [2017] adopted a class-wise weight in MMD to flexibly deal with class-prior change between domains.

Fernando et al. [2013] have shown that simply aligning second-order statistics (mean and covariance) between domains works well, which can be done by choosing an appropriate linear subspace of the input-data space. Sun et al. [2017] adopted a similar idea but also extended it to utilize a non-linear feature transform. More recently, optimal transport has been used to evaluate the distributional discrepancy in Courty et al. [2017], Damodaran et al. [2018], and they have shown that it works substantially better than MMD especially when the distribution of target data is largely different from that of source data.

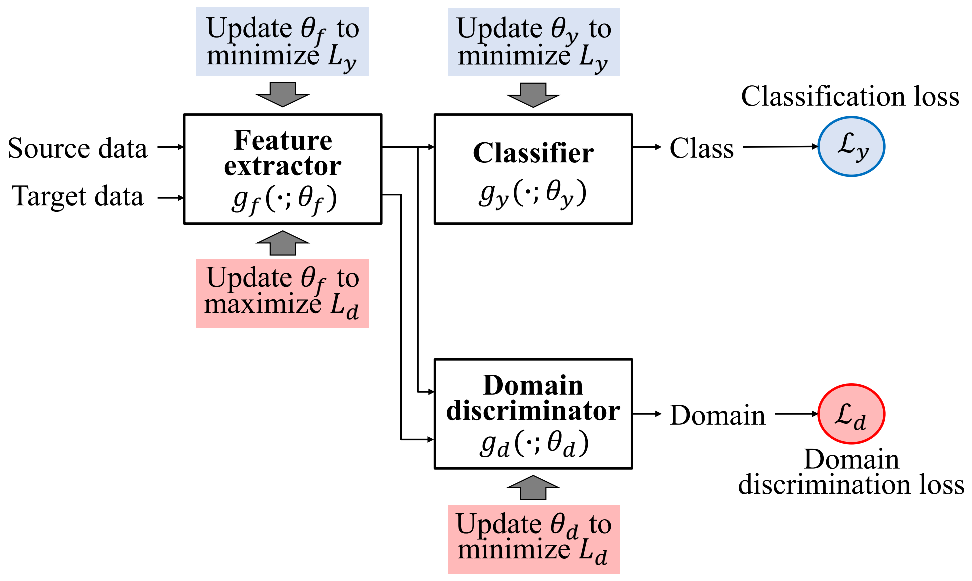

Update θ_f to minimize L_y
Update θ_y to minimize L_y
Classification loss
Source data
Target data
Feature extractor $g_f(\cdot;\theta_f)$
Classifier $g_y(\cdot;\theta_y)$
Class
$\mathcal{L}_y$
Update θ_f to maximize L_d
Domain discriminator $g_d(\cdot;\theta_d)$
Domain
$\mathcal{L}_d$
Domain discrimination loss
Update θ_d to minimize L_d

Figure 2.2: Domain Adversarial Neural Network.

2.2.4 Adversarial-training based approach

Adversarial training has been intensely adopted in state-of-the-art domain adaptation methods similarly to recent generative models called Generative Adversarial Networks (GANs) [Goodfellow et al., 2014]. In adversarial-training based approach [Ganin et al., 2016, Saito et al., 2017, Tzeng et al., 2017, Long et al., 2018, Shu et al., 2018, Deng et al., 2019, Zhang et al., 2019, Jiang et al., 2020], an auxiliary classifier, which is often called a domain discriminator, is adopted in addition to the target classifier as shown in Fig. 2.1(c). Given features extracted from input data, it is trained to discriminate which domain those features stem from. On the other hand, feature representation is jointly trained to fool the domain discriminator. Through this adversarial training, domain-invariant feature representation is obtained, which enables us to train a good classifier by simply minimizing the source risk in the feature space.

The pioneering work in this approach is done by Ganin et al. [2016], and they proposed the Domain Adversarial Neural Network (DANN). Since DANN is one of the important baselines in our work, we describe how DANN works in detail. As shown in Fig. 2.2, three modules are jointly trained in DANN: feature extractor $g_f : \mathbb{R}^m \to \mathbb{R}^{m'}$, classifier $g_y : \mathbb{R}^{m'} \to \mathcal{C}$, and domain discriminator $g_d : \mathbb{R}^{m'} \to \{\mathrm{S}, \mathrm{T}\}$. This joint training is based on two losses. The first loss is a classification loss that penalizes misclassification of the classifier:

$$\mathcal{L}_y(\theta) = \sum_{y \in \mathcal{C}} \int l_y(x, y, \theta) p_{\mathrm{S}}(x, y) \mathrm{d}x, \tag{2.8}$$

$$l_y(x, y, \theta) = l_{\mathrm{cp}}(g_y(g_f(x; \theta_f); \theta_y), y), \tag{2.9}$$

where $\theta = \{\theta_f, \theta_y, \theta_d\}$ is a set of trainable parameters of the three modules and l_{cp} is a loss function for class prediction. Note that $\mathcal{L}_y$ is simply defined as the source risk.

The second loss is a domain discrimination loss that penalizes mis-discrimination of the domain discriminator:

$$\mathcal{L}_d(\theta) = \frac{1}{2}\int l_d(x,d,\theta)p_{\mathrm{S}}(x)\mathrm{d}x + \frac{1}{2}\int l_d(x,d,\theta)p_{\mathrm{T}}(x)\mathrm{d}x, \tag{2.10}$$

$$l_d(x,d,\theta) = l_{\mathrm{dp}}(g_d(g_f(x;\theta_f);\theta_d),d), \tag{2.11}$$

where l_{dp} is a loss function for domain prediction. Based on these two losses, training DANN is formulated as the following optimization:

$$\mathcal{L}(\theta) = \mathcal{L}_y(\theta) - \lambda\mathcal{L}_d(\theta), \tag{2.12}$$

$$\theta_d^* = \arg\max_{\theta_d} \mathcal{L}(\theta), \tag{2.13}$$

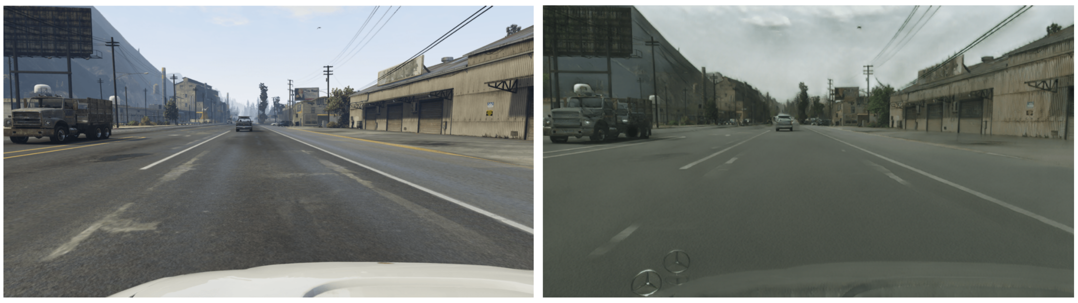
$$(\theta_f^*, \theta_y^*) = \arg\min_{\theta_f,\theta_y} \mathcal{L}(\theta). \tag{2.14}$$

The optimization in Eq. (2.13) aims to discriminate domains of the input data by g_d, while that in Eq. (2.14) aims to fool g_d by g_f as well as accurately classifying the data by g_y. As a result, $g_f(x)$ becomes good feature embedding for domain adaptation, because it would be easy to classify as well as hard to discriminate domains. Goodfellow et al. [2014] has proved that, under some strict assumptions, this adversarial training minimizes distributional discrepancy in terms of the Jensen-Shannon divergence. Bousmalis et al. [2016] has empirically shown that DANN generally works better than MMD-based domain adaptation methods.

After DANN was proposed, many domain adaptation studies have adopted this adversarial training based approach and extended DANN in several directions. While DANN only matches marginal distributions between domains, several recent methods [Saito et al., 2017, Long et al., 2018, Shu et al., 2018, Deng et al., 2019, Jiang et al., 2020] aim to match each class-conditional distribution by using a sort of pseudo labels to boost the classification accuracy. Tzeng et al. [2017] adopted two feature extractors to use them in respective domains, which improves flexibility of feature transform to effectively match the feature distributions. Zhang et al. [2019] clarified connection between adversarial training and a traditional domain adaptation theory [Ben-David et al., 2010] and designed a specific loss for adversarial training to receive benefits from theoretical guarantee on target risk minimization.

2.2.5 Generative approach

If it is possible to "translate" given source data from a source domain to a target domain, a target classifier can be obtained by simply training with the translated data as shown in Fig. 2.1(d). In generative approach [Benaim and Wolf, 2017, Hoffman et al., 2018, Murez et al., 2018, Li et al., 2019], such translation is realized by training a conditional generative model that generates appropriate target data from given source data. Figure 2.3 shows an example of image-translation from computer graphics to a real image adopted in Hoffman et al. [2018]. Especially when conditional



(a) An original source image (synthetic image). (b) An adapted image (translated to real images).

Figure 2.3: An example of translated images. These images are from Hoffman et al. [2018]

GANs [Mirza and Osindero, 2014] are used for training the translation model, this approach is quite similar to the adversarial-training based approach, because both approaches utilize adversarial training to achieve distributional alignment between domains. However, the generative approach conducts the distributional alignment in the input-data space, while the adversarial-training approach does it in a certain feature space. An important benefit of this approach is understandability of how adaptation goes on by checking adapted source data.

Since the goal is to train a good classifier, the translation model should be carefully trained so as to generate data that are sufficiently informative to train the classifier. To this end, cycle consistency regularization [Zhu et al., 2017] is often adopted for the training in this approach [Benaim and Wolf, 2017, Hoffman et al., 2018, Murez et al., 2018, Li et al., 2019]. Here, a source-to-target translation model $G_{\mathrm{S}\to\mathrm{T}}$ is jointly trained with a target-to-source translation model $G_{\mathrm{T}\to\mathrm{S}}$. Given these two models, cycle consistency regularization minimizes the reconstruction error between original source data x^{S} and $G_{\mathrm{T}\to\mathrm{S}}(G_{\mathrm{S}\to\mathrm{T}}(x^{\mathrm{S}}))$ (and same for target data). This regularization enforces translated data to be informative as the original data, and it means that translated data is to be discriminative at the same level of corresponding original data, which is a suitable property for domain adaptation.

2.3 Domain adaptation under harder situations

In this section, we introduce several variants of domain adaptation that assume harder situations than closed-set domain adaptation. As shown in Table 2.1, the harder situations tackled in the literature can be categorized into two types: scarcity of data and a large domain-gap. The former one is dealt with in few-shot domain adaptation, zero-shot domain adaptation, and source-free adaptation, while the latter one is tackled in partial domain adaptation and open-set domain adaptation. In the following

Table 2.1: Domain adaptation under harder situations.

Cause of difficulty	Problem setting	Situation
Scarcity of data	Few-shot domain adaptation	$0 < \lvert\mathcal{D}_\mathrm{T}\rvert \ll \lvert\mathcal{D}_\mathrm{S}\rvert$.
	Zero-shot domain adaptation	$\lvert\mathcal{D}_\mathrm{T}\rvert = 0$.
	Source-free domain adaptation	$\lvert\mathcal{D}_\mathrm{S}\rvert = 0$.
Large domain-gap	Partial domain adaptation	$\mathcal{C}_\mathrm{S} \supset \mathcal{C}_\mathrm{T}$.
	Open-set domain adaptation	$\mathcal{C}_\mathrm{S} \subset \mathcal{C}_\mathrm{T}$.

subsections, we describe each problem setting and introduce representative studies.

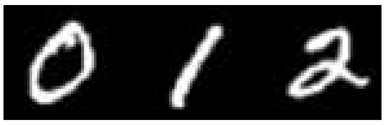
2.3.1 Few-shot domain adaptation

Due to the motivation of domain adaptation, the amount of target data used in domain adaptation is often small in practice. Few-shot domain adaptation methods [Motiian et al., 2017, Xu et al., 2019, Teshima et al., 2020] are specifically designed to be used in such cases. In the few-shot setting, target data contains only a few or few tens of samples in general. Consequently, simple closed-set domain adaptation methods introduced in the previous section would fail, because any empirical approximation over the target distribution, which is necessary to be done in these methods, is substantially inaccurate due to the scarcity of target data. How to tackle this issue is the most important challenge in few-shot domain adaptation.

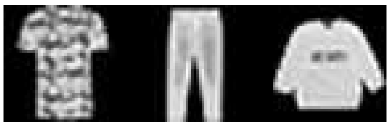
Motiian et al. [2017] proposed an adversarial-training based method in which the domain discriminator is extended to take a pair of data as an input. This discriminator classifies the input pair into four classes by discriminating from the following two aspects: (1) whether samples of a pair come from the source domain or the target domain, and (2) whether samples of a pair belong to the same class or not. Since target data are always used accompanied with randomly selected source data, the discriminator tends to avoid overfitting to target data. Xu et al. [2019] adopted a similar idea but used a pair of data to compute distributional discrepancy between domains instead of adversarial training. While these two methods augmented few target data by pairing them with source data, Teshima et al. [2020] utilized nonlinear Independent Component Analysis (ICA) [Hyvärinen and Pajunen, 1999] on multiple source-domain datasets to augment target data by manipulating independent components.

2.3.2 Zero-shot domain adaptation

Zero-shot domain adaptation is an extreme case of domain adaptation with scarce data in which no target data is available during adaptation. If we do not have any information about the target domain, zero-shot domain adaptation is obviously impossible. Consequently, zero-shot domain adaptation methods [Yang and Hospedales, 2015, Peng et al., 2018, Wang and Jiang, 2019] require another kind of information on the target domain instead of target data.


(a) Source-domain task-relevant data.

(b) Target-domain task-relevant data. (not available during adaptation)


(c) Source-domain task-irrelevant data.

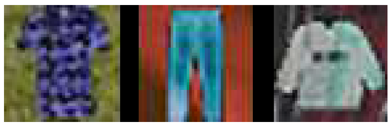

(d) Target-domain task-irrelevant data.

Figure 2.4: Examples of task-relevant data and task-irrelevant data used in Peng et al. [2018].

Peng et al. [2018] assumed the availability of both source-domain and target-domain data for another task, called task-irrelevant data. Figure 2.4 shows examples of task-irrelevant data as well as task-relevant data. In this example, source-domain and target-domain data are gray-scale and colored images, respectively, and the goal is to train a model that can accurately classify colored digit images. Even though the task-irrelevant data in both domains are not digit images, they should be informative to learn how to recognize semantics in an image while ignoring whether the image is gray-scale or not. Therefore, in Peng et al. [2018], task-irrelevant data are used to learn domain-invariant feature representation via the discrepancy-estimation based approach. Wang and Jiang [2019] adopted a similar assumption but used those data to train generative models to synthesize target-domain task-relevant data.

Yang and Hospedales [2015] assumed that multiple source-domain datasets are given and also assumed that these source domains as well as the target domain are represented by the common domain indicators. For example, in their experiments with object recognition datasets, the indicators are the blur size and brightness of images in each domain. Before adaptation, a classifier is trained with each source-domain dataset, and its parameters are stored with the corresponding domain indicator. Then, the parameters of the classifier adapted to the target domain are estimated by a kernel regression method using the stored dataset.

2.3.3 Source-free domain adaptation

Source-free domain adaptation is also an extreme case of domain adaptation with scarce data, but its setting is symmetric to that of zero-shot domain adaptation: no source data is available during adaptation. Instead of source data, the model pretrained with source data is given, and the goal of source-free adaptation is to adapt the pretrained model to the target domain by training with unlabeled target data. In the pioneering work [Chidlovskii et al., 2016], this challenging problem was tackled by constraining the model to be linear, but recent studies [Kundu et al., 2020, Li et al., 2020, Liang et al., 2020a] have proposed source-free domain adaptation methods that can be applied to more complex models, specifically, deep neural networks.

Liang et al. [2020a] explicitly divided the pretrained model into two modules, called a feature encoder and a classifier, similarly to DANN shown in Section 2.2.4, and trained the target-specific feature encoder while fixing the classifier. To make the classifier work well with the target features, they jointly minimized two losses for this training. The first one is the information-maximization loss, which encourages the classifier's outputs to be always one-hot vectors and also to be diverse over target data. The second loss is the classification loss computed with pseudo labels of the target data that are estimated via clustering of the target features. By jointly minimizing these two losses, the target encoder is trained so that the fixed classifier can accurately classify the target features.

Kundu et al. [2020] adopted a similar architecture to Liang et al. [2020a], but it has three modules: a backbone model, a feature extractor, and a classifier. In the adaptation phase, only the feature extractor is tuned for the target domain by minimizing the entropy of the classifier's output. Li et al. [2020] took a generative approach, and they jointly trained the target model and the conditional GAN that is to generate annotated target data. In the training phase, the conditional GAN is updated to generate more easy-to-classify data for the target classifier, and the classifier is updated to more accurately classify the generated data, while it is constrained to be sufficiently close to the pretrained classifier.

2.3.4 Partial domain adaptation

When a large-scale source dataset is available, target data may contain only a subset of class candidates, while all class candidates appear in source data. Under this setting, partial domain adaptation methods aim to train a classifier that performs well in the target domain over the subset of class candidates appeared in the target data. If we could precisely extract source data corresponding to the classes appeared in the target data, standard domain adaptation methods would simply be applicable to the extracted source data and target data. However, this cannot be easily conducted due to lack of target labels.

To tackle this issue, partial domain adaptation methods [Cao et al., 2018b,a, Zhang et al., 2018, Liang et al., 2020b] often adopt instance weights to extract relevant source data and jointly optimize the weight as well as the adapted model. In Cao et al. [2018b] and Cao et al. [2018a], class-wise weights are adopted in the adversarial-training based method and are estimated by simply taking the average of the classifier's outputs over target data. Liang et al. [2020b] also adopted class-wise weights but used them accompanied with artificial noise on domain labels of source data, which works as an augmentation of target data, to stabilize the adaptation in the early stage of training. On the other hand, Zhang et al. [2018] utilized sample-wise weights as similar to instance-weight based domain adaptation shown in Section 2.2.2, and these weights are efficiently estimated by the domain discriminator's outputs.

2.3.5 Open-set domain adaptation

If an environment where target data are obtained is not properly controlled, target data may contain additional classes that do not appear in source data. Due to the unseen-class data, the distribution of target data cannot be essentially matched with that of source data, which leads to poor performance of standard domain adaptation methods. Open-set domain adaptation methods have been proposed to tackle such situations. Although the setting of open-set domain adaptation seems just a symmetric setting of partial domain adaptation, it has distinct difficulty due to no restriction on the additional unseen classes in target data, while additional classes are annotated in source data under the partial domain adaptation setting.

The open-set domain adaptation problem has been first tackled in Panareda Busto and Gall [2017]. They defined the $(|\mathcal{C}| + 1)$-th class label for unseen-class data and explicitly assigned a label for each target sample so that the class-wise distribution of target data is well aligned with that of source data. Once target labels are assigned, standard domain adaptation can be conducted by removing unseen-class data from target data, which results in obtaining better feature representation where distributional discrepancy between domains is further reduced. By iteratively conducting the label assignment and feature representation learning, both domain data are well matched without negative effect by unseen classes in target data. Saito et al. [2018] adopted a similar idea but jointly conducted label assignment and feature representation learning by extending the adversarial-training based approach. Baktashmotlagh et al. [2018] took a sequential approach: without assigning labels, they first learned factorized feature representation that comprises domain-shared one and target-private one, and target labels were assigned based on the norm of features in each space.

2.4 Summary

In this chapter, we reviewed a literature of domain adaptation. We first introduced closed-set domain adaptation as the most fundamental setting in the literature. After formulating this setting, we reviewed existing studies via categorizing them into four approaches: the instance-weighting based approach, discrepancy-estimation based approach, adversarial-training based approach, and generative approach. Then, we showed several variants of domain adaptation that tackle harder situations. Scarcity of data is tackled by few-shot domain adaptation, zero-shot domain adaptation, and source-free domain adaptation, while large domain-gap is dealt with by partial domain adaptation and open-set domain adaptation.

As shown in Section 2.2, closed-set domain adaptation methods basically aim to align distributions between domains in an input space or feature space during adaptation, which has shown good performance and has been also theoretically validated in the literature [Ben-David et al., 2010]. However, as we will indicate in the subsequent chapters, existing methods often gave up this most promising direction, when data is scarce, for example, in zero-shot domain adaptation or source-free domain adaptation. In this dissertation, we propose novel domain adaptation algorithms that are designed to appropriately conduct distributional alignment between domains even in such situations. Our contributions make it possible to conduct domain adaptation with scarce data in a theoretically grounded manner and also to achieve surprisingly good performance.

Chapter 3

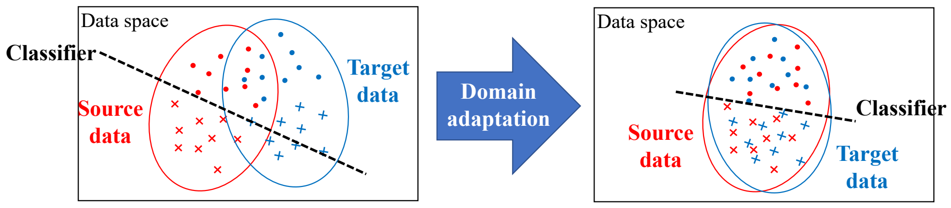
Partially Zero-shot Domain Adaptation from Incomplete Target Data with Missing Classes

In this chapter, we describe our first contribution to domain adaptation with scarce data. We first introduce a new problem setting, called partially zero-shot domain adaptation, which has not been explored in the literature but can appear in many practical situations. Then, we propose a novel adversarial-training based method that adopts instance weighting to tackle the partially zero-shot setting. Experimental results with several benchmark datasets validate the advantage of our method.

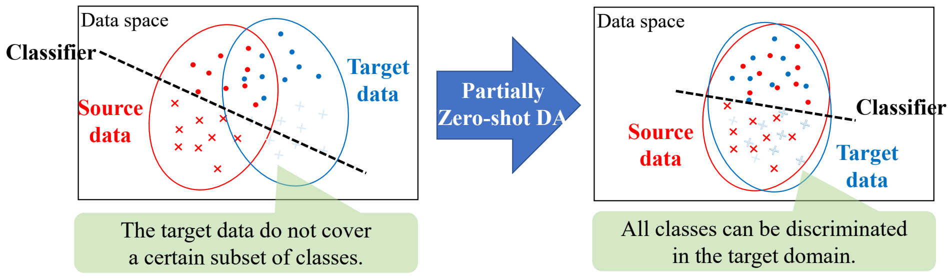
3.1 Introduction

The latest deep neural networks (DNNs) exhibit promising performance in various applications [Hatcher and Yu, 2018], such as image classification, audio recognition, and robotics. One of the key factors of their success is the availability of large-scale training datasets. Since a DNN has extremely high flexibility, a tremendous number of training data are required to utilize its full potential capability. However, obtaining such large-scale datasets may be typically hard in practical cases, therefore it has become a large obstacle to develop practical products or solutions using DNNs. To solve this problem, many researchers have worked on domain adaptation [Csurka, 2017, Wilson and Cook, 2020] that enables us to reduce the number of training data by transferring knowledge or data from other related domains.

As introduced in the previous chapter, standard domain adaptation methods basically aim to match data distributions between the source and target domains so that a classifier trained on the matched source data performs well in the target domain [Csurka, 2017] as shown in Fig. 3.1(a). Due to the motivation of the domain adaptation, the amount of target data is often small, which results in insufficient variation of target data to be matched with source data. A possible and important situation we want to focus on in this chapter is that, in classification tasks, unlabeled target data do not cover a certain subset of class candidates (we call them missing classes), while labeled source data include enough data from all classes. Differently from partial domain adaptation described in Section 2.3.4, although any target data from missing



(a) Standard closed-set domain adaptation.


Data space
Classifier
Source data
Target data
The target data do not cover a certain subset of classes.
Partially Zero-shot DA
Data space
Source data
Classifier
Target data
All classes can be discriminated in the target domain.

(b) Partially zero-shot domain adaptation.

Figure 3.1: Comparison of problem setting between standard closed-set domain adaptation and partially zero-shot domain adaptation.

classes are not observed, our goal is still to train a model that performs well in the target domain where target data can stem from all classes including missing classes as shown in Fig. 3.1(b). Therefore, we call this problem setting "partially zero-shot domain adaptation."

Suppose we want to detect anomalous behavior from surveillance videos by action recognition. Here, the action recognition is required to discriminate anomalous actions (e.g., fighting or falling down) as well as normal actions (e.g., walking or standing). When we have already constructed a training dataset for this recognition task at a certain surveillance camera, we can transfer it to a new camera at a different location by conducting the domain adaptation. In this case, unlabeled target data obtained by the new camera may not contain any anomalous actions, because such actions are rare. Although we have no annotations for the target data, we can guarantee that the target data do not contain the anomaly, if we know that any incident did not occur at the surveillance area while obtaining the target data. In this setting, standard closed-set domain adaptation methods would fail, because the source data that correspond to the missing classes do not have appropriate target data to be matched, and we cannot essentially match the source data with the target data as a whole. Generally, such a situation can appear if the class candidates have higher level groups (e.g., normal vs. anomalous actions) or a hierarchical structure. In that case, the target data may be only obtained at a subset of classes, and several classes are missing.

In this chapter, we tackle a domain adaptation problem under partially zero-shot setting. We adopt an adversarial-training based scheme that is widely used in the existing domain adaptation methods as described in Section 2.2.4 and extend it so that it can explicitly match the source data of the missing classes with the corresponding unseen target data. To this end, we consider the weighted loss of the given data instead of the loss of the unseen target data. By appropriately setting the instance weight based on the posterior estimated by DNNs, we can conduct accurate domain adaptation even though we do not have target data of the missing classes. Experimental results with several benchmark datasets validate the advantage of our method over both a standard domain adaptation method and a state-of-the-art partial domain adaptation method.

3.2 Partially zero-shot domain adaptation

In this section, we first formally define our partially zero-shot setting. After briefly reviewing a baseline method of domain adaptation, we clarify what problem will occur when we use the baseline method in that setting.

3.2.1 Definition of partially zero-shot setting

In our scenario, a certain class (or classes) does not appear in target data. Let $x \in \mathbb{R}^m, y \in \mathcal{C}$ and $d \in \{\mathrm{S}, \mathrm{T}\}$ denote input data, labels, and domains, respectively, where m is the dimensionality of the input data, $\mathcal{C}$ is the set of the class candidates, S represents the source domain, and T represents the target domain. For the moment, to make our formulation simple, we assume binary classification ($y \in \{\mathrm{P}, \mathrm{N}\}$: the positive class vs. the negative class). Note that we will extend our method to multi-class classification in Section 3.3.5 and that experiments will also be carried out for multi-class problems. Without loss of generality, we can consider that the negative class does not appear in target data. Specifically, in the partially zero-shot setting, given target data $\mathcal{D}_\mathrm{T}$ satisfy the following condition:

$$\mathcal{D}_\mathrm{T} = \{x_i\} \sim p(x|y = \mathrm{P}, d = \mathrm{T}). \tag{3.1}$$

This is the definition of the partially zero-shot setting for a binary classification task. Note that, in standard closed-set domain adaptation, $\mathcal{D}_\mathrm{T} = \{x_i\} \sim p(x|d = \mathrm{T})$. Importantly, in the partially zero-shot setting, $p(y = \mathrm{N}|d = \mathrm{T})$ is not necessarily equal to zero, which means that the missing class can appear in test target data, which are not available during adaptation.

A similar situation to ours was considered in several recent studies. The most related study is partial domain adaptation [Zhang et al., 2018, Cao et al., 2018a,b], and it considered the missing class in the source domain as an outlier class that we can ignore in classification. Our setting can also be considered as an extreme case of the domain adaptation with a class prior shift [Zhang et al., 2013, Chen et al., 2018] where the priors corresponding to the missing classes are set to zero in the target domain. In open-set domain adaptation [Panareda Busto and Gall, 2017, Saito et al., 2018, Baktashmotlagh et al., 2018], several unknown classes appear only in the target domain, while some classes are missing in the target domain in our setting. All the above existing works cannot solve our problem, because they cannot explicitly match the source data of missing classes with corresponding unseen target data, which results in poor performance of the classifier with the missing-class data. Since we want to discriminate all classes included in the source data in the target domain, we need to consider how to match the missing-class data between both domains.

3.2.2 Baseline method

As a baseline method, we chose Domain Adversarial Neural Networks (DANN) [Ganin et al., 2016], which is one of the most popular domain adaptation methods based on adversarial training. Since we have already introduced DANN in Section 2.2.4, we briefly review it here. In DANN, we jointly train feature extractor $g_f : \mathbb{R}^m \to \mathbb{R}^{m'}$, classifier $g_y : \mathbb{R}^{m'} \to C$, and domain discriminator $g_d : \mathbb{R}^{m'} \to \{\mathrm{S}, \mathrm{T}\}$ by the following optimizations:

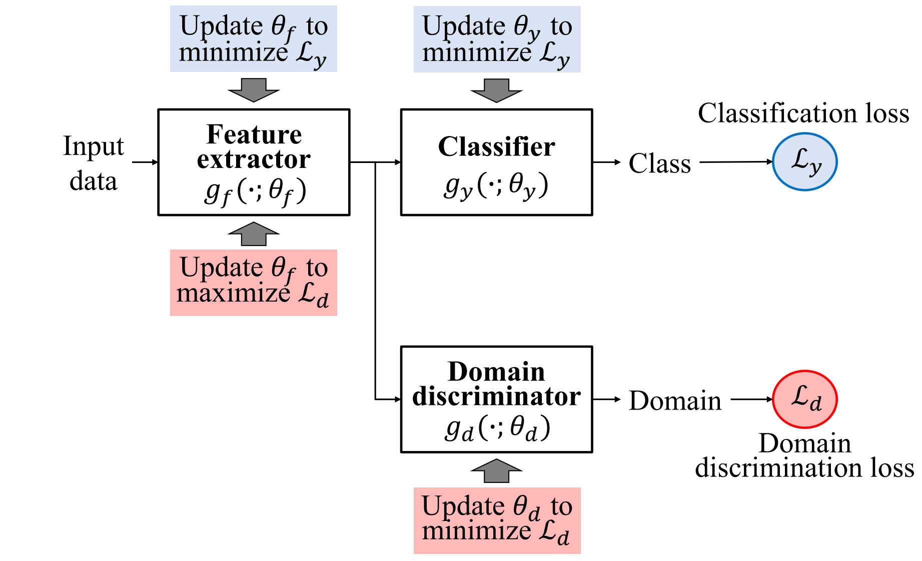
$$\begin{aligned} \mathcal{L}(\theta) &= \mathcal{L}_y(\theta) - \lambda \mathcal{L}_d(\theta), && (3.2)\\ \theta_d^* &= \arg\max_{\theta_d} \mathcal{L}(\theta), && (3.3)\\ \left(\theta_f^*, \theta_y^*\right) &= \arg\min_{\theta_f, \theta_y} \mathcal{L}(\theta), && (3.4) \end{aligned}$$

where $\theta = \{\theta_f, \theta_y, \theta_d\}$ is a set of trainable parameters of each network. $\mathcal{L}_y$ and $\mathcal{L}_d$ are classification loss and domain discrimination loss, respectively, and are defined as follows:

$$\begin{aligned} L_y(\theta) &= \sum_y \int l_y(x, y|\theta) p(x, y) \mathrm{d}x, && (3.5)\\ l_y(x, y|\theta) &= l_{\mathrm{cp}}(G_y(G_f(x; \theta_f); \theta_y), y), && (3.6)\\ L_d(\theta) &= \sum_d \int l_d(x, d|\theta) p(x, d) \mathrm{d}x, && (3.7)\\ l_d(x, d|\theta) &= l_{\mathrm{dp}}(G_d(G_f(x; \theta_f); \theta_d), d), && (3.8) \end{aligned}$$

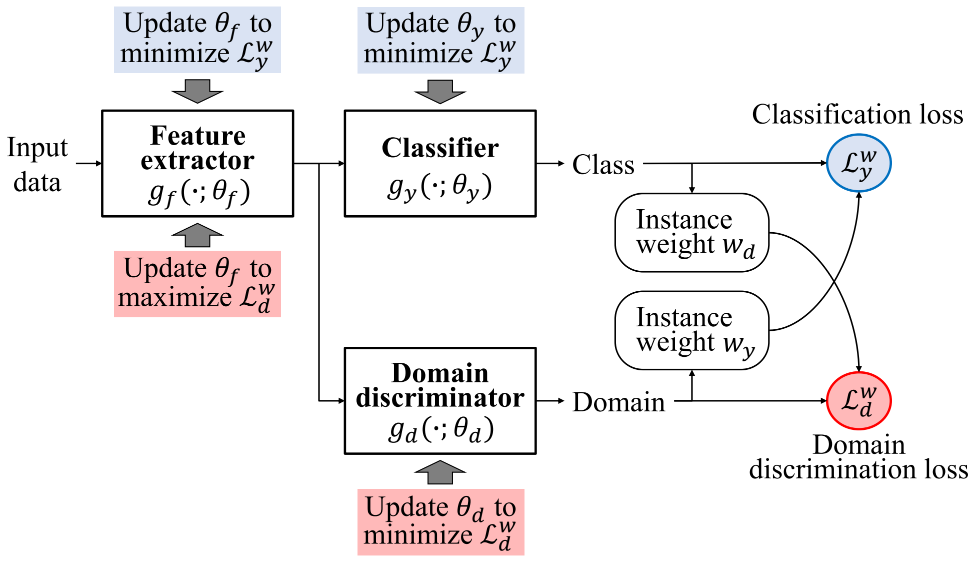
where l_{cp} and l_{dp} are loss functions for class prediction and domain discrimination, respectively. Note that, for ease of explanation of our method, we slightly change some notations from those in Section 2.2.4 by treating d as a random variable.

The optimization in Eq. (3.3) aims to discriminate domains of the input data by g_d, while that in Eq. (3.4) aims to fool g_d by g_f as well as accurately classifying the data by G_y. As a result, $g_f(x; \theta_f^*)$ becomes good feature embedding for domain adaptation, because it would be easy to classify as well as hard to discriminate domains.



Update θ_f to minimize $\mathcal{L}_y$
Update θ_y to minimize $\mathcal{L}_y$
Input data
Feature extractor $g_f(\cdot;\theta_f)$
Classifier $g_y(\cdot;\theta_y)$
Class
Classification loss
$\mathcal{L}_y$
Update θ_f to maximize $\mathcal{L}_d$
Domain discriminator $g_d(\cdot;\theta_d)$
Domain
$\mathcal{L}_d$
Domain discrimination loss
Update θ_d to minimize $\mathcal{L}_d$

(a) Domain adversarial neural networks [Ganin et al., 2016]



Update θ_f to minimize $\mathcal{L}_y^w$
Update θ_y to minimize $\mathcal{L}_y^w$
Input data
Feature extractor $g_f(\cdot;\theta_f)$
Classifier $g_y(\cdot;\theta_y)$
Class
Classification loss
$\mathcal{L}_y^w$
Instance weight w_d
Instance weight w_y
Update θ_f to maximize $\mathcal{L}_d^w$
Domain discriminator $g_d(\cdot;\theta_d)$
Domain
$\mathcal{L}_d^w$
Domain discrimination loss
Update θ_d to minimize $\mathcal{L}_d^w$

(b) Proposed method

Figure 3.2: Comparison of the architecture for domain adaptation between (a) the baseline method and (b) the proposed method.

3.2.3 The problem of the baseline method under the partially zero-shot setting

To clarify the problem of the baseline method in our scenario, we rewrite $\mathcal{L}_d$ in Eq. (3.7) as follows:

$$\begin{aligned}
\mathcal{L}_d(\theta) &= \sum_d \int l_d(x, d; \theta) \left(\sum_y p(x, y, d) \right) \mathrm{d}x \\
&= \pi_{\mathrm{PS}} \int l_d(x, \mathrm{S}; \theta) p(x|y = \mathrm{P}, d = \mathrm{S}) \mathrm{d}x \\
&\quad + \pi_{\mathrm{NS}} \int l_d(x, \mathrm{S}; \theta) p(x|y = \mathrm{N}, d = \mathrm{S}) \mathrm{d}x \\
&\quad + \pi_{\mathrm{PT}} \int l_d(x, \mathrm{T}; \theta) p(x|y = \mathrm{P}, d = \mathrm{T}) \mathrm{d}x \\
&\quad + \pi_{\mathrm{NT}} \int l_d(x, \mathrm{T}; \theta) p(x|y = \mathrm{N}, d = \mathrm{T}) \mathrm{d}x, \qquad (3.9)
\end{aligned}$$

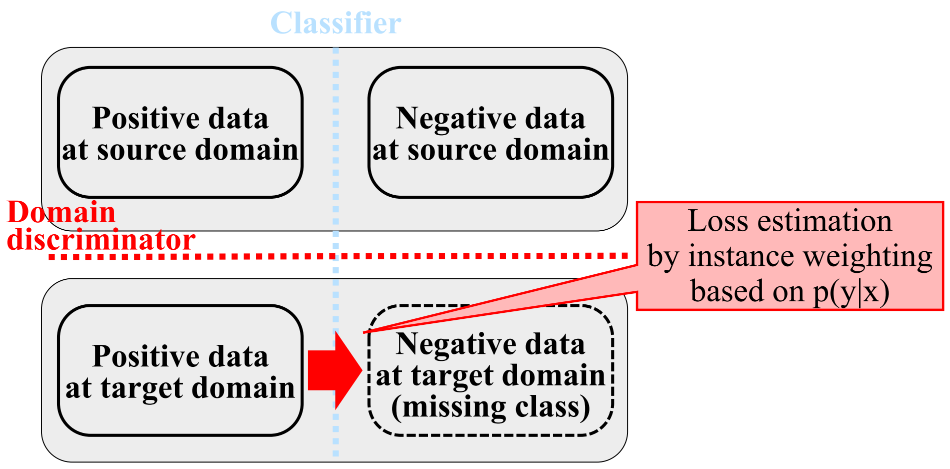
where $\pi_{\cdot *} = p(y = \cdot, d = *)$. These four terms correspond to the domain discrimination losses for the source positive data, source negative data, target positive data, and target negative data, respectively. Since we do not have any target negative data, we cannot calculate the fourth term. If we ignore the fourth term by setting π_{NT} to be zero, the negative data only appear in the second term. It means that, if $g_f(x)$ is easy to classify in terms of class prediction, g_d can easily discriminate the domain of the negative data by just predicting as the source domain. Consequently, to fool g_d, we need to make $g_f(x)$ hard to classify, which results in poor classification performance.

3.3 Proposed method

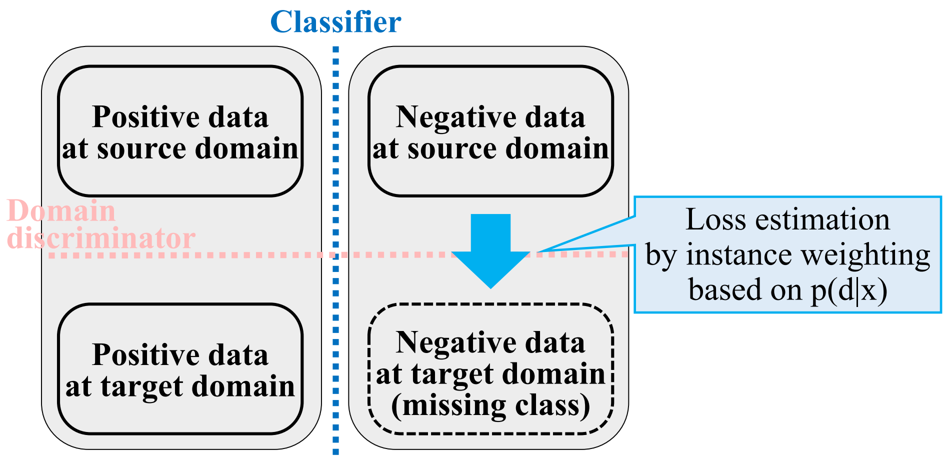
In this section, we propose a novel method for partially zero-shot domain adaptation.

3.3.1 Overview

Following the scheme of DANN, we formulate our partially zero-shot domain adaptation method by taking the adversarial-training based approach. A key trick of how to avoid the problem described in Section 3.2.3 is instance weighting based on the class posterior $p(y|x)$ as shown in Fig. 3.3(a), which makes it possible to calculate the loss related to unseen missing-class target data. By using outputs of the classifier, we can efficiently estimate the class posterior and consequently compute the instance weights. The same trick is also adopted for the classification loss, which results in instance weighting based on the domain posterior $p(d|x)$ as shown in Fig. 3.3(b). Finally, we formulate the overall optimization for our domain adaptation. We focus on binary classification to make this formulation simple but will extend it to multi-class classification later.


Classifier
Positive data at source domain
Negative data at source domain
Domain discriminator
Loss estimation by instance weighting based on p(y|x)
Positive data at target domain
Negative data at target domain (missing class)

(a) Calculation of the domain discrimination loss.


Classifier
Positive data at source domain
Negative data at source domain
Domain discriminator
Loss estimation by instance weighting based on p(d|x)
Positive data at target domain
Negative data at target domain (missing class)

(b) Calculation of the classification loss.

Figure 3.3: Loss calculation in the proposed method under a binary classification setting.

3.3.2 Instance weighting for domain discrimination loss

To avoid the problem mentioned in Section 3.2.3, we need to calculate the fourth term in Eq. (3.9) without the target negative data. Inspired by positive confidence learning [Ishida et al., 2018], we estimate the value of that term based on the target positive data by adopting instance weights as

$$\begin{aligned}
&\int l_d(x, \mathrm{T}; \theta) p(x|y = \mathrm{N}, d = \mathrm{T}) \mathrm{d}x \\
= &\int w_d(x) l_d(x, \mathrm{T}; \theta) p(x|y = \mathrm{P}, d = \mathrm{T}) \mathrm{d}x,
\end{aligned} \tag{3.10}$$

where $w_d(x)$ is the instance weight that can be defined as

$$w_d(x) \;=\; \frac{p(x|y = \mathrm{N}, d = \mathrm{T})}{p(x|y = \mathrm{P}, d = \mathrm{T})}. \tag{3.11}$$

By using Bayes' theorem and assuming the covariate shift condition [Shimodaira, 2000], $p(y|x, d = \mathrm{S}) = p(y|x, d = \mathrm{T}) = p(y|x)$, we can rewrite the weight as follows:

$$\begin{aligned}
w_d(x) &= \frac{p(y = \mathrm{N}|x, d = \mathrm{T}) p(d = \mathrm{T}|x) p(x)}{p(y = \mathrm{N}, d = \mathrm{T})} \\
&\quad \cdot \frac{p(y = \mathrm{P}, d = \mathrm{T})}{p(y = \mathrm{P}|x, d = \mathrm{T}) p(d = \mathrm{T}|x) p(x)} \\
&= \frac{p(y = \mathrm{P}, d = \mathrm{T})}{p(y = \mathrm{N}, d = \mathrm{T})} \cdot \frac{p(y = \mathrm{N}|x, d = \mathrm{T})}{p(y = \mathrm{P}|x, d = \mathrm{T})} \\
&= \frac{\pi_{\mathrm{PT}}}{\pi_{\mathrm{NT}}} \cdot \frac{p(y = \mathrm{N}|x)}{p(y = \mathrm{P}|x)}.
\end{aligned} \tag{3.12}$$

If we empirically estimate π_{NT}, it should be zero and the weight diverges. Therefore, we regard π_{NT} and π_{PT} as hyper-parameters in our method. In this paper, we set them as $\pi_{\mathrm{NT}} = \pi_{\mathrm{NS}}$ and $\pi_{\mathrm{PT}} = \pi_{\mathrm{PS}}$. This setting corresponds to the assumption where $p(d = \mathrm{S})$ is equal to $p(d = \mathrm{T})$ and the class prior does not change between the source and target domains. Based on this assumption, Eq. (3.12) can be rewritten as

$$\begin{aligned}
w_d(x) &= \frac{\pi_{\mathrm{PS}}}{\pi_{\mathrm{NS}}} \cdot \frac{p(y = \mathrm{N}|x)}{p(y = \mathrm{P}|x)} \\
&= \frac{p(y = \mathrm{P}|d = \mathrm{S})}{p(y = \mathrm{N}|d = \mathrm{S})} \cdot \frac{p(y = \mathrm{N}|x)}{p(y = \mathrm{P}|x)}.
\end{aligned} \tag{3.13}$$

In Eq. (3.13), $p(y|d = \mathrm{S})$ is the class prior of the source data, therefore, it can be easily estimated by using the source data. On the other hand, $p(y|x)$ is the class posterior, and how to obtain it is not a trivial problem. In [Ishida et al., 2018], the posterior is assumed to be given, but it is not a reasonable assumption in domain adaptation problems, so we need to estimate it. Fortunately, in DANN, we jointly train the classifier as well as the domain discriminator, and the output of the classifier is a probability distribution over the class candidates, which can be used as the class posterior like in [Zhang et al., 2018]. We use the output of the classifier instead of the class posterior to calculate the instance weight in Eq. (3.13).

By substituting Eq. (3.13) into Eq. (3.10) and using it instead of the fourth term in Eq. (3.9), we can obtain a new formulation of the domain discrimination loss $\mathcal{L}_d^w$

that can be calculated without the target negative data as

$$\begin{aligned}\mathcal{L}_d^w(\theta) &= \pi_{\mathrm{PS}} \int l_d(x, \mathrm{S}; \theta) p_{\mathrm{PS}}(x) \mathrm{d}x \\ &+ \pi_{\mathrm{NS}} \int l_d(x, \mathrm{S}; \theta) p_{\mathrm{NS}}(x) \mathrm{d}x \\ &+ \pi_{\mathrm{PS}} \int \left(1 + \frac{\pi_{\mathrm{NS}}}{\pi_{\mathrm{PS}}} \hat{w}_d(x)\right) l_d(x, \mathrm{T}; \theta) p_{\mathrm{PT}}(x) \mathrm{d}x, \qquad (3.14)\end{aligned}$$

where $\hat{w}_d(x)$ is the instance weight in Eq. (3.13) calculated by the output of the classifier, and $p_{\cdot *}(x)$ represents $p(x|y = \cdot, d = *)$.

When conducting adaptation, we compute the domain discrimination loss in Eq. (3.14) by empirically approximating it with given source and target data as the following equation:

$$\begin{aligned}\hat{\mathcal{L}}_d^w(\theta) &= \frac{1}{2|\mathcal{D}_{\mathrm{S}}|} \sum_{(x_i^{\mathrm{S}}, y_i^{\mathrm{S}}) \in \mathcal{D}_{\mathrm{S}}} l_d(x_i^{\mathrm{S}}, \mathrm{S}; \theta) \\ &+ \frac{n_{\mathrm{PS}}}{2|\mathcal{D}_{\mathrm{S}}||\mathcal{D}_{\mathrm{T}}|} \sum_{x_i^{\mathrm{T}} \in \mathcal{D}_{\mathrm{T}}} \left(1 + \frac{n_{\mathrm{NS}}}{n_{\mathrm{PS}}} \hat{w}_d(x_i^{\mathrm{T}})\right) l_d(x_i^{\mathrm{T}}, \mathrm{T}; \theta), \qquad (3.15)\end{aligned}$$

where n_{PS} and n_{NS} are the number of the source positive data and that of the source negative data, respectively.

3.3.3 Instance weighting for classification loss

Since we assume binary classification, we can know that the target data are all positive data, which should be useful information for training the classifier. However, if we use the target data to calculate the classification loss L_y, the similar problem will happen as we have previously shown in the case of the domain discrimination loss. As in Eq. (3.9), we can decompose $\mathcal{L}_y$ in Eq. (3.5) as follows:

$$\begin{aligned}\mathcal{L}_y(\theta) &= \pi_{\mathrm{PS}} \int l_y(x, \mathrm{P}; \theta) p_{\mathrm{PS}}(x) \mathrm{d}x \\ &+ \pi_{\mathrm{NS}} \int l_y(x, \mathrm{N}; \theta) p_{\mathrm{NS}}(x) \mathrm{d}x \\ &+ \pi_{\mathrm{PT}} \int l_y(x, \mathrm{P}; \theta) p_{\mathrm{PT}}(x) \mathrm{d}x \\ &+ \pi_{\mathrm{NT}} \int l_y(x, \mathrm{N}; \theta) p_{\mathrm{NT}}(x) \mathrm{d}x. \qquad (3.16)\end{aligned}$$

Again, we cannot calculate the fourth term in Eq. (3.16). Compared with the previous case, it seems not so problematic, because we do not fool the classifier. Even though the target data can be easily classified into the negative class in this case, it does not directly affect the training of DANN. However, since it leads to over-estimation of the performance of the classifier, it would disturb the training of the classifier. Therefore, we also adopt the instance weight to estimate the loss for the target negative data. In this case, we estimate the loss value based on the source negative data as

$$\int l_y(x, \mathrm{N}; \theta) p_{\mathrm{NT}}(x) \mathrm{d}x = \int w_y(x) l_y(x, \mathrm{N}; \theta) p_{\mathrm{NS}}(x) \mathrm{d}x, \tag{3.17}$$

where $w_y(x)$ is the instance weight that can be defined in a similar manner to Eq. (3.12), which results in

$$w_y(x) = \frac{\pi_{\mathrm{NS}}}{\pi_{\mathrm{NT}}} \cdot \frac{p(d = \mathrm{T}|x)}{p(d = \mathrm{S}|x)} = \frac{p(d = \mathrm{T}|x)}{p(d = \mathrm{S}|x)}. \tag{3.18}$$

Since we have already assumed $\pi_{\mathrm{NS}} = \pi_{\mathrm{NT}}$, $\frac{\pi_{\mathrm{NS}}}{\pi_{\mathrm{NT}}}$ was eliminated. Unlike the case of the domain discrimination loss, we need the domain posterior $p(d|x)$ instead of the class posterior to calculate the instance weights. Similarly to the class posterior, we use the output of the domain discriminator as an estimate of the domain posterior. By substituting Eq. (3.18) into Eq. (3.17) and using it instead of the fourth term in Eq. (3.16), we obtain a new formulation of the classification loss $\mathcal{L}_y^w$ that can be calculated without the target negative data:

$$\begin{aligned}
\mathcal{L}_y^w(\theta) &= \pi_{\mathrm{PS}} \int l_y(x, \mathrm{P}; \theta) p_{\mathrm{PS}}(x) \mathrm{d}x \\
&+ \pi_{\mathrm{NS}} \int \left(1 + \hat{w}_y(x)\right) l_y(x, \mathrm{N}; \theta) p_{\mathrm{NS}}(x) \mathrm{d}x \\
&+ \pi_{\mathrm{PS}} \int l_y(x, \mathrm{P}; \theta) p_{\mathrm{PT}}(x) \mathrm{d}x,
\end{aligned} \tag{3.19}$$

where $\hat{w}_y(x)$ represents the instance weight in Eq. (3.18) calculated by the output of the domain discriminator.

When conducting adaptation, we also empirically approximate the classification loss in Eq. (3.19) to compute it with given source and target data as shown below:

$$\begin{aligned}
\hat{\mathcal{L}}_y^w(\theta) &= \frac{1}{2|\mathcal{D}_{\mathrm{S}}|} \sum_{(x_i^{\mathrm{S}}, \mathrm{P}) \in \mathcal{D}_{\mathrm{S}}} l_y(x_i^{\mathrm{S}}, \mathrm{P}; \theta) \\
&+ \frac{1}{2|\mathcal{D}_{\mathrm{S}}|} \sum_{(x_i^{\mathrm{S}}, \mathrm{N}) \in \mathcal{D}_{\mathrm{S}}} \left(1 + \hat{w}_y(x_i^{\mathrm{S}})\right) l_y(x_i^{\mathrm{S}}, \mathrm{N}; \theta) \\
&+ \frac{n_{\mathrm{PS}}}{2|\mathcal{D}_{\mathrm{S}}||\mathcal{D}_{\mathrm{T}}|} \sum_{x_i^{\mathrm{T}} \in \mathcal{D}_{\mathrm{T}}} l_y(x_i^{\mathrm{T}}, \mathrm{P}; \theta).
\end{aligned} \tag{3.20}$$

3.3.4 Adversarial training with instance weights

Since we have already derived a new domain discrimination loss and a classification loss that can be calculated without the target negative data, we can train DNNs based on these losses for the partially zero-shot domain adaptation. In a similar manner with DANN, we formulate our domain adaptation as follows:

$$\mathcal{L}^w(\theta) = \mathcal{L}_y^w(\theta) - \lambda \mathcal{L}_d^w(\theta), \tag{3.21}$$

$$\theta_d^* = \arg\max_{\theta_d} \mathcal{L}^w(\theta), \tag{3.22}$$

$$\left(\theta_f^*, \theta_y^*\right) = \arg\min_{\theta_f, \theta_y} \mathcal{L}^w(\theta). \tag{3.23}$$

The above optimization seems to be the same as that of the baseline method, but, by adopting appropriate instance weights, we can accurately conduct domain adaptation, even though we do not have any target negative data.

3.3.5 Extension to multi-class classification

In the case of binary classification, only one of the classes does not appear in the target data. On the other hand, in the case of multi-class classification, multiple classes can disappear in the target data. Let A and M denote a set of classes that appear in the target data and a set of classes that do not appear, respectively. By using $y \in A$ instead of $y = \mathrm{P}$ and $y \in M$ instead of $y = \mathrm{N}$, we can derive our method under multi-class classification setting in a similar manner to what we have shown in the previous subsection. For the domain discrimination loss, the instance weight in Eq. (3.13) is formulated as

$$w_d(x) = \frac{p(y \in A|d = \mathrm{S})}{p(y \in M|d = \mathrm{S})} \cdot \frac{p(y \in M|x)}{p(y \in A|x)}. \tag{3.24}$$

Using this weight, we can derive the domain discrimination loss for multi-class classification setting as

$$\begin{aligned}\mathcal{L}_d^w(\theta) &= \pi_{\mathrm{AS}} \int l_d(x, \mathrm{S}; \theta) p_{\mathrm{AS}}(x) \mathrm{d}x \\ &\quad + \pi_{\mathrm{MS}} \int l_d(x, \mathrm{S}; \theta) p_{\mathrm{MS}}(x) \mathrm{d}x \\ &\quad + \pi_{\mathrm{AS}} \int \left(1 + \frac{\pi_{\mathrm{MS}}}{\pi_{\mathrm{AS}}} \hat{w}_d(x)\right) l_d(x, \mathrm{T}; \theta) p_{\mathrm{AT}}(x) \mathrm{d}x, \end{aligned} \tag{3.25}$$

where $\pi_{\cdot *} = p(y \in \cdot, d = *)$, $p_{\cdot *}(x) = p(x|y \in \cdot, d = *)$, and $\hat{w}_d(x)$ is the instance weight in Eq. (3.24) calculated by the output of the classifier. Since l_d does not depend on y, how to calculate the domain discrimination loss is almost the same as that for binary classification setting. On the other hand, in the case of the classification loss, it becomes somewhat different. The instance weight for the classification loss in Eq. (3.18) stays same, because it does not depend on y. However, the classification loss is changed to

$$\begin{aligned}\mathcal{L}_y^w(\theta) &= \pi_{\mathrm{AS}} \sum_{i \in A} \int l_y(x, i; \theta) p_{i\mathrm{S}}(x) \mathrm{d}x \\ &\quad + \pi_{\mathrm{MS}} \sum_{j \in M} \int (1 + \hat{w}_y(x))\, l_y(x, j; \theta) p_{j\mathrm{S}}(x) \mathrm{d}x \\ &\quad + \pi_{\mathrm{AS}} \int l_y(x, A; \theta) p_{\mathrm{AT}}(x) \mathrm{d}x. \end{aligned} \tag{3.26}$$

For the first and second terms in the right-hand side of Eq. (3.26), we can easily calculate l_y, because the labels are provided in the source data. However, for the third term, since there are no labels in the target data, we cannot calculate l_y in the same way as that in the first and second terms. Considering that l_y is the cross entropy loss, we adopt simple extension of the loss function as

$$l_y(x, A; \theta) = -\log \sum_{y \in A} \hat{p}(y|x; \theta), \tag{3.27}$$

where $\hat{p}(y|x;\theta)$ is the output of the classifier when the input is x and the parameters of DNNs are set to θ.

3.3.6 Implementation details

Our partially zero-shot domain adaptation adopts the instance weight to calculate the losses, and the weight is calculated by taking the ratio between the outputs of the classifier or those of the domain discriminator as shown in Eqs. (3.13) and (3.18), respectively. In the early stage of training DNNs, these outputs are often inaccurate, and this error would be magnified by taking the ratio, which results in quite inaccurate estimation of the weight that can severely damage the training process. Since evaluating how accurate the weights are is hard, we suppress the influence of the weight to alleviate this problem by two heuristic techniques. For simplicity, we explain these techniques for the binary classification case, but the same ones can be also applied in the multi-class classification case. First, to avoid extremely large weights, we clip the weight to be less than a certain upper bound:

$$\hat{w}_d(x) = \min\left[\alpha, \frac{p(y=\mathrm{P}|d=\mathrm{S})}{p(y=\mathrm{N}|d=\mathrm{S})} \cdot \frac{\hat{p}(y=\mathrm{N}|x;\theta)}{\hat{p}(y=\mathrm{P}|x;\theta)+\epsilon}\right], \tag{3.28}$$

$$\hat{w}_y(x) = \min\left[\alpha, \frac{\hat{p}(d=\mathrm{T}|x;\theta)}{\hat{p}(d=\mathrm{S}|x;\theta)+\epsilon}\right], \tag{3.29}$$

where α represents the upper bound of the weight. Here, a small positive constant ϵ is also introduced to avoid computational instability. In this paper, we set $\alpha = 5$ and $\epsilon = 0.01$, and they are fixed in all experiments. Second, to suppress the influence of the weighted losses, we reduce the contribution of the loss that stems from the target negative data by using $\beta\hat{w}_d(x)$ and $\beta\hat{w}_y(x)$ instead of $\hat{w}_d(x)$ and $\hat{w}_y(x)$ to calculate $\mathcal{L}_d^w$ in Eq. (3.14) and $\mathcal{L}_y^w$ in Eq. (3.19), respectively. The coefficient β ($0 < \beta \leq 1$) represents the confidence of the loss from the target negative data, and we set it to $1/\alpha$ in all experiments, which means that we allow the influence of the loss from the unseen target-negative data at most the same as that from the available target-positive data.

3.4 Experiments

In this section, we demonstrate the advantage of the proposed method through experiments with several benchmark datasets.

3.4.1 Setup

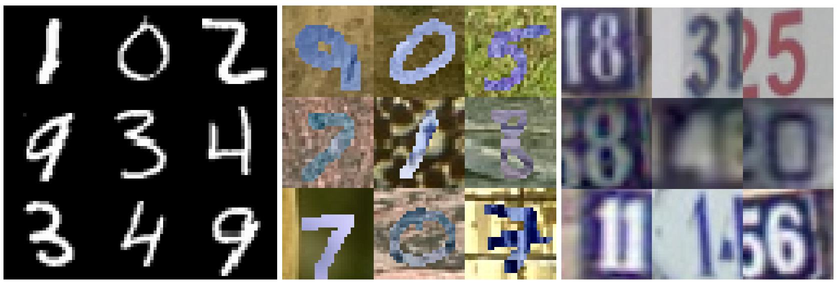
We conducted experiments with several datasets that are commonly used to benchmark domain adaptation methods in existing works. Specifically, we used digit classification datasets (MNIST [LeCun et al., 1998], MNIST-M [Ganin et al., 2016], and SVHN [Netzer et al., 2011]) and an object recognition dataset (Office-31 dataset [Saenko et al., 2010]). To make partially zero-shot setting, we chose a certain number of missing classes from the dataset with alphabetical order of class name and did not use corresponding target data while training. Note that the missing-class data do not appear in the training data of the target domain but are included in the test data. After training, we applied the trained classifier to the test data of the target domain and evaluated its accuracy to compare the performance of the domain adaptation methods.

For comparison, we used DNN trained with only source data (without domain adaptation) and DANN [Ganin et al., 2016] as baseline methods and Partial Adversarial Domain Adaptation (PADA) [Cao et al., 2018b] as a state-of-the-art partial domain adaptation method that are most related to our work. In PADA, the class-wise instance weight is set to be proportional to the average of the classifier predictions over the target data and is used when calculating the losses. This instance weighting enables one to ignore the missing-class data, because the missing class would be rarely predicted. Since PADA ignores the missing-class data in the classification loss as well as in the domain discrimination loss, it would result in poor classification performance in our scenario in which the missing class will appear in test data. Therefore, we also compared a variant of PADA that ignores the missing class only in the domain discrimination loss but does not ignore it in the classification loss. This method is referred to as the PADA-classifier in [Cao et al., 2018b]. For each method, we conducted experiments five times with random initialization of DNNs and will report their averaged accuracy.

Another possible choice of partial domain adaptation methods would be Importance Weighted Adversarial Nets (IWAN) [Zhang et al., 2018]. However, IWAN is not essentially suitable for partially zero-shot setting. Since IWAN adopts two separate feature extractors for each domain, the missing class data are never learned while training of the target-side feature extractor. It results in poor classification performance in our scenario, and we cannot easily avoid this problem. Therefore, we did not include IWAN in our experiment.

3.4.2 Digit image classification

MNIST, MNIST-M, and SVHN are digit image datasets, and the task is to classify these images into ten classes that correspond to digit numbers. Example images of each dataset are shown in Fig. 3.4. In the experiments, we tried two popular domain adaptation scenarios: from MNIST to MNIST-M and from SVHN to MNIST. Since MNIST-M images are made by artificially synthesizing MNIST images with



(a) MNIST (b) MNIST-M (c) SVHN

Figure 3.4: Example images of MNIST, MNIST-M, and SVHN.

Table 3.1: The network architecture used for the domain adaptation from MNIST to MNIST-M. MP2, BN, and FC denote 2×2 max-pooling, batch normalization, and a fully-connected layer, respectively.

Feature extractor	
Layer type	Filter size / # Filters
conv. + ReLU	5×5 / 32
MP2 + BN	2×2 / 32
conv. + ReLU	5×5 / 48
MP2 + BN	2×2 / 48

Classifier	
FC + ReLU	1 / 100
FC + ReLU	1 / 100
FC + softmax	1 / 10

Domain discriminator	
FC + ReLU	1 / 100
FC + sigmoid	1 / 1

background extracted from natural images, domain adaptation between MNIST and MNIST-M is relatively easy. On the other hand, SVHN and MNIST images are collected at totally different environment. Therefore, these images have largely different appearance to each other, which makes the domain adaptation harder. We almost followed the same setup for the experiments in [Ganin et al., 2016]. The network architectures are shown in Table 3.1 and 3.2, respectively. These networks are trained by the stochastic gradient descent with momentum. The size of mini-batch was set to 128. Its half is for the source data, and the rest is for the target data. The momentum was set to 0.9, and the learning rate was decayed while training as $\mu = \mu_0 / (1 + \alpha p)^{\beta}$, where p is the training progress linearly changing from 0 to 1, $\mu_0 = 0.01$, $\alpha = 10$, and $\beta = 0.75$. The hyper-parameter λ in Eq. (3.21) was set to 1, but, for training feature extractor, we adopt gradually changing λ as $\lambda = 2/(1 + \exp(-\gamma p)) - 1$, where γ was set to 10. For fair comparison, we used doubled value of the above equation as λ in our method to match the contribution of the original discrimination loss of the target

Table 3.2: The network architecture used for the domain adaptation from SVHN to MNIST. MP3s2, BN, and FC denote 3×3 max-pooling with stride 2, batch normalization, and a fully-connected layer, respectively.

Feature extractor	
Layer type	Filter size / # Filters
BN + conv. + ReLU	5×5 / 64
MP3s2 + BN	3×3 / 64
conv. + ReLU	5×5 / 64
MP3s2 + BN	3×3 / 64
conv. + ReLU + BN	5×5 / 128

Classifier	
FC + ReLU	1 / 500
FC + ReLU	1 / 500
FC + softmax	1 / 10

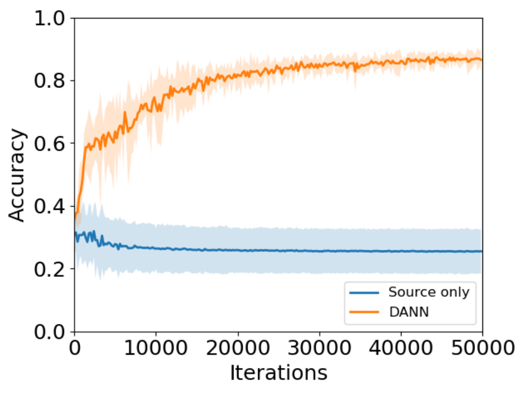
Domain discriminator	
FC + ReLU	1 / 100
FC + sigmoid	1 / 1

Table 3.3: Experimental results of domain adaptation from MNIST to MNIST-M.

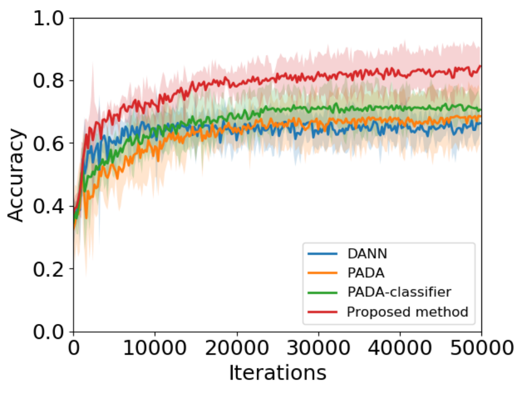
	The number of missing classes				
	0	3	5	7	9
Source only	26.7%	-	-	-	-
DANN	85.2%	64.9%	51.7%	32.6%	24.7%
PADA	-	65.5%	58.8%	50.0%	27.1%
PADA-classifier	-	70.1%	57.9%	56.4%	36.8%
Proposed method	-	**85.5%**	**85.4%**	**81.0%**	**52.2%**

data between our method and the other methods. For the partially zero-shot setting, we evaluated the accuracy with varying the number of missing classes.

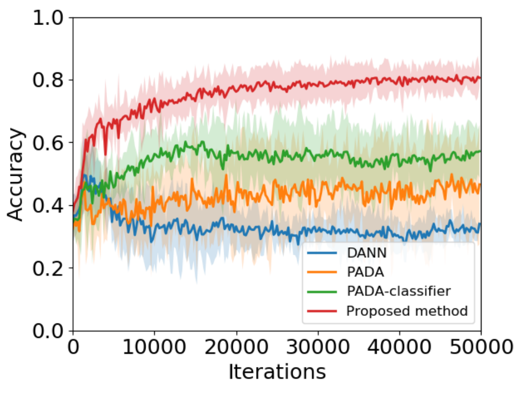
Table 3.3 shows the experimental results of the adaptation from MNIST to MNIST-M. When the number of the missing classes is zero, which corresponds to the standard setting of existing domain adaptation methods, DANN works well and substantially increases the accuracy from 26.7% to 85.2%. As increasing the number of the missing classes, the performance of DANN becomes deteriorated and even worse than that of source only case when nine classes disappear in the target data. This is because DANN hopelessly tries to match the distribution of the source data with that of the target data as a whole even though the source data of the missing class do not have appropriate data to be matched due to lack of the missing class in the target data. On the other hand, PADA and PADA-classifier actively ignore the missing class while domain adaptation, which results in better performance than DANN under the partially zero-shot setting. PADA-classifier performs better than PADA, because it does not ignore the missing-class data when calculating the classification loss. Com-



(a) No missing classes.



(b) Three classes are missing.



(c) Seven classes are missing.

Figure 3.5: Accuracy on the test data of the target domain while training adaptation from MNIST to MNIST-M. Solid line represents the averaged accuracy, and shaded area represents $\pm 2\sigma$.

pared with these methods, our method achieves much better performance, because it explicitly considers the loss of unseen target data of the missing classes for domain adaptation, while the other methods do not. Amazingly, the proposed method keeps almost same accuracy with the standard adaptation setting even when a half of classes do not appear in the target data.

The test accuracy while training is shown in Fig. 3.5. When there is no missing class, the test accuracy of DANN gradually increases while training and converges to a certain level in the end as shown in Fig. 3.5(a). On the other hand, in partially zero-shot setting, the accuracy of DANN does not substantially increase and even decrease when seven classes are missing. PADA works slightly better than DANN, but its accuracy fluctuates when the number of missing classes is large as shown in Fig. 3.5(c). This is because PADA tries to ignore the missing-class data while training, which results in instable accuracy on the missing classes in the test data. Compared

Table 3.4: Experimental results of domain adaptation from SVHN to MNIST.

	The number of missing classes				
	0	3	5	7	9
Source only	63.4%	-	-	-	-
DANN	67.8%	66.5%	67.6%	**66.7%**	62.5%
PADA	-	54.7%	58.3%	59.9%	49.0%
PADA-classifier	-	66.0%	66.6%	64.8%	**64.1%**
Proposed method	-	**67.2%**	**69.0%**	58.1%	62.2%

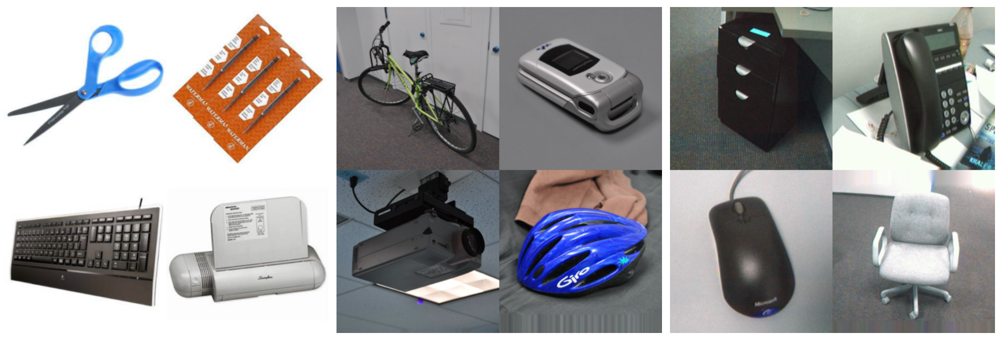

(a) Amazon (b) DSLR (c) Webcam

Figure 3.6: Example images of Office-31 dataset.

with PADA, the proposed method show better and more stable performance due to considering the missing class in the calculation of the losses. Specifically, the proposed method behaves almost same as DANN with no missing classes, even when seven classes are missing.

Table 3.4 shows the results of the adaptation from SVHN to MNIST. All methods except for our method behaved similar to those in the previous experiment, but our method gets relatively worse when the number of the missing classes becomes large. Considering that PADA-classifier that adopts the class posterior to calculate the loss works well, this is probably because the estimation error of the domain posterior becomes quite large due to the large discrepancy between SVHN and MNIST as well as a large amount of missing-class data. In addition, our assumption on the class prior in Eq. (3.13) is violated in this experiment. The number of training data is almost balanced between classes in MNIST, but it largely varies in SVHN, which results in the change of the class prior.

3.4.3 Object recognition

The Office-31 dataset is one of the most popular datasets specialized for benchmarking domain adaptation methods. This dataset contains object images with 31 categories, and three domains are defined: Amazon, DSLR, and webcam. Example im-

Table 3.5: Experimental results of domain adaptation from Amazon to webcam in Office-31 dataset.

	The number of missing classes		
	0	10	20
Source only	61.5%	-	-
DANN	70.1%	61.3%	52.4%
PADA	-	62.9%	31.6%
PADA-classifier	-	65.6%	55.3%
Proposed method	-	**68.2**%	**63.5**%

Table 3.6: Experimental results of domain adaptation from Amazon to DSLR in Office-31 dataset.

	The number of missing classes		
	0	10	20
Source only	66.3%	-	-
DANN	71.1%	62.2%	54.2%
PADA	-	64.2%	23.4%
PADA-classifier	-	67.9%	59.9%
Proposed method	-	**69.2**%	**70.0**%

ages in each domain are shown in Fig. 3.6. We can consider six scenarios of domain adaptation in this dataset, but we focus on the most difficult two scenarios that are from Amazon to webcam and from Amazon to DSLR. This is because DSLR and webcam are relatively similar to each other and the effect of the original DANN is marginal (e.g. $96.1\% \rightarrow 96.4\%$ in the case of DSLR-to-webcam reported in [Ganin et al., 2016]). We also follow the same experimental setup with [Ganin et al., 2016]. We used AlexNet pretained with ImageNet [Krizhevsky et al., 2012] to construct the initial feature extractor by removing the output layer and adding the 256-dimensional bottleneck. For the classifier and the domain discriminator, we used a single fully-connected layer ($256 \rightarrow 31$) and a fully-connected network with two hidden layers ($256 \rightarrow 1024 \rightarrow 1024 \rightarrow 1$). The learning rate for the pretrained layers is set to be ten times smaller than the other layers. The other setting is same as that in the previous experiment.

Table 3.5 and 3.6 show the results with Office-31 dataset. As in the previous experiment, the proposed method achieves substantially better performance than the other methods. Even when a large proportion of the classes, such as twenty out of thirty one, does not appear in the target data, our method achieves better performance than source only case and does not fall into negative transfer.

3.5 Conclusion

In this chapter, we introduced a new problem setting, called partially zero-shot domain adaptation, which has not been explored in the literature but can often appear in real-world applications. In this setting, a certain subset of classes only appear in the source data but do not appear in the target data, while we want to discriminate all classes at the target data after the domain adaptation. To tackle this problem, we proposed a novel domain adaptation method that utilizes both adversarial training and instance weighting. We derived how to estimate losses of unseen missing-class target data by adopting the instance weights that are estimated based on the outputs of DNNs. In the experiments, our method has shown excellent performance under partially zero-shot setting compared with existing domain adaptation methods.

Chapter 4

Zero-shot Domain Adaptation based on Attribute Information

In this chapter, we describe our second contribution to domain adaptation with scarce data. We propose a novel method for zero-shot domain adaptation. We consider the situation where domain shift is caused by change in the prior of a specific factor and assume that we know how the prior changes between source and target domains. We call this factor an attribute, and reformulate the domain adaptation problem to utilize the attribute prior instead of target data. In our method, source data are reweighted with the sample-wise weight estimated by the attribute prior and the data themselves so that they are useful in the target domain. We theoretically reveal that our method provides more precise estimation of sample-wise transferability than a straightforward attribute-based reweighting approach. Experimental results with both toy datasets and benchmark datasets show that our method can perform well, though it does not use any target data.

4.1 Introduction

In many algorithms for supervised learning, it is assumed that training data are obtained from the same distribution as that of test data [Hastie et al., 2009]. Unfortunately, this assumption is often violated in practical applications. For example, Fig. 4.1 shows images of two different surveillance videos that are obtained from Video Surveillance Online Repository [Vezzani and Cucchiara, 2010]. Suppose we want to recognize vehicles from these videos. Since the position and pose of the camera are different, the appearance of the vehicle is somewhat different between two videos. Due to this difference, even if we train a highly accurate classifier on video A, it may work poorly on video B. Such discrepancy has recently become a major problem in pattern recognition, because it is often difficult to obtain training data that are sufficiently similar to the test data. To deal with this problem, domain adaptation techniques have been proposed.

Given two datasets, called source and target data, domain adaptation aims to adapt source domain data to the target domain data so that distributions of both datasets are

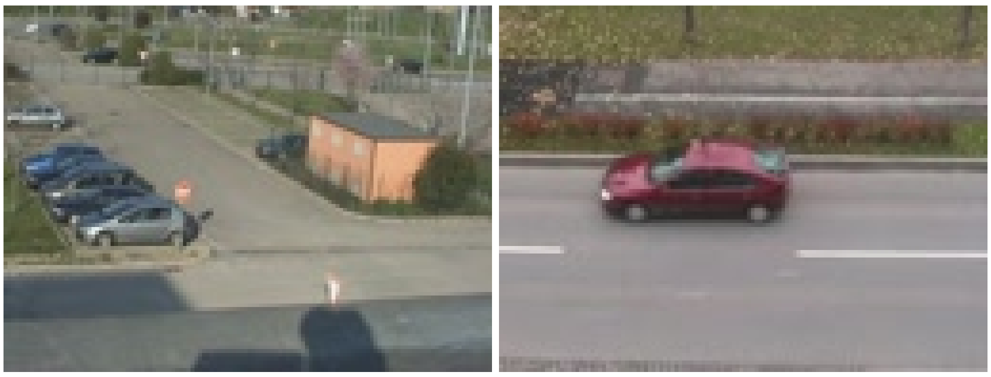

(a) Video A (b) Video B

Figure 4.1: Example images of surveillance videos. Since the position and pose of the surveillance camera is different, the appearance of the vehicle is somewhat different between two videos.

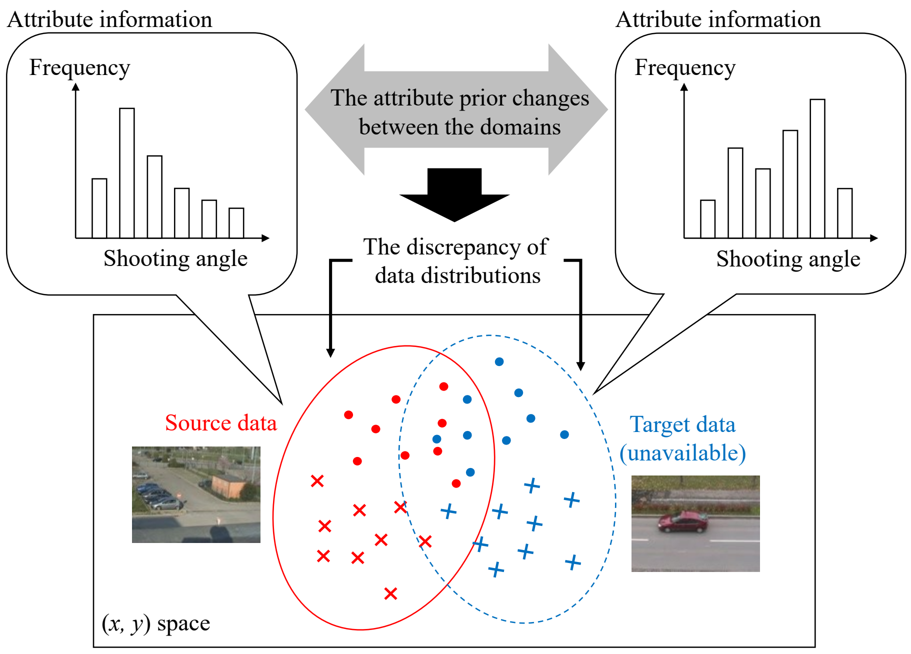


Figure 4.2: The situation we are considering in this work.

matched [Csurka, 2017]. By applying domain adaptation, classifiers trained on the adapted source data can achieve high accuracy on the target data. Since the discrepancy between two distributions is measured based on observed data, we need a sufficient number of data in each dataset to estimate the distributional discrepancy accurately. However, due to the motivation of the domain adaptation, obtaining a large number of target data is often hard, which limits the application of domain adaptation methods to practical cases.

In this chapter, we consider the most extreme case in which we cannot obtain any target data, called zero-shot domain adaptation. A few recent studies [Yang and Hospedales, 2015, Peng et al., 2018] have tackled this challenging problem, but they require additional data such as multiple source datasets [Yang and Hospedales, 2015] or target data of another task [Peng et al., 2018] that are not easy to obtain in practice. In this work, we propose a novel method of zero-shot domain adaptation that would be more suitable for practical cases. We assume that we have prior knowledge about what factor causes the difference in distributions between source and target data. For example, in Fig. 4.1, the shooting angle for vehicles can be considered as a major factor that causes the appearance change between videos. Other examples include gender information in an age estimation task from facial images and the azimuth of captured objects in an object recognition task, both of which are examined in our experiments.

We call such a factor an attribute, and assume that we can only obtain attribute priors in the target domain instead of the target data. We then reformulate the domain adaptation problem so that we can conduct adaptation based only on attribute priors. In addition, we clarify requirements for the attribute to be useful in domain adaptation, and reveal that our method provides more precise estimation of sample-wise transferability than the straightforward attribute-based reweighting approach. Experimental results with both toy datasets and benchmark datasets validate the advantage of our method, even though it does not use any target data.

We explain our setting by using vehicle recognition from surveillance videos as an example shown in Fig. 4.2. In this task, input data and labels are cropped video frames and vehicle types, respectively. Suppose that we have already constructed training datasets from existing surveillance cameras and want to transfer those datasets to a classifier for a new surveillance camera. If the new camera is not installed yet, we cannot obtain any target videos, therefore, we cannot apply a standard domain adaptation method nor evaluate how much data can be transfered via domain adaptation. But, if where and how the new camera will be installed have been already determined, we can estimate the shooting angle for the target vehicle. Since the shooting angle is a major factor that causes the appearance change of vehicles, we can consider the shooting angle as an attribute. In this case, we calculate it for each sample in the source domain and also estimate how often the vehicle will be captured with a certain shooting angle in the target domain by using the information about the pose and position of a camera. As shown in the above example, the assumption about attribute information in our method is sufficiently practical, and we believe that our method can be applied in many practical applications, especially for computer vision tasks.

4.2 Problem formulation and related works

Since our method adopts instance-weighting approach for adaptation, we briefly review how the existing instance-weighting based methods tackle standard closed-set domain adaptation. Let us consider a supervised classification task, and let $x \in \mathbb{R}^m, y \in \mathcal{C}$ and $d \in \{\mathrm{S}, \mathrm{T}\}$ denote input data, labels, and domains, respectively. Here, m is the dimensionality of the input data, $\mathcal{C}$ is the set of the class candidates, and $\{\mathrm{S}, \mathrm{T}\}$ represent the source and target domains, respectively. Note that we treat d as a random variable. We assume that the joint distributions of (x, y) are different between domains, which means $p(x, y|d = \mathrm{S}) \neq p(x, y|d = \mathrm{T})$. Given labeled source data $\mathcal{D}_S = \{(x_i^{\mathrm{S}}, y_i^{\mathrm{S}})\} \sim p(x, y|d = \mathrm{S})$ and unlabeled target data $\mathcal{D}_T = \{x_i^{\mathrm{T}}\} \sim p(x|d = \mathrm{T})$, our goal is to train a model $f : \mathbb{R}^m \to \mathcal{C}$ that can accurately predict labels for input data in the target domain. More specifically, supposing f is parameterized by θ, we want to obtain the optimal θ that minimizes the target risk defined as

$$R_{\mathrm{T}}(\theta) = \sum_{y \in \mathcal{C}} \int l(x, y, \theta) p(x, y|d = \mathrm{T}) \mathrm{d}x, \tag{4.1}$$

where $l(x, y, \theta)$ is a loss when y is predicted by f with θ at x.

Since the target data are not labeled, we cannot directly estimate the risk in Eq. (4.1) by empirical approximation. Instead, we try to use the source data to estimate it. The target risk can be related to the source risk with instance weights as:

$$R_{\mathrm{T}}(\theta) = \sum_{y \in \mathcal{C}} \int w(x, y) l(x, y, \theta) p(x, y|d = \mathrm{S}) \mathrm{d}x, \tag{4.2}$$

$$w(x, y) = \frac{p(x, y|d = \mathrm{T})}{p(x, y|d = \mathrm{S})} \tag{4.3}$$

where $w(x, y)$ is an instance weight for the corresponding data (x, y). By assuming covariate shift [Shimodaira, 2000], that means $p(y|x)$ is common in the source and target domains, we can simplify the weight as follows

$$\begin{aligned} \frac{p(x, y|d = \mathrm{T})}{p(x, y|d = \mathrm{S})} &= \frac{p(y|x, d = \mathrm{T})}{p(y|x, d = \mathrm{S})} \frac{p(x|d = \mathrm{T})}{p(x|d = \mathrm{S})} \\ &= \frac{p(x|d = \mathrm{T})}{p(x|d = \mathrm{S})} := w(x). \end{aligned} \tag{4.4}$$

The covariate shift assumption is intuitively reasonable in many pattern recognition tasks, so it is often adopted not only explicitly in the instance-weighting based methods but also implicitly in the recent adversarial-training based methods [Ganin et al., 2016, Tzeng et al., 2017] that aim to match $p(x|d)$ instead of $p(x, y|d)$ between the domains.

Equation (4.2) indicates that we can obtain the optimal θ by minimizing the weighted source risk. Therefore, many existing instance-weighting based methods [Huang et al., 2007, Sugiyama et al., 2008, Kanamori et al., 2009] basically try to accurately estimate the weight defined in Eq. (4.4). When we estimate the weight, we assume that the weight is always finite. Once we obtain the weight for each sample in the source data, we can calculate the empirically approximated risk $\hat{R}_{\mathrm{T}}(\theta)$ as:

$$\hat{R}_{\mathrm{T}}(\theta) = \frac{1}{|\mathcal{D}_S|} \sum_{(x_i, y_i) \in \mathcal{D}_S} \hat{w}(x_i) l(x_i, y_i, \theta), \tag{4.5}$$

where $\hat{w}(x_i)$ is the estimated weight for (x_i, y_i). By minimizing this empirical risk, we can estimate the optimal θ.

In our zero-shot scenario, the standard instance-weighting based approach cannot be directly adopted, because they require target data as well as source data to estimate the weight. Therefore, the main problem in our scenario is how to estimate the weight without target data. We will show that it can be solved by utilizing the attribute information instead of the unavailable target data.

As described in Section 2.3.2, a few recent studies [Yang and Hospedales, 2015, Peng et al., 2018] have tackled zero-shot domain adaptation problem, but they require additional data such as multiple source datasets [Yang and Hospedales, 2015] or target data of another task [Peng et al., 2018] that are not easy to obtain in practice. Moreover, they gave up explicitly aligning data distributions between domain. Therefore, their performance cannot be guaranteed by well-explored theories in the literature [Ben-David et al., 2010]. In this work, we explicitly conduct distributional alignment between domains using attribute information and also theoretically clarify what property the attribute should hold for effective domain adaptation.

In terms of utilizing attribute information, attribute-based zero-shot learning [Romera-Paredes and Torr, 2015] or few-shot learning [Li et al., 2006] is somewhat related to our work. However, there is a significant difference; the attribute information is utilized for representing an unseen " class " in zero-shot learning while it is used for representing an unseen " domain " in zero-shot domain adaptation. In this work, we establish the algorithm specialized for zero-shot domain adaptation and theoretically clarify the condition required for zero-shot domain adaptation.

4.3 Proposed method

In this section, we first describe our zero-shot domain adaptation method based on the attribute information. Then, we theoretically derive specific requirements for the attribute information to make our method work correctly. Lastly, we clarify some characteristics of our method and show the robustness against insufficient attribute information.

4.3.1 Our approach

We assume that we can obtain attribute information in both the source and target domains which contains information on the discrepancy between the data distributions. More specifically, in the source domain, attribute z for each sample is also given in addition to (x, y), and in the target domain, we cannot obtain any data or attributes as well, but only the probability distribution of attributes $p(z|d = \mathrm{T})$ is given. In our method, we try to estimate instance weights introduced in the previous section without target data but using attribute information. To make our formulation simple, we assume a single categorical attribute, but our method can be extended to multivariate or continuous attributes in a straightforward way.

4.3.2 Instance weight estimation based on attribute information

First, we transform the probability density ratio in Eq. (4.4). Since we do not have any information about the domain prior $p(d)$ especially for the target domain, we assumed a uniform distribution ($p(d = \mathrm{S}) = p(d = \mathrm{T})$) that is often used as a non-informative prior. By using this assumption and Bayes' theorem, we obtain the following equation:

$$w(x) = \frac{p(x|d=\mathrm{T})}{p(x|d=\mathrm{S})} = \frac{p(d=\mathrm{T}|x)}{p(d=\mathrm{S}|x)} \frac{p(d=\mathrm{S})}{p(d=\mathrm{T})} = \frac{p(d=\mathrm{T}|x)}{p(d=\mathrm{S}|x)}. \tag{4.6}$$

Then, based on the attribute information, we approximate $p(d|x)$ as follows:

$$p(d|x) \approx \sum_z p(d|z)p(z|x). \tag{4.7}$$

We will discuss what condition is required for the approximation in Eq. (4.7) in the next subsection. Substituting Eq. (4.7) into Eq. (4.6), we obtain

$$w(x) = \frac{\sum_z p(d = \mathrm{T}|z)p(z|x)}{\sum_z p(d = \mathrm{S}|z)p(z|x)}. \tag{4.8}$$

By adopting the approximation in Eq. (4.7), we can calculate $w(x)$ by estimating $p(d|z)$ and $p(z|x)$. It means that we do not need the target data, because $p(d|z)$ can be estimated from the given information about the attributes, and $p(z|x)$ that does not depend on domains can be estimated from the source data. This is the key trick of our method.

Since we assume that $p(z|d)$ is given and $p(d = \mathrm{S}) = p(d = \mathrm{T})$, $p(d|z)$ can be calculated by using Bayes' theorem as follows:

$$p(d = \mathrm{T}|z) = \frac{p(z|d = \mathrm{T})}{p(z|d = \mathrm{S}) + p(z|d = \mathrm{T})}, \tag{4.9}$$

$$p(d = \mathrm{S}|z) = \frac{p(z|d = \mathrm{S})}{p(z|d = \mathrm{S}) + p(z|d = \mathrm{T})}. \tag{4.10}$$

For the estimation of $p(z|x)$, we adopt the k-nearest neighbor method which is the simplest method for the posterior estimation: given x, we search k nearest samples

Algorithm 1 Zero-shot domain adaptation based on attribute information

```
Require: Source data $(x, y, z) \sim p(x, y, z|d = S)$ are given
Require: Target attribute information $p(z|d = T)$ is given
Require: Equation (4.7) and $p(d = S) = p(d = T)$ hold
  Calculate $p(d|z)$ by Eq. (4.9) and (4.10)
  Estimate $p(z|x)$ of the source data ($k$-NN method is used in this work)
  Estimate $w(x)$ of the source data by Eq. (4.8) using $p(d|z)$ and $p(z|x)$
  Obtain $\theta^*$ by minimizing the weighted source risk in Eq. (4.5)
  return $\theta^*$
```

from the source data and extract the corresponding attributes. Since we assumed that the attributes are categorical, we calculate the proportion of each attribute class within the extracted attributes and use it as an estimated $p(z|x)$ as shown below:

$$p(z = i|x) = \frac{1}{k} \sum_{(x',z') \in \mathcal{N}_k(x)} 1(z' = i), \tag{4.11}$$

where $\mathcal{N}_k(x)$ is a set of k-nearest neighbor samples from x. If the attribute is continuous, we may use kernel density estimation.

4.3.3 Requirements for the attribute information

The most important assumption in our method is Eq. (4.7). In this subsection, we clarify requirements for this approximation. Since $p(d|x)$ equals $\sum_z p(d|x, z)p(z|x)$, we need the following approximation to have Eq. (4.7):

$$p(d|x, z) \approx p(d|z). \tag{4.12}$$

By multiplying $p(x|z)$ to both sides of Eq. (4.12), we can obtain

$$p(d, x|z) \approx p(d|z)p(x|z). \tag{4.13}$$

Therefore, this approximation assumes that x and d are conditionally independent given z.

We show another aspect of this approximation. By using Bayes' theorem, the left-hand side of Eq. (4.12) can be transformed as follows:

$$\begin{aligned} p(d|x, z) &= \frac{p(x, z|d)p(d)}{p(x, z)} \\ &= \frac{p(x|z, d)p(z|d)p(d)}{p(x|z)p(z)} \\ &= \frac{p(x|z, d)}{p(x|z)} p(d|z). \end{aligned} \tag{4.14}$$

By substituting Eq. (4.14) into Eq. (4.12), we obtain

$$p(x|z, d) \approx p(x|z). \tag{4.15}$$

This equation indicates that, given a certain z, the probability distribution of x is common between domains. Since marginal probability density $p(x|d) = \sum_z p(x|z, d)p(z|d)$ is different between the source and target domains while $p(x|z)$ is common, only the attribute prior given a domain $p(z|d)$ is different between domains. Therefore, the approximation in Eq. (4.7) corresponds to the *latent prior change assumption* that is adopted in some existing works [Storkey and Sugiyama, 2007, Hu et al., 2018b].

Let us explain this assumption by using vehicle recognition from surveillance videos that is the example shown at the end of Section 1. Here, x, d, and z correspond to a cropped video, camera ID, and shooting angle to the target object, respectively. The assumption described in Eq. (4.15) means that the appearance of the target object from a certain shooting angle does not depend on which camera captures the object, which is reasonable if the environment of the captured area is sufficiently similar among different cameras. The discrepancy between the source and target domains stems only from the change of the frequency of the shooting angle.

4.3.4 Characteristics of the proposed method

We clarify some characteristics of our method. First, we take two special cases to explain how our method works, and after that we show how our method is different from the straightforward attribute-based instance weighting.

If the attribute prior is identical between the source and target domains, that means $p(z|d = \mathrm{S}) = p(z|d = \mathrm{T})$, $p(d|z)$ in Eqs. (4.9) and (4.10) are always 0.5 regardless of the value of z. This results in $w(x) = 1$, which indicates that the source data have been already adapted to the target data and we do not need to conduct domain adaptation. This is natural behavior, because we assumed that only the attribute prior changes between domains as noted in the previous subsection.

If $p(z|x)$ is the delta function $\delta(z = z^*)$ where z^* is the attribute value that corresponds to given sample x, $w(x)$ in Eq. (4.8) can be simplified as follows:

$$w(x) = \frac{p(d = \mathrm{T}|z = z^*)}{p(d = \mathrm{S}|z = z^*)} = \frac{p(z = z^*|d = \mathrm{T})}{p(z = z^*|d = \mathrm{S})}. \tag{4.16}$$

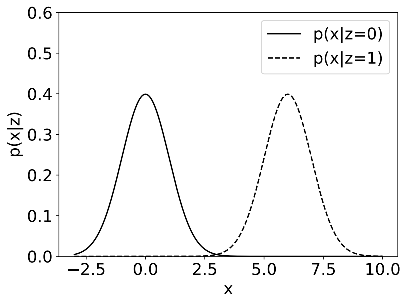
This means that the weight is determined based on only attribute information and not on data. It corresponds to the straightforward approach for attribute-based instance weighting. If we define the weight as

$$w(x, y, z) = \frac{p(x, y, z|d = \mathrm{T})}{p(x, y, z|d = \mathrm{S})}, \tag{4.17}$$

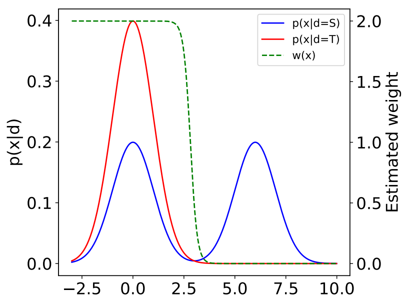
and assume $p(x, y|z, d = \mathrm{S}) = p(x, y|z, d = \mathrm{T})$ that is somewhat a stronger assumption in Eq. (4.15), we can derive the above instance weight as follows:

$$\begin{aligned} w(x, y, z) &= \frac{p(x, y|z, d = \mathrm{T})p(z|d = \mathrm{T})}{p(x, y|z, d = \mathrm{S})p(z|d = \mathrm{S})} \\ &= \frac{p(z|d = \mathrm{T})}{p(z|d = \mathrm{S})}. \end{aligned} \tag{4.18}$$

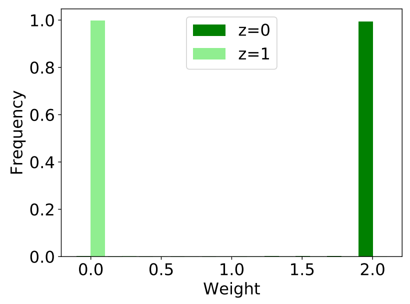
As shown above, our method includes the straightforward attribute-based method as a special case. In other cases, that mean $p(z|x)$ is not a delta function, our method behaves differently compared with the straightforward method.



(a) $p(x|z)$.

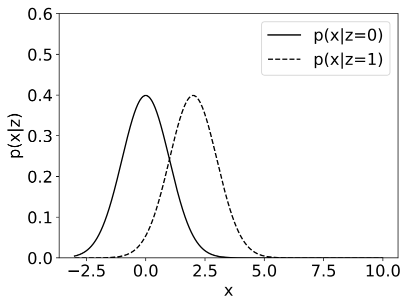


(b) $p(x|d)$ and estimated weight.

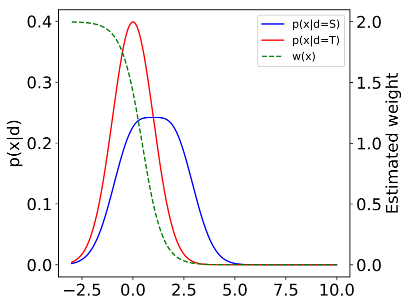


(c) Histogram of the weight for each attribute class

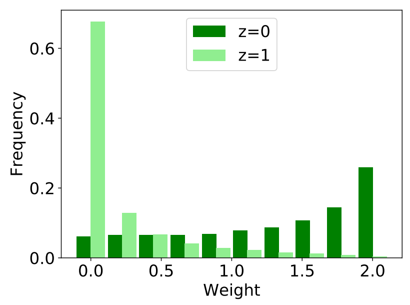
Figure 4.3: One-dimensional example when the overlap between $p(x|z = 0)$ and $p(x|z = 1)$ is small.



(a) $p(x|z)$.



(b) $p(x|d)$ and estimated weight.



(c) Histogram of the weight for each attribute class

Figure 4.4: One-dimensional example when the overlap between $p(x|z = 0)$ and $p(x|z = 1)$ is large.

Let us illustrate the behavior of our method using a simple example. Suppose there are only two attribute classes $z \in \{0, 1\}$ that have one-dimensional Gaussian distributions with different means as shown in Fig. 4.3(a). In the source domain, $[p(z=0|d=\mathrm{S}), p(z=1|d=\mathrm{S})]$ is set to $[0.5, 0.5]$, while it is set to $[1.0, 0.0]$ in the target domain. In this case, the weight estimated in the straightforward method (Eq. (4.18)) leads to a simple delta function, that is $w(x, y, z) = 2 \cdot \delta(z=0)$. In contrast, the weight in our method (Eq. (4.8)) behaves differently according to the amount of overlap between $p(x|z=0)$ and $p(x|z=1)$. Figure 4.3 shows the case in which the overlap is quite small. The weight function $w(x)$ becomes almost the same as a step function over x as shown in Fig. 4.3(b). As a result, the weight over z becomes the delta function that is the same as that in the straightforward method as shown in Fig. 4.3(c). In contrast, when the overlap is large, our method shows somewhat different behavior as presented in Fig. 4.4. In this case, $w(x)$ becomes a smoother function compared with the previous case as shown in Fig. 4.4(b). It leads to non-zero weights for the samples with $z=1$ as shown in Fig. 4.4(c), which means that we can transfer these samples even though the samples with $z=1$ do not appear in the target domain. This characteristic is not available in the straightforward method, because it focuses only on the attribute to estimate the weight. On the other hand, our method utilizes the information of $p(z|x)$, which results in smoother weights that can transfer the source data more efficiently.

4.3.5 Sensitivity to insufficient attribute information

So far, we assume that all attribute information is given. However, this assumption would be unreasonable in real-world situations, and there might be some unknown attributes that affect distributional discrepancy between domains. In this subsection, we analyze how sensitive the proposed method is to such insufficient attribute information.

When given attribute information is not sufficient, the approximation in Eq. (4.12) does not hold, and $p(d|z)$ can be different from $p(d|z, x)$. Here, we assume that given attribute information $p(d|z)$ can be written as a combination of ground-truth information $p(d|z, x)$ and a certain unknown factor, which is formulated as the following equations:

$$p(d=\mathrm{S}|z=i) = (1-\epsilon)p(d=\mathrm{S}|z=i, x) + \epsilon\delta_i(x), \tag{4.19}$$

$$p(d=\mathrm{T}|z=i) = (1-\epsilon)p(d=\mathrm{T}|z=i, x) + \epsilon(1-\delta_i(x)), \tag{4.20}$$

where $\delta_i(x)$ represents the unknown factor that satisfies $0 \le \delta_i(x) \le 1$, and ϵ represents the scale of the unknown factor that satisfies $0 \le \epsilon \le 0.5$. Note that the scale of the the unknown factor is upper-bounded by 0.5, which means that the ground-truth information is more contained in given attribute information than the unknown factor. Substituting Eq. (4.19) and Eq. (4.20) into Eq. (4.8), we can rewrite the instance weight computed in our method as follows:

$$\begin{aligned} w(x) &= \frac{\sum_i \left((1-\epsilon)p(d=\mathrm{T}|z=i,x)+\epsilon(1-\delta_i(x))\right)p(z=i|x)}{\sum_i \left((1-\epsilon)p(d=\mathrm{S}|z=i,x)+\epsilon\delta_i(x)\right)p(z=i|x)} \\ &= \frac{(1-\epsilon)p(d=\mathrm{T}|x)+\epsilon(1-\sum_i \delta_i(x)p(z=i|x))}{(1-\epsilon)p(d=\mathrm{S}|x)+\epsilon\sum_i \delta_i(x)p(z=i|x)} \\ &= \frac{p(d=\mathrm{T}|x)+\epsilon'(1-\alpha(x))}{p(d=\mathrm{S}|x)+\epsilon'\alpha(x)}, \end{aligned} \tag{4.21}$$

where $\epsilon' = \epsilon/(1-\epsilon)$ and $\alpha(x) = \sum_i \delta_i(x)p(z=i|x)$. From the definition of ϵ and $\delta_i(x)$, the range of ϵ' and $\alpha(x)$ can be specified as

$$0 \le \epsilon' \le 1,\ 0 \le \alpha(x) \le 1. \tag{4.22}$$

To evaluate how much the computed weight is biased, we evaluate the ratio between this computed weight $w(x)$ and the ground-truth weight $w^*(x)$. Considering that the ground-truth weight is formulated as shown in Eq. (4.6), we can obtain the ratio $\beta(x)$ as the following equation:

$$\begin{aligned} \beta(x) &= \frac{w(x)}{w^*(x)} \\ &= \frac{p(d=\mathrm{S}|x)\left(p(d=\mathrm{T}|x)+\epsilon'(1-\alpha(x))\right)}{p(d=\mathrm{T}|x)\left(p(d=\mathrm{S}|x)+\epsilon'\alpha(x)\right)}. \end{aligned} \tag{4.23}$$

If $\epsilon = 0$, ϵ' equals zero, and, consequently, $\beta(x)$ is equal to one for any x, which means that the instance weight computed in the proposed method matches the ground-truth weight if all attribute information is given. Otherwise, $\beta(x)$ can be smaller or larger than one, and the instance weight is over- or under-estimated due to insufficient attribute information. Here, we focus on the worst case, namely, the maximum and minimum of $\beta(x)$ over all possible variations of the unknown factor $\delta_i(x)$. Since ϵ' and $\alpha(x)$ are constrained to be in the range shown in Eq. (4.22), $\beta(x)$ takes its maximal value $\beta_{\max}(x)$ when $\alpha(x) = 0$, while it takes its minimal value $\beta_{\min}(x)$ when $\alpha(x) = 1$. Specifically, $\beta_{\max}(x)$ and $\beta_{\min}(x)$ are obtained as follows:

$$\beta_{\max}(x) = \frac{p(d=\mathrm{T}|x)+\epsilon'}{p(d=\mathrm{T}|x)}, \tag{4.24}$$

$$\beta_{\min}(x) = \frac{p(d=\mathrm{S}|x)}{p(d=\mathrm{S}|x)+\epsilon'}. \tag{4.25}$$

Figure 4.5 shows how $\beta_{\max}(x)$ and $\beta_{\min}(x)$ change according to the ground-truth weight and ϵ. When the ground-truth weight is small, $\beta_{\max}(x)$ can be quite large depending on ϵ, but $\beta_{\min}(x)$ is relatively close to one. On the other hand, when the ground-truth weight is large, $\beta_{\min}(x)$ can be large, but $\beta_{\max}(x)$ is relatively close to one. Such behavior indicates that the estimated weight in the proposed method tends to avoid an extremely large or small value, which should result in stable performance even when insufficient attribute information is given.

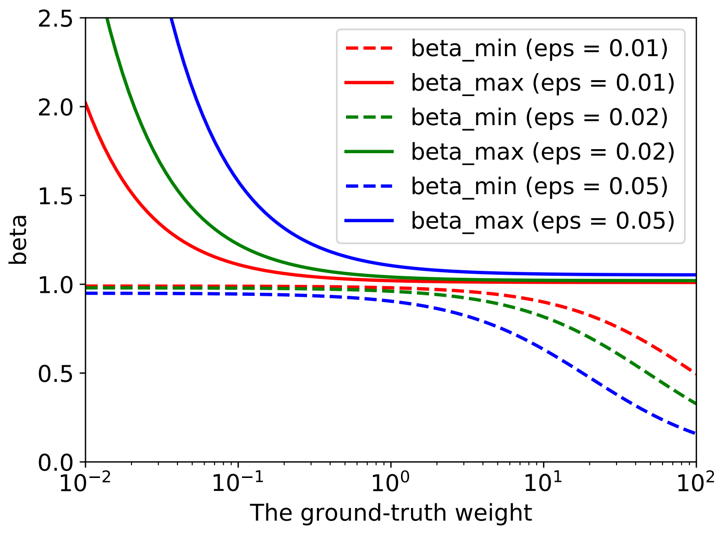


Figure 4.5: The upper-bound and lower-bound of $\beta(x)$.

Table 4.1: The mixing ratios of GMM for toy datasets.

Dataset		Centroid				
		-0.75π	-0.5π	0.0	0.5π	0.75π
A	$d = \mathrm{S}$	0.1	0.1	0.2	0.4	0.2
	$d = \mathrm{T}$	0.2	0.4	0.2	0.1	0.1
B	$d = \mathrm{S}$	0.05	0.05	0.1	0.5	0.3
	$d = \mathrm{T}$	0.3	0.5	0.1	0.05	0.05
C	$d = \mathrm{S}$	0.05	0.05	0.1	0.1	0.7
	$d = \mathrm{T}$	0.7	0.1	0.1	0.05	0.05

4.4 Experiments

In this section, we show the experimental results with both toy datasets and benchmark datasets.

4.4.1 Toy datasets

We conducted experiments with a 2-dimensional toy dataset for binary classification, In this dataset, the first feature x_0 stemmed from a Gaussian mixture model (GMM) that has five centroids (-0.75π, -0.5π, 0.0, 0.5π, 0.75π) with common standard deviation $\sigma = 0.2\pi$, and the second feature x_1 stemmed from the uniform distribution from -2.0 to 2.0. For each sample, the index of the corresponding centroid was treated as attribute $z \in \{0, 1, 2, 3, 4\}$. The mixing ratio of GMM was set differently for the source and target domains as shown in Table 4.1. Note that Eq. (4.15) exactly holds in this dataset, since the Gaussian distribution for each centroid is common among domains. To change the difficulty of domain adaptation, we constructed

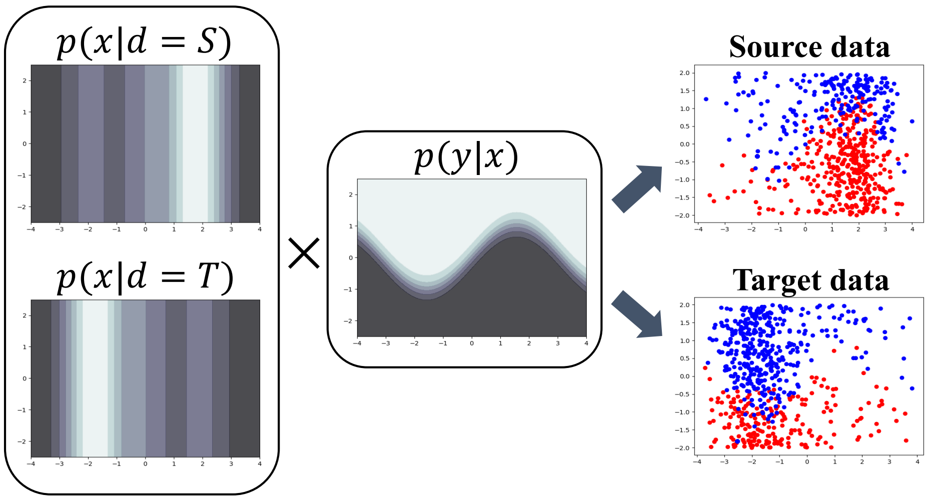


Figure 4.6: Generation of toy datasets.

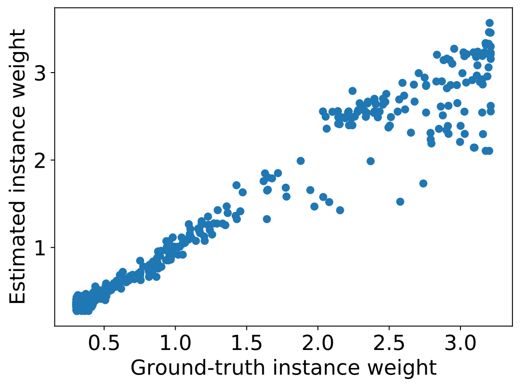
three datasets (Datasets A–C) by changing the discrepancy of the ratios between the domains. The posterior $p(y|x)$ is determined by

$$p(y|x) = \frac{1}{1 + \exp\left(-5.0(x_1 - \sin x_0)\right)}. \tag{4.26}$$

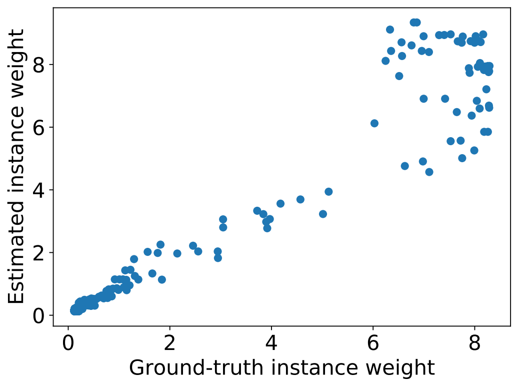
To make the dataset, first, we generated the sample (x, z) according to the data distribution that is previously described, then, we determined its label by randomly sampling according to the above posterior. Figure 4.6 shows a brief flow of how to generate the toy datasets. We generated 600 samples as source and target data, respectively. Note that we can obtain ground-truth $w(x)$ by calculating Eq. (4.4) with true probability density functions $p(x|d)$.

First, we evaluated the accuracy of the weights estimated by our method by comparing them with the ground-truth weights. To quantitatively evaluate the accuracy, we compared our method with unconstrained Least-Squares Importance Fitting (uLSIF) [Kanamori et al., 2009] that is one of the representative methods to estimate a probability density ratio. Using the target data, we estimated the weight by uLSIF, and compared its estimation error with that of our method. We measured the error by the root mean squared error. The results are shown in Table 4.2. Although our method does not use any target data, it shows better performance than uLSIF. This indicates that attribute information can be more useful to estimate the probability density ratio. Figure 4.7 shows the results for each dataset, in which the horizontal and vertical axises represent the ground-truth weight and the estimated weight, respectively. We can see that many samples are close to the diagonal line, which means that our method successfully estimates the weights accurately.

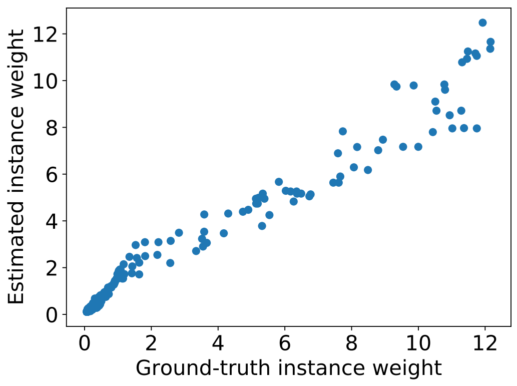
We also evaluate the performance of our method as domain adaptation. We trained a classifier with weighted source data and tested it with the target data. To train a classifier, we used C-support vector machine (C-SVM) with the Gaussian kernel. To



(a) Dataset A



(b) Dataset B



(c) Dataset C

Figure 4.7: Instance weights estimated by our method.

Table 4.2: The estimation error of weights.

	Dataset		
	A	B	C
The proposed method	0.179	0.573	0.679
uLSIF	0.291	0.664	0.743

tune its hyper-parameters that are regularization coefficient C and kernel width σ, we conducted importance-weighted cross validation, which requires only source data for model selection. First, we split the source data into the training and validation datasets. We trained the instance weight estimator and the classifier with the training dataset, and the classifier is tested with the validation dataset that is weighted by the weight estimator. We compared three methods: training without weights, training with estimated weights, and training with the ground-truth weights. Table 4.3 shows the accuracy of the SVM trained by each method. Our method achieved higher accuracy than that without importance weights and almost reached the same performance

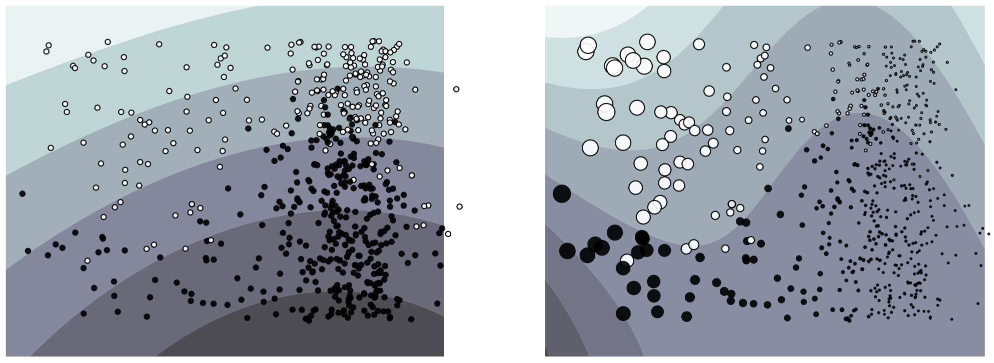

(a) Without instance weights (b) With estimated weights

Figure 4.8: Visualization of instance weights and the trained classifier (○: positive-class instances, ●: negative-class instances).

Table 4.3: The accuracy of the trained SVM.

	Dataset		
	A	B	C
w/o weights	$91.3 \pm 1.1\%$	90.4 ± 1.0 %	88.1 ± 1.3 %
w/ estimated weights	$92.4 \pm 0.4\%$	91.0 ± 0.5 %	90.2 ± 0.8 %
w/ ground-truth weights	$92.4 \pm 0.4\%$	90.9 ± 0.6 %	90.4 ± 0.7 %

as that with ground-truth weights, though our method does not utilize ground-truth weights or any target data. Figure 4.8 visualizes the instance weights and the trained classifier. The size of circles corresponds to the value of the instance weight, and contour lines represent the output of the decision function of SVM. Note that the true decision boundary is a sinusoidal curve as shown in Fig. 4.6. Since only few source data are distributed at the left-hand side while many target data are at that side, large weights are assigned to those source data in our method, which results in a more accurate classifier especially at the left-hand side.

4.4.2 Digit image classification

For the first experiment, we used the MNIST dataset [LeCun et al., 1998] that contains handwritten digit images. The task is to classify these images into ten classes that correspond to digit numbers. We randomly chose 10,000 samples from the training data, and used them as source data, while the test data that includes 10,000 samples were used as target data. To make the source and target data have different data distributions, we clockwisely rotated each image with a randomly determined angle, where we set different probability distributions of the rotation angle for source and target data as shown in Table 4.4. We measured the performance of our method by the ac-

Table 4.4: The probability distributions of the rotation angle used in the experiment with the MNIST dataset.

	Rotation angle				
	$-\frac{1}{3}\pi$	$-\frac{1}{6}\pi$	0	$+\frac{1}{6}\pi$	$+\frac{1}{3}\pi$
Source	0.05	0.05	0.1	0.5	0.3
Target	0.3	0.5	0.1	0.05	0.05

Table 4.5: The network architectures used in the experiments with the MNIST dataset. MP2, BN, and FC denote 2×2 max-pooling, batch normalization, and a fully-connected layer, respectively.

Layer type	Size / number of filters
convolution + ReLU	5×5 / 20
MP2 + BN	2×2 / 20
convolution + ReLU	5×5 / 50
MP2 + BN	2×2 / 50
FC + ReLU	1 / 200
FC + softmax	1 / 10

curacy of the classifier trained with weighted source data similarly to the previous experiments. Instead of SVM, we used a deep neural network in this experiment. Table 4.5 shows its network architecture that is loosely based on *LeNet* [LeCun et al., 1998] but is modified by adding batch normalization layers. We trained the network by stochastic gradient descent with momentum, and the number of total update iterations was 10,000. To calculate the weight in our method, we estimated $p(z|x)$ by the k-nearest neighbor method with the features at the last hidden layer of the network. Since the calculation cost of the weight estimation is not small compared with that of the training network, we calculated the weights after each 100 iterations, and fixed them for the next 100 iterations. We used the weights to calculate the sampling probability of each sample when making a mini-batch.

Table 4.6 shows the accuracy of the trained classifier on the MNIST dataset. Without instance weights, the accuracy decreased from 97.1% to 93.8% when shifting from the source to target domains. On the other hand, our method suppressed this degradation of the classification performance, and achieved 94.9% in the target domain. Interestingly, the accuracy in the source domain remains almost unchanged while adopting the instance weights.

4.4.3 Age estimation from facial image

For the second experiment, we used the Adience dataset [Eidinger et al., 2014] that contains facial images with age and gender annotations. In this experiment, we con-

Table 4.6: Accuracy of the trained DNN on the MNIST dataset.

	Target data	Source data
w/o weights	93.8%	97.1%
Our method	94.9%	97.0%

Figure 4.9: Adience dataset [Eidinger et al., 2014]. The images are from authors' website.

ducted age estimation while considering gender as an attribute. Since eight age groups are defined in this dataset, age estimation can be formulated as an eight-class classification problem. There are five sub-datasets in this dataset, and we used the fifth sub-dataset as target data and the other sub-datasets as source data. While gender in this dataset is almost balanced, we artificially made it imbalanced in the target data to change the data distribution. We varied this imbalance, and evaluated our method for each setting. The network architecture for this experiment is shown in Table 4.7. The number of total update iterations was 5,000. The other setting is the same as that in the previous experiment.

Table 4.8 shows the accuracy of the trained classifier on the Adience dataset. When the ratio between male and female samples in the target data is set to $[0.5, 0.5]$, the accuracy of our method is almost the same as that of the other methods. This is because the ratio in the source data is also balanced and the data distribution is almost the same between the source and target data. In contrast, when the ratio became imbalanced, our method achieved better performance. It indicates that the effectiveness of our method gets more significant as the discrepancy between the source and target data distributions becomes larger. The straightforward attribute-based weight did not lead to better performance, because it could not effectively utilize female samples in heavily imbalanced case. For example, when the ratio was set to $[0.9, 0.1]$, the average weight of female examples was 9 times smaller than that of male examples in the

Table 4.7: The network architectures used in the experiments with Adience and VisDA2017 datasets. MP2, BN, and FC denote 2×2 max-pooling, batch normalization, and a fully-connected layer, respectively.

Layer type	Size / number of filters
convolution + ReLU	3×3 / 16
MP2 + BN	2×2 / 16
convolution + ReLU	3×3 / 24
MP2 + BN	2×2 / 24
convolution + ReLU	3×3 / 32
MP2 + BN	2×2 / 32
FC + ReLU	1 / 500
FC + softmax	1 / 2 or 12

Table 4.8: Accuracy of the trained DNN on Adience dataset.

	[male, female] at target data		
	$[0.5, 0.5]$	$[0.7, 0.3]$	$[0.9, 0.1]$
w/o weights	$39.8 \pm 0.5\%$	$40.0 \pm 0.9\%$	$39.7 \pm 0.5\%$
The straightforward attribute-based weight	$39.3 \pm 0.3\%$	$39.7 \pm 0.5\%$	$39.9 \pm 0.3\%$
Our method	$39.9 \pm 0.4\%$	$40.8 \pm 0.7\%$	$41.4 \pm 0.3\%$

straightforward method, while, in the proposed method, it became 2.2 times smaller, which is substantially more smooth weight than the straightforward method.

4.4.4 Object recognition

For more large-scale experiment, we used the VisDA2017 classification dataset [Peng et al., 2017]. This dataset contains object images with twelve categories, and the task is to discriminate the object category from the given image. Since the azimuth of the captured object is also provided in this dataset, we discretized the azimuth into five classes and used it as an attribute. We constructed the source and target data as shown in Table 4.9. Intuitively, the source domain is biased to “front-view” images, while the target domain is biased to “rear-view” images. We varied these bias by changing r in Table 4.9. The network architecture and the setting for training the network are same as in the previous experiment.

Table 4.10 shows the experimental result with VisDA2017 dataset. When r is small, the discrepancy between the source and target domain is not large, which results in almost the same accuracy of all methods. As r increases, the advantage of our method becomes large as same with the result of the previous experiment.

Table 4.9: The number of data used in the experiment with the VisDA2017 dataset. M was set to 24,000, and r was varied in the experiment to control the discrepancy between domains.

	Azimuth of the captured objects				
	10-61	78-129	146-197	214-265	282-333
Source	M	M/r	M/r	M/r^2	M/r
Target	M/r^2	M/r	M/r	M	M/r

Table 4.10: Accuracy of the trained DNN on VisDA2017 dataset. Standard errors are omitted, because they are very small (≤ 0.1) in this experiment.

	Dataset		
	$r = 2$	$r = 3$	$r = 4$
w/o weights	95.6%	93.7%	91.5%
The straightforward attribute-based weight	95.6%	93.7%	92.1%
Our method	95.6%	94.0%	92.5%

4.5 Conclusion

In this chapter, we proposed a zero-shot domain adaptation method based on attribute information. We showed how to estimate instance weights for source data by using the attribute information, and also clarified requirements for the attribute information to be useful, which is actually the same assumption adopted in some existing works. In addition, we revealed that our method can provide more precise estimation of sample-wise transferability than a straightforward attribute-based reweighting approach. Experimental results with both toy datasets and benchmark datasets showed that our method can accurately estimate the instance weights and performed well as domain adaptation. Future works include integration of our method with other recent domain adaptation methods and extension to the case in which the attribute information is partially available.

Chapter 5

Source-free Domain Adaptation via Distributional Alignment by Matching Batch Normalization Statistics

In this chapter, we describe our third contribution to domain adaptation with scarce data. Specifically, we propose a novel method for source-free domain adaptation. In the source-free setting, we cannot access source data during adaptation, while unlabeled target data and a model pretrained with source data are given. Due to lack of source data, we cannot directly match the data distributions between domains unlike typical domain adaptation algorithms. To cope with this problem, we propose utilizing batch normalization statistics stored in the pretrained model to approximate the distribution of unobserved source data. Specifically, we fix the classifier part of the model during adaptation and only fine-tune the remaining feature encoder part so that batch normalization statistics of the features extracted by the encoder match those stored in the fixed classifier. Additionally, we also maximize the mutual information between the features and the classifier's outputs to further boost the classification performance. Experimental results with several benchmark datasets show that our method achieves competitive performance with state-of-the-art domain adaptation methods even though it does not require access to source data.

5.1 Introduction

In typical statistical machine learning algorithms, test data are assumed to stem from the same distribution as training data [Hastie et al., 2009]. However, this assumption is often violated in practical situations, and the trained model results in unexpectedly poor performance [Quionero-Candela et al., 2009]. This situation is called domain shift, and many researchers have intensely worked on domain adaptation [Csurka, 2017, Wilson and Cook, 2020] to overcome it. A common approach for domain adaptation is to jointly minimize a distributional discrepancy between domains in a feature space as well as the prediction error of the model [Wilson and Cook, 2020], as shown in Fig. 5.1(a). Deep neural networks (DNNs) are particularly popular for this joint training, and recent methods using DNNs have demonstrated excellent performance under domain shift [Wilson and Cook, 2020].

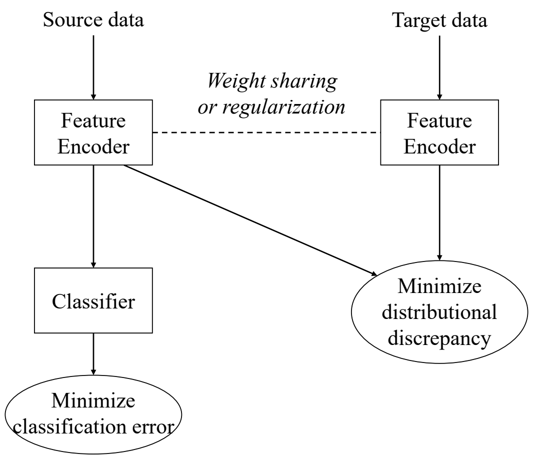
Many domain adaptation algorithms assume that they can access labeled source data as well as target data during adaptation. This assumption is essentially required to evaluate the distributional discrepancy between domains as well as the accuracy of the model's prediction. However, it can be unreasonable in some cases, for example, due to data privacy issues or too large-scale source datasets to be handled at the environment where the adaptation is conducted. To tackle this problem, a few recent studies [Kundu et al., 2020, Li et al., 2020, Liang et al., 2020a] have proposed source-free domain adaptation methods in which they do not need to access the source data.

In source-free domain adaptation, the model trained with source data is given instead of source data themselves, and it is fine-tuned through adaptation with unlabeled target data so that the fine-tuned model works well in the target domain. Since it seems quite hard to evaluate the distributional discrepancy between unobservable source data and given target data, previous studies mainly focused on how to minimize the prediction error of the model with unlabeled target data, for example, by using pseudo-labeling [Liang et al., 2020a] or a conditional generative model [Li et al., 2020]. However, due to lack of the distributional alignment, the performance of those methods is not guaranteed by theories exploited in typical domain adaptation studies [Ben-David et al., 2010].

In this chapter, we propose a novel method for source-free domain adaptation. Figure 5.1(b) shows our setup in comparison with that of typical domain adaptation methods shown in Fig. 5.1(a). In our method, we explicitly minimize the distributional discrepancy between domains by utilizing batch normalization (BN) statistics stored in the pretrained model. Since we fix the pretrained classifier during adaptation, the BN statistics stored in the classifier can be regarded as representing the distribution of source features extracted by the pretrained encoder. Based on this idea, to minimize the discrepancy, we train the target-specific encoder so that the BN statistics of the target features extracted by the encoder match with those stored in the classifier. We also adopt information maximization as in Liang et al. [2020a] to further boost the classification performance of the classifier in the target domain. Our method is apparently simple but effective; indeed, we will validate its advantage through extensive experiments on several benchmark datasets.

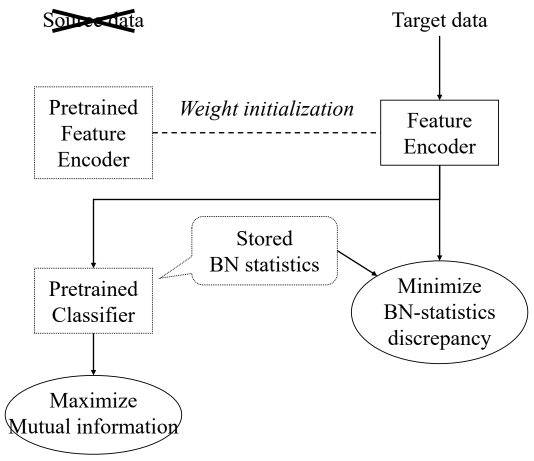
5.2 Related work

In this section, we introduce existing works on domain adaptation that are related to ours and also present a formulation of batch normalization.



Source data
Target data
Weight sharing or regularization
Feature Encoder
Feature Encoder
Classifier
Minimize distributional discrepancy
Minimize classification error

(a) General setup commonly adopted in recent typical domain adaptation methods. This visualization is inspired by [Wilson and Cook, 2020].



(b) Our setup for source-free domain adaptation.

Figure 5.1: Comparison between typical domain adaptation methods and our method. A rectangle with solid lines represents a trainable component, while that with dotted lines represent a fixed component during adaptation.

5.2.1 Domain adaptation

Given source and target data, the goal of domain adaptation is to obtain a good prediction model that performs well in the target domain [Csurka, 2017, Wilson and Cook, 2020]. Importantly, the data distributions are significantly different between the domains, which means that we cannot simply train the model with source data to maximize the performance of the model for target data. Therefore, in addition to minimizing the prediction error using labeled source data, many domain adaptation algorithms try to align the data distributions between domains by adversarial training [Ganin et al., 2016, Tzeng et al., 2017, Deng et al., 2019, Xu et al., 2019] or explicitly minimizing a distributional-discrepancy measure [Long et al., 2015, Bousmalis et al., 2016, Long et al., 2017]. This approach has empirically shown excellent performance and is also validated in theory [Ben-David et al., 2010]. However, since this distribution alignment requires access to source data, these methods cannot be directly applied to the source-free domain adaptation setting.

In source-free domain adaptation, we can only access target data but not source data, and the model pretrained with the source data is given instead of the source data. This challenging problem has been tackled in recent studies. Li et al. [2020] proposed joint training of the target model and the conditional GAN (Generative Adversarial Network) [Mirza and Osindero, 2014] that is to generate annotated target data. Liang et al. [2020a] explicitly divided the pretrained model into two modules, called a feature encoder and a classifier, and trained the target-specific feature encoder while fixing the classifier. To make the classifier work well with the target features, this training jointly conducts both information maximization and self-supervised pseudo-labeling with the fixed classifier. Kundu et al. [2020] adopted a similar architecture but it has three modules: a backbone model, a feature extractor, and a classifier. In the adaptation phase, only the feature extractor is tuned for the target domain by minimizing the entropy of the classifier's output. Since the methods shown above do not try to align data distributions between domains, they cannot essentially avoid confirmation bias of the model and also cannot benefit from well-exploited theories in the studies on typical domain adaptation problems [Ben-David et al., 2010].

5.2.2 Batch normalization

Batch normalization (BN) [Ioffe and Szegedy, 2015] has been widely used in modern architectures of deep neural networks to make their training faster as well as being stable. It normalizes each input feature within a mini-batch in a channel-wise manner so that the output has zero-mean and unit-variance. Let B and $\{z_i\}_{i=1}^{B}$ denote the mini-batch size and the input features to the batch normalization, respectively. Here, we assume that the input features consist of C channels as $z_i = [z_i^{(1)}, ..., z_i^{(C)}]$ and each channel contains n_c features. BN first computes the means $\{\mu_c\}_{c=1}^{C}$ and variances $\{\sigma_c^2\}_{c=1}^{C}$ of the features for each channel within the mini-batch:

$$\mu_c = \frac{1}{n_c B} \sum_{i}^{B} \sum_{j}^{n_c} z_i^{(c)}[j] \tag{5.1}$$

$$\sigma_c^2 = \frac{1}{n_c B} \sum_{i}^{B} \sum_{j}^{n_c} (z_i^{(c)}[j] - \mu_c)^2, \tag{5.2}$$

where $z_i^{(c)}[j]$ is the j-th feature in $z_i^{(c)}$. Then, it normalizes the input features by using the computed BN statistics:

$$\tilde{z}_i^{(c)} = \frac{z_i^{(c)} - \mu_c}{\sqrt{\sigma_c^2 + \epsilon}}, \tag{5.3}$$

where ϵ is a small positive constant for numerical stability. In the inference phase, BN cannot always compute those statistics, because the input data do not necessarily compose a mini-batch. Instead, BN stores the exponentially weighted averages of the BN statistics in the training phase and uses them in the inference phase to compute $\tilde{z}$ in Eq. (5.3). To this end, at each iteration during training, the stored BN statistics are updated with the BN statistics of the current mini-batch as follows:

$$\hat{\mu}_c^{(\tau+1)} = \gamma\hat{\mu}_c^{(\tau)} + (1-\gamma)\mu_c, \tag{5.4}$$

$$\hat{\sigma}_c^{(\tau+1)} = \gamma\hat{\sigma}_c^{(\tau)} + (1-\gamma)\sigma_c, \tag{5.5}$$

where $\hat{\mu}_c^{(\tau)}$ and $\hat{\sigma}_c^{(\tau)}$ are the stored BN statistics at the τ-th iteration, and γ is a hyper-parameter that satisfies $0 < \gamma < 1$.

Since BN renormalizes features to have zero-mean and unit-variance, several methods [Li et al., 2018b, Chang et al., 2019, Wang et al., 2019] adopted domain-specific BN to explicitly align both the distribution of source features and that of target features into a common distribution. Since the domain-specific BN methods are jointly trained during adaptation, we cannot use these methods in the source-free setting.

5.3 Proposed method

In this section, we propose a novel method for source-free domain adaptation.

5.3.1 Overview

Figure 5.2 shows an overview of our method. We assume that the model pretrained with source data is given, and it conducts BN at least once somewhere inside the model. Before conducting domain adaptation, we divide the model in two sub-models: a feature encoder and a classifier, so that BN comes at the very beginning of the classifier. Then, for domain adaptation, we fine-tune the encoder with unlabeled target data with the classifier fixed. After adaptation, we use the fine-tuned encoder and the fixed classifier to predict the class of test data in the target domain.

To make the fixed classifier work well in the target domain after domain adaptation, we aim to obtain a fine-tuned encoder that satisfies the following two properties:

- The distribution of target features extracted by the fine-tuned encoder is well aligned to that of source features extracted by the pretrained encoder.
- The features extracted by the fine-tuned encoder are sufficiently discriminative for the fixed classifier to accurately predict the class of input target data.

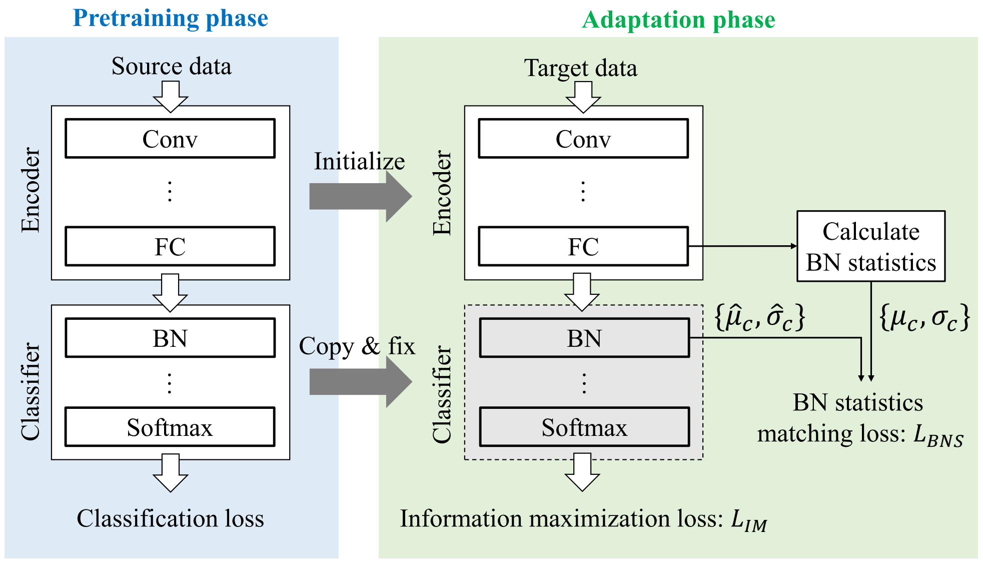


Figure 5.2: An overview of the proposed method.

To this end, we jointly minimize both the BN-statistics matching loss and information maximization loss to fine-tune the encoder. In the former loss, we approximate the distribution of unobservable source features by using the BN statistics stored in the first BN layer of the classifier, and the loss explicitly evaluates the discrepancy between source and target feature distributions based on those statistics. Therefore, minimizing this loss leads to satisfying the first property shown above. On the other hand, the latter loss is to make the predictions by the fixed classifier certain for every target sample as well as diverse within all target data, and minimizing this loss leads to fulfilling the second property. Below, we describe the details of these losses.

5.3.2 Distribution alignment by matching batch normalization statistics

Since the whole model is pretrained with source data and we fix the classifier while finetuning the encoder, the BN statistics stored in the first BN in the classifier can be seen as the statistics of the source features extracted by the pretrained encoder. We approximate the source-feature distribution by using these statistics. Specifically, we simply use a Gaussian distribution for each channel denoted by $\mathcal{N}(\hat{\mu}_c, \hat{\sigma}_c^2)$ where $\hat{\mu}_c$ and $\hat{\sigma}_c^2$ are the mean and variance of the Gaussian distribution which are the stored BN statistics corresponding to the c-th channel.

To match the feature distributions between domains, we define the BN-statistics matching loss, which evaluates the averaged Kullback-Leibler (KL) divergence from the target-feature distribution to the approximated source-feature distribution:

$$\begin{aligned} L_{\mathrm{BNM}}(\{x_i\}_{i=1}^{B}, \theta) &= \frac{1}{C}\sum_{c=1}^{C} \mathrm{KL}\left(\mathcal{N}(\hat{\mu}_c, \hat{\sigma}_c^2) || \mathcal{N}(\mu_c, \sigma_c^2)\right) \\ &= \frac{1}{2C}\sum_{c=1}^{C}\left(\log\frac{\sigma_c^2}{\hat{\sigma}_c^2} + \frac{\hat{\sigma}_c^2 + (\hat{\mu}_c - \mu_c)^2}{\sigma_c^2} - 1\right), \end{aligned} \tag{5.6}$$

where $\{x_i\}_{i=1}^{B}$ is a mini-batch from the target data, θ is a set of trainable parameters of the encoder, and μ_c and σ_c are the BN statistics of the c-th channel computed from the target mini-batch. Note that, since μ_c and σ_c are calculated from the features extracted by the encoder, they depend on θ. Here, we also approximate the target-feature distribution with another Gaussian distribution so that the KL divergence can be efficiently computed in a parametric manner. By minimizing this loss, we can explicitly reduce the discrepancy between the distribution of unobservable source features and that of target features.

In Eq. (5.6), we chose the KL divergence to measure the distributional discrepancy between domains. There are two reasons for this choice. First, the KL divergence between two Gaussian distributions is easy to compute with the BN statistics as shown in Eq. (5.6). Moreover, since these statistics are naturally computed in the BN layer, calculating this divergence only requires tiny calculation costs. Secondly, it can be theoretically justified from the perspective of risk minimization in the target domain. When we consider a binary classification task, the expected risk of any hypothesis h in the target domain can be upper-bounded under some mild assumptions as the following inequality [Ben-David et al., 2010]:

$$R_{\mathrm{T}}(h) \leq R_{\mathrm{S}}(h) + d_1(p_{\mathrm{S}}, p_{\mathrm{T}}) + \beta, \tag{5.7}$$

where $R_{\mathrm{S}}(h)$ and $R_{\mathrm{T}}(h)$ denote the expected risk of h under the source-data distribution p_{S} and target-data distribution p_{T}, respectively, $d_1(p, q)$ represents the total variation distance between p and q, and β is a constant value that is expected to be sufficiently small. This inequality roughly gives a theoretical justification to recent domain adaptation algorithms, that is, joint minimization of both the distributional discrepancy between domains (corresponding to the second term of the bound in Eq. (5.7)) and the prediction error of the model (corresponding to the first term of the bound in Eq. (5.7)). Here, the total variation distance can be related to the KL divergence by Pinsker's inequality [Csiszar and Körner, 2011]:

$$d_1(p, q) \leq \sqrt{\frac{1}{2}\mathrm{KL}(p||q)}. \tag{5.8}$$

Consequently, we can guarantee that minimizing the KL divergence between domains minimizes the bound of the target risk.

5.3.3 Mutual information maximization

Only aligning the marginal feature-distributions between domains does not guarantee that the fixed classifier works well in the target domain, because the features extracted by the encoder are not necessarily discriminative. If the features are sufficiently discriminative for the classifier, we can expect that the output of the classifier is almost always a one-hot vector but is diverse within the target data. Therefore, following the approach presented in Liang et al. [2020a], we also adopt the information maximization loss to make the classifier work accurately.

$$L_{\mathrm{IM}}(\{x_i\}_{i=1}^B, \theta) = -H\left(\frac{1}{B}\sum_i^B f_\theta(x_i)\right) + \frac{1}{B}\sum_i^B H\left(f_\theta(x_i)\right), \tag{5.9}$$

where $H(p(y)) = -\sum_{y'} p(y') \log p(y')$ is the entropy function, and $f_\theta(x)$ denotes the output of the classifier. The first term in the right-hand side of Eq. (5.9) is the negative entropy of the averaged output of the classifier, and minimizing it leads to large diversity of the output within the mini-batch. The second term is the averaged entropy of the classifier's output, and minimizing it makes the outputs close to one-hot vectors. Therefore, the features extracted by the target encoder are induced to be discriminative by minimizing the information maximization loss.

5.3.4 Optimization

Finally, our source-free domain adaptation method is formulated as joint minimization of both the BN-statistics matching loss in Eq. (5.6) and the information maximization loss in Eq. (5.9):

$$\min_\theta \mathbb{E}_{\{x_i\}_{i=1}^B \sim \mathcal{D}_t}\left[L_{\mathrm{IM}}(\{x_i\}_{i=1}^B, \theta) + \lambda L_{\mathrm{BNM}}(\{x_i\}_{i=1}^B, \theta)\right], \tag{5.10}$$

where $\mathcal{D}_t$ is the target dataset from which the mini-batch is sampled, and a hyper-parameter λ controls the balance between the two terms. Note that this optimization can be conducted without the source data, which means that we do not need to access to the source data during adaptation.

5.4 Experiments

We conducted experiments with several datasets that are commonly used in existing works on domain adaptation. Specifically, we used digit recognition datasets (MNIST [LeCun et al., 1998], USPS [LeCun et al., 1990], and SVHN [Netzer et al., 2011]) and an object recognition dataset (Office-31 dataset [Saenko et al., 2010]). In the experiment, we first pretrained the model with the source training data. Following the setup in Liang et al. [2020a], we used standard cross-entropy loss with label smoothing for this pretraining. Then, we apply our source-free domain adaptation method to fine-tune the pretrained model with the target training data. We used Adam optimizer for both pretraining and adaptation. The number of iterations in the optimization was set to 30,000, and the batch size was set to 64. The hyper-parameter λ in Eq. (5.10) is set to 10 in all experiments except for those in section 5.4.3. The performance of the domain adaptation is evaluated by test accuracy of the fine-tuned model on the target test data. We report the averaged accuracy as well as the standard deviation over five runs with random initialization of the model at the pretraining phase.

We compared the performance of our method with those of the state-of-the-art methods for source-free domain adaptation [Li et al., 2020, Liang et al., 2020a], which are most related to our work. We did not include the work by Kundu et al. [2020] in this comparison, because it is designed for more difficult setting, called universal domain adaptation. For reference, we also show the performance of the recent methods for typical domain adaptation [Tzeng et al., 2017, Deng et al., 2019, Xu et al., 2019], though they require access to the source data during adaptation.

5.4.1 Object recognition

The Office-31 dataset comprises three domains: Amazon (A), DSLR (D), and Webcam (W). We examined all possible combinations for the adaptation, which results in six scenarios. Following the setup of Ganin et al. [2016], Liang et al. [2020a], we used ResNet-50 pretrained with the ImageNet classification dataset as a backbone model. We removed the original FC layer from the pretrained ResNet-50 and added a bottleneck FC layer (256 units) and a classification FC layer (31 units). A BN layer is put before and after the bottleneck layer, and we used the last one to calculate our BN-statistics matching loss. Note that the backbone part is fixed in our experiments.

Table 5.1 shows the test accuracy of the adapted models at the target data. The results shown above the double line are those of the source-free domain adaptation methods, while the remaining ones are those of the other typical domain adaptation methods. Our method achieved the best accuracy at three out of six scenarios, and, surprisingly, its performance reached or exceeded the performance of the state-of-the-art typical domain adaptation methods in those cases. Moreover, our method also shows competitive performance in the other scenarios except for A $\rightarrow$ D. Since SHOT [Liang et al., 2020a] also adopts the information maximization loss, these results indicate that our BN-statistics matching loss substantially improves the performance of the adaptation by successfully reducing the distributional discrepancy between domains. The model adaptation [Li et al., 2020] also works well through the all scenarios. However, considering that it requires training of a conditional GAN while adaptation, our method is quite appealing due to simplicity of its training procedure as well as its high performance.

Table 5.1: Experimental results with Office-31 dataset. The bold number represents the highest test accuracy among the source-free domain adaptation methods, and the underline represents the second highest one.

Method	A→D	A→W	D→A	D→W	W→A	W→D
SHOT [Liang et al., 2020a]	**94.0**	90.1	74.7	98.4	74.3	$\underline{99.9}$
Model adaptation [Li et al., 2020]	$\underline{92.7}$±0.4	**93.7**±0.2	$\underline{75.3}$±0.5	$\underline{98.5}$±0.1	**77.8**±0.1	99.8±0.2
Our method	89.0±0.2	$\underline{91.7}$±1.0	**78.5**±0.2	**98.9**±0.1	$\underline{76.6}$±0.7	**100.0**±0.0
ADDA [Tzeng et al., 2017]	77.8±0.3	86.2±0.5	69.5±0.4	96.2±0.3	68.9±0.5	98.4±0.3
rRevGrad+CAT [Deng et al., 2019]	90.8±1.8	94.4±0.1	72.2±0.6	98.0±0.1	70.2±0.1	100.0±0.0
d-SNE [Xu et al., 2019]	94.7±0.4	96.6±0.1	75.5±0.4	99.1±0.2	74.2±0.2	100.0±0.0

Table 5.2: Experimental results with digit recognition datasets. The bold number represents the highest test accuracy among the source-free domain adaptation methods, and the underline represents the second highest one.

Method	USPS→MNIST	MNIST→USPS	SVHN→MNIST
SHOT [Liang et al., 2020a]	98.4±0.6	**98.0**±0.2	98.9±0.0
Model adaptation [Li et al., 2020]	**99.3**±0.1	97.3±0.2	**99.4**±0.1
Our method	$\underline{99.1}$±0.0	$\underline{97.7}$±0.2	$\underline{99.1}$±0.0
ADDA [Tzeng et al., 2017]	90.1±0.8	89.4±0.2	76.0±1.8
rRevGrad+CAT [Deng et al., 2019]	96.0±0.9	94.0±0.7	98.8±0.0
d-SNE [Xu et al., 2019]	98.5±0.4	99.0±0.1	96.5±0.2

Table 5.3: The network architectures used in the experiments with USPS $\leftrightarrow$ MNIST. MP2, BN, and FC denote 2×2 max-pooling, batch normalization, and a fully-connected layer, respectively.

Layer type	Size / number of filters
convolution	5×5 / 20
MP2 + BN + ReLU	2×2 / 20
convolution + dropout (ratio $= 0.5$)	5×5 / 50
MP2 + BN + ReLU	2×2 / 50
FC + ReLU	1 / 256
FC + softmax	1 / 10

5.4.2 Digit image classification

We examined USPS $\leftrightarrow$ MNIST and SVHN $\rightarrow$ MNIST scenarios. Following the previous studies [Long et al., 2018, Liang et al., 2020a], we used the classical LeNet-5 network for the former scenario as shown in Table 5.3, while a variant of LeNet, called DTN, is used for the latter one as shown in Table 5.4. For both models, we used the last BN layer in the model to calculate the BN statistics matching loss in our method.

Table 5.2 shows the experimental results with the digit recognition datasets. Although our method did not achieve the best performance among the source-free methods, it stably achieved the second highest accuracy in all scenarios. Similarly in the results with Office-31 dataset, our method exceeds the performance of the typical domain adaptation methods in two scenarios, namely USPS$\rightarrow$MNIST and SVHN$\rightarrow$MNIST.

5.4.3 Performance sensitivity to the hyper-parameter and dataset size

We investigated the performance sensitivity of our method to the hyper-parameter setting and that to the size of the target dataset. In this experiment, we used SVHN$\rightarrow$MNIST scenario, and the experimental settings are same with those in the previous experiment unless otherwise noted.

Our method introduces single hyper-parameter, which is λ in Eq. (5.10). We first investigated the performance sensitivity to the value of λ. Since we can only access the unlabeled target data during adaptation, it is essentially hard to appropriately tune the hyper parameter. Therefore, high stability of the performance under a suboptimal setting of the hyper-parameter is required in the source-free domain adaptation. In the experiment, we varied the value of λ from 0.01 to 100 and used it in our method to conduct the adaptation with the digit recognition datasets. Figure 5.3 shows how the test accuracy of the adapted model changes according to the value of λ. In all adaptation scenarios, the performance of our method is quite stable against the change of the value of λ. It keeps almost same within the wide range of the value of λ, specifically $0.2 \leq \lambda \leq 50$.

Table 5.4: The network architectures used in the experiments with SVHN $\rightarrow$ MNIST. BN and FC denote batch normalization and a fully-connected layer, respectively.

Layer type	Size / number of filters
convolution	5×5 / 64 (stride 2 and padding 2)
BN + dropout (ratio $= 0.1$) + ReLU	-
convolution	5×5 / 128 (stride 2 and padding 2)
BN + dropout (ratio $= 0.3$) + ReLU	-
convolution	5×5 / 256 (stride 2 and padding 2)
BN + dropout (ratio $= 0.5$) + ReLU	-
FC + ReLU	1 / 256
FC + softmax	1 / 10

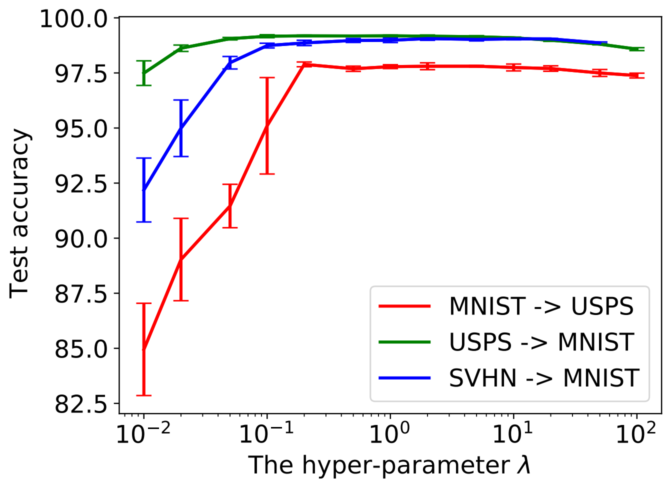


Figure 5.3: Performance sensitivity to the hyper-parameter.

Since our method relies on BN statistics, we also investigated the sensitivity of the performance to the size of a mini-batch. In the previous experiment, we set $B = 64$, which is the same setting as prior works [Li et al., 2020, Liang et al., 2020a], for fair comparison. In this experiment, we varied the mini-batch size from 64 to 320. Table 5.5 shows the test accuracy obtained by our method for each mini-batch size. The performance of our method gets better as the mini-batch size increases. This is because BN statistics with a larger mini-batch represent the feature distribution over a whole dataset more accurately, which results in more effective distributional alignment between domains by matching BN statistics. Therefore, a larger mini-batch is preferred in our method.

Lastly, we investigated the performance of our method in case of small-scale target data. This investigation is crucial, because, considering the motivation of domain adaptation, we cannot always expect sufficiently large amount of the target data. To make the small-scale target data, we randomly selected a subset of the original target

Table 5.5: Performance sensitivity to the mini-batch size.

Batch size	64	128	192	256	320
Test accuracy	99.05%	99.18%	99.26%	99.31%	99.31%

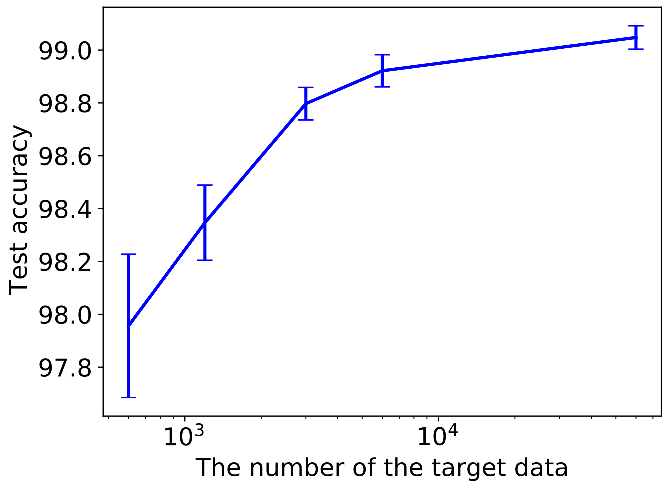


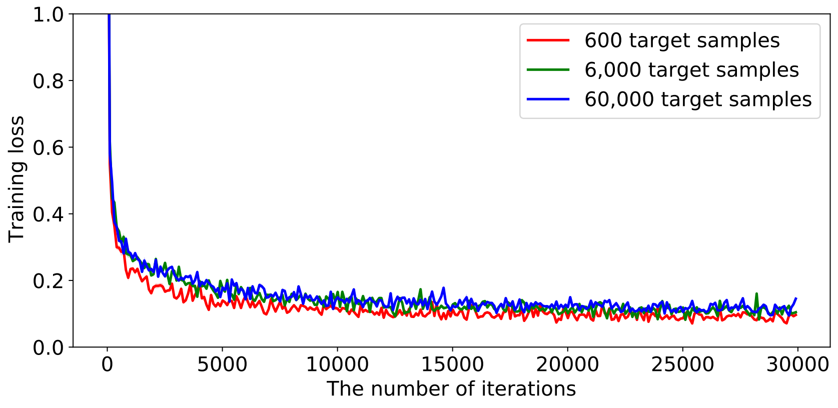
Figure 5.4: Performance sensitivity to the dataset size.

training data while keeping the class prior same with that in the original dataset. Figure 5.4 shows how the test accuracy after the adaptation changes according to the size of the target dataset. As decreasing the number of the target data, the performance of our method becomes deteriorated to some extent. However, even when there are only 600 samples in the target dataset, our method still achieved 98.0%, which is comparable performance with those of the typical domain adaptation methods using full target dataset as well as source dataset.

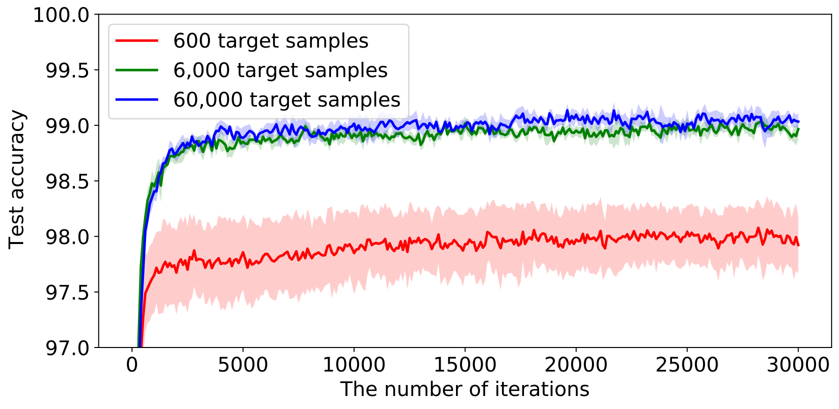
Figure 5.5 shows how the training loss and test accuracy by the model change during adaptation. The accuracy is stably and monotonically improved even when the number of the target data is small. It means that our method can effectively avoid overfitting to the small-scale target dataset.

5.5 Conclusion

We proposed a novel domain adaptation method for source-free setting. To match the distributions between unobservable source data and given target data, we utilize the BN statistics stored in the pretrained model to explicitly estimate and minimize the distributional discrepancy between domains. This approach is quite efficient in terms of computational cost and is also theoretically validated. Experimental results with several benchmark datasets have shown that our method performs well even though it does not require the access to the source data. Moreover, its performance was empirically quite stable against suboptimal hyper-parameter setting or limited size of



(a) Training loss.



(b) Test accuracy.

Figure 5.5: Training loss and test accuracy curves during adaptation in our method.

the target dataset. In concolusion, we argue that our method is quite promising to tackle many real-world problems that are hard to solve with existing domain adaptation methods.

Chapter 6

Conclusion and Future Works

In this chapter, we conclude our contributions in this dissertation and discuss future directions to make domain adaptation technique more useful for real-world applications.

6.1 Conclusion

Machine learning has become an indispensable technology to provide intelligent products, systems, and services that can enrich people's lives. This huge success cannot be achieved without the availability of large-scale training datasets, and the standardization of such datasets has substantially accelerated the research and development of machine learning technology. However, in many practical situations, we cannot always expect a large-scale training dataset, which substantially limits the applicability of machine learning technology to diverse real-world problems. To tackle this issue, this dissertation contributed to developing domain adaptation algorithms that can effectively work with scarce data.

We presented three contributions in this dissertation as summarized below.

- **Partially Zero-shot Domain Adaptation from Incomplete Target Data with Missing Classes**
  In Chapter 3, we introduced a new problem setting, called partially zero-shot domain adaptation. In this setting, a certain subset of classes is missing in target data, but all classes are to be discriminated after adaptation. Our method adopts a new instance-weighting strategy to take unobservable missing-class target data into account during minimization of the distributional discrepancy between domains. The experimental results with several benchmark datasets showed that our method performs better than existing methods.

- **Zero-shot Domain Adaptation based on Attribute Information**
  In Chapter 4, we tackled zero-shot domain adaptation in which no target data are available for adaptation. We first clarified a possible scenario where we can assume availability of some knowledge instead of target data, which enables us to effectively conduct domain adaptation. In this scenario, a domain

shift is caused by a prior change of a specific factor, called an attribute, and we are given the knowledge on how the prior changes between the source and target domains. We reformulated instance-weighting based domain adaptation to utilize the attribute prior instead of target data. Our theoretical analysis and empirical evaluations showed that the proposed method outperforms baseline methods and achieves effective adaptation, though it does not use any target data.

- **Source-free Domain Adaptation via Distributional Alignment by Matching Batch Normalization Statistics**
  In Chapter 5, we tackled source-free domain adaptation in which no source data are available during adaptation. Here, we assumed that a model pretrained with source data is given and also assumed that batch normalization is conducted at least once in the pretrained model. We utilized batch normalization statistics stored in the pretrained model to approximate the distribution of unobserved source data. This approximation makes it possible to explicitly evaluate the distributional discrepancy between domains without access to source data, and our method conducts domain adaptation by minimizing this discrepancy. Experimental results with several benchmark datasets showed that our method achieves competitive performance with state-of-the-art domain adaptation methods, though it does not require access to source data during adaptation.

Our proposed algorithms are designed with a common perspective of matching data distributions between domains, though it was substantially challenging due to data scarcity that we focused on throughout this dissertation. The experimental results with several benchmark datasets have demonstrated their advantages. Therefore, we conclude that we have succeeded in making it possible to apply domain adaptation to more diverse situations that is conceivable in real-world problems.

6.2 Future work towards more practical domain adaptation

In this section, we describe several subjects that are worth investigating in our future work towards more practical domain adaptation.

6.2.1 Domain adaptation with biased target data

In Chapter 3, we focused on the situation in which several classes do not appear in given target data but need to be discriminated in the inference phase. From another perspective, in this situation, the given target data do not exactly follow the target distribution where a model should be generalized. Here, we can consider more general situation — there is a distributional discrepancy between given target data and unseen "ground-truth" target data. Since retrieval of target data is often limited in practice, such a situation would occur in real-world applications [Peng et al., 2017]. We expect that the idea of our zero-shot domain adaptation method presented in Chapter 4 can be combined with standard domain adaptation to tackle this problem by reducing the distributional discrepancy between seen and unseen target data.

6.2.2 Cross-task domain adaptation

In this dissertation, we assumed that both source and target data are obtained to tackle a common classification task. Specifically, both source and target distributions have a common support in the label space, which is necessary to guarantee that a model trained with adapted source data works well in the target domain [Ben-David et al., 2010]. However, this assumption is violated, if the source dataset is constructed for a different task from the target task. This challenging situation has been tackled in very recent studies [You et al., 2019, Kundu et al., 2020], but a simple yet strong solution is still to fine-tune a model with labeled target data after pretraining with source data [Zamir et al., 2018]. Since the source distribution would not be essentially matched to the target distribution, our methods cannot be simply applied in this situation. We expect that feature disentanglement techniques [Locatello et al., 2019] would enable us to extract task-invariant features to be matched between domains and would make it possible to utilize our algorithms based on these features.

6.2.3 Pretraining for adaptation with scarce data

In Chapter 5, we tackled how to fine-tune a pretrained model with unlabeled target data. There, we did not require any specific type of training for pretraining, which made our method widely applicable. On the other hand, if we aim to obtain a pretrained model that can be effectively adapted to any target domain, it may be possible to design a specific pretraining method for this purpose. A similar problem has been tackled in domain generalization [Li et al., 2018a], and it mainly focuses on designing how to train a model to make it work in an unseen target domain. Considering that batch normalization can be utilized for source-free domain adaptation as shown in Chapter 5, we can also design a new model architecture for this purpose, for example, by using neural architecture search techniques [Elsken et al., 2019].

6.2.4 Explainable domain adaptation

When a given dataset is scarce, it is essentially difficult to precisely validate the performance of the trained model, though it is quite important in practice to confirm whether the training goes well or not. In this dissertation, we simply conducted cross-validation or demonstrated high stability of performance against suboptimal hyperparameter settings to empirically show the practicality of our methods. On the other hand, recent studies on explainable artificial intelligence (XAI) [Arrieta et al., 2020] proposed another approach to confirm the validity of machine learning process, in which they tried to provide intuitive information that explains how the model predicts the output. In Chapters 3 and 4, we adopted instance weights for adaptation, which can be understandable as sample-wise importance in the adaptation process. From this perspective, it might be possible to extend our method by adopting feature-wise importance that can provide finer information to users on how the data are adapted, and user feedbacks can be reflected to the adaptation process by editing such importance.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Professor Masashi Sugiyama. He provides me an excellent research environment and consistently advises me to conduct significant research, which makes my experience in a Ph.D. program quite valuable. I would like to thank Professor Issei Sato and Professor Junya Honda for valuable discussions and insightful comments that further strengthen my research. I would like to thank Ms. Yuko Kawashima and all the members of Professor Sugiyama's, Professor Sato's, and Professor Honda's laboratories for their help in my research.

I started a Ph.D. program while working at NEC. I would like to thank Dr. Atsushi Sato for giving me this great opportunity. I would like to also thank Dr. Takashi Shibata for encouraging me to enter a Ph.D. program.

My research project was partially supported by RIKEN AIP-NEC Collaboration Center. I would like to thank Professor Takashi Takenouchi for giving me a lot of valuable advises and comments. During working at RIKEN AIP, I often had casual but fruitful discussions with Mr. Akira Tanimoto. I would like to thank the staff in RIKEN AIP for their help in my research activities.

I would like to express my gratitude to the members of the defense committee: Professor Imari Sato, Professor Shinpei Kato, Professor Naoto Yokoya, Professor Shin'ichi Sato, Professor Toshihiko Yamasaki, Professor Yusuke Miyao, for serving as the committee members in spite of their busy schedule and providing a lot of valuable comments and feedbacks.

Last but not least, I would like to thank my wife Misaki for her dedication and support, which makes my focusing on study and research feasible. I wish my research will contribute in some way to a bright future of our sweet sons, Akito and Takuma.

References

Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020.

Mahsa Baktashmotlagh, Masoud Faraki, Tom Drummond, and Mathieu Salzmann. Learning factorized representations for open-set domain adaptation. In *International Conference on Learning Representations*, 2018.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

Sagie Benaim and Lior Wolf. One-sided unsupervised domain mapping. In *Advances in neural information processing systems*, pages 752–762, 2017.

Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. In *Advances in Neural Information Processing Systems*, pages 343–351. 2016.

Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Michael I. Jordan. Partial transfer learning with selective adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2724–2732, 2018a.

Zhangjie Cao, Lijia Ma, Mingsheng Long, and Jianmin Wang. Partial adversarial domain adaptation. In *European Conference on Computer Vision*, pages 139–155, 2018b.

Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

Woong-Gi Chang, Tackgeun You, Seonguk Seo, Suha Kwak, and Bohyung Han. Domain-specific batch normalization for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7354–7362, 2019.

Qingchao Chen, Yang Liu, Zhaowen Wang, Ian Wassell, and Kevin Chetty. Re-weighted adversarial adaptation network for unsupervised domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7976–7985, 2018.

Zhiyuan Chen and Bing Liu. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–207, 2018.

Boris Chidlovskii, Stephane Clinchant, and Gabriela Csurka. Domain adaptation in the absence of source domain data. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 451–460, 2016.

Corinna Cortes, Mehryar Mohri, Michael Riley, and Afshin Rostamizadeh. Sample selection bias correction theory. In *International conference on algorithmic learning theory*, pages 38–53, 2008.

Corinna Cortes, Yishay Mansour, and Mehryar Mohri. Learning bounds for importance weighting. In *Advances in neural information processing systems*, pages 442–450, 2010.

Nicolas Courty, Rémi Flamary, Amaury Habrard, and Alain Rakotomamonjy. Joint distribution optimal transportation for domain adaptation. In *Advances in Neural Information Processing Systems*, pages 3730–3739, 2017.

Imre Csiszar and János Körner. *Information theory: coding theorems for discrete memoryless systems*. Cambridge University Press, 2011.

Gabriela Csurka, editor. *Domain Adaptation in Computer Vision Applications*. Advances in Computer Vision and Pattern Recognition. Springer, 2017.

Raj Dabre, Chenhui Chu, and Anoop Kunchukuttan. A survey of multilingual neural machine translation. *ACM Computing Surveys (CSUR)*, 53(5):1–38, 2020.

Bharath Bhushan Damodaran, Benjamin Kellenberger, Rémi Flamary, Devis Tuia, and Nicolas Courty. Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 447–463, 2018.

Zhijie Deng, Yucen Luo, and Jun Zhu. Cluster alignment with a teacher for unsupervised domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9944–9953, 2019.

Eran Eidinger, Roee Enbar, and Tal Hassner. Age and gender estimation of unfiltered faces. *IEEE Transactions on Information Forensics and Security*, 9(12):2170–2179, 2014.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20:1–21, 2019.

Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117, 2004.

Basura Fernando, Amaury Habrard, Marc Sebban, and Tinne Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *IEEE International Conference on Computer Vision*, pages 2960–2967, 2013.

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*, 17(1): 2096–2030, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680. 2014.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning — Data Mining, Inference, and Prediction*. Springer, second edition, 2009.

William Grant Hatcher and Wei Yu. A survey of deep learning: Platforms, applications and emerging research trends. *IEEE Access*, 6:24411–24432, 2018.

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*, pages 770–778, 2016.

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*, pages 1989–1998. PMLR, 2018.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018a.

Weihua Hu, Gang Niu, Issei Sato, and Masashi Sugiyama. Does distributionally robust supervised learning give robust classifiers? In *International Conference on Machine Learning*, pages 2034–2042, 2018b.

Jiayuan Huang, Arthur Gretton, Karsten M Borgwardt, Bernhard Schölkopf, and Alex J Smola. Correcting sample selection bias by unlabeled data. In *Advances in Neural Information Processing Systems*, pages 601–608, 2007.

Aapo Hyvärinen and Petteri Pajunen. Nonlinear independent component analysis: Existence and uniqueness results. *Neural networks*, 12(3):429–439, 1999.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

Takashi Ishida, Gang Niu, and Masashi Sugiyama. Binary classification from positive-confidence data. In *Advances in Neural Information Processing Systems*, pages 5921–5932. 2018.

Xiang Jiang, Qicheng Lao, Stan Matwin, and Mohammad Havaei. Implicit class-conditioned domain alignment for unsupervised domain adaptation. *International Conference on Machine Learning*, 2020.

Takafumi Kanamori, Shohei Hido, and Masashi Sugiyama. A least-squares approach to direct importance estimation. *Journal of Machine Learning Research*, 10:1391–1445, 2009.

Ira Kemelmacher-Shlizerman, Steven M Seitz, Daniel Miller, and Evan Brossard. The megaface benchmark: 1 million faces for recognition at scale. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4873–4882, 2016.

Yassin Kortli, Maher Jridi, Ayman Al Falou, and Mohamed Atri. Face recognition systems: A survey. *Sensors*, 20(2):342, 2020.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105. 2012.

Jogendra Nath Kundu, Naveen Venkat, R Venkatesh Babu, et al. Universal source-free domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4544–4553, 2020.

Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, et al. The open images dataset v4. *International Journal of Computer Vision*, pages 1–26, 2020.

Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Fei-Fei Li, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611, 2006.

Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain generalization with adversarial feature learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5400–5409, 2018a.

Jingjing Li, Erpeng Chen, Zhengming Ding, Lei Zhu, Ke Lu, and Zi Huang. Cycle-consistent conditional adversarial transfer networks. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 747–755, 2019.

Rui Li, Qianfen Jiao, Wenming Cao, Hau-San Wong, and Si Wu. Model adaptation: Unsupervised domain adaptation without source data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9641–9650, 2020.

Yanghao Li, Naiyan Wang, Jianping Shi, Xiaodi Hou, and Jiaying Liu. Adaptive batch normalization for practical domain adaptation. *Pattern Recognition*, 80:109–117, 2018b.

Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International Conference on Machine Learning*, 2020a.

Jian Liang, Yunbo Wang, Dapeng Hu, Ran He, and Jiashi Feng. A balanced and uncertainty-aware approach for partial domain adaptation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020b.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *international conference on machine learning*, pages 4114–4124. PMLR, 2019.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pages 97–105. PMLR, 2015.

Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I. Jordan. Deep transfer learning with joint adaptation networks. In *International Conference on Machine Learning*, pages 2208–2217, 2017.

Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In *Advances in Neural Information Processing Systems*, pages 1640–1650, 2018.

Mishaim Malik, Muhammad Kamran Malik, Khawar Mehmood, and Imran Makhdoom. Automatic speech recognition: a survey. *Multimedia Tools and Applications*, pages 1–47, 2020.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

George A Miller. *WordNet: An electronic lexical database*. MIT press, 1998.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Tom Michael Mitchell. *The discipline of machine learning*, volume 9. Carnegie Mellon University, School of Computer Science, Machine Learning ⋯, 2006.

Saeid Motiian, Quinn Jones, Seyed Iranmanesh, and Gianfranco Doretto. Few-shot adversarial domain adaptation. In *Advances in Neural Information Processing Systems*, pages 6670–6680, 2017.

Zak Murez, Soheil Kolouri, David Kriegman, Ravi Ramamoorthi, and Kyungnam Kim. Image to image translation for domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4500–4509, 2018.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22:1345–1359, 2010.

Pau Panareda Busto and Juergen Gall. Open set domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 754–763, 2017.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

Kuan-Chuan Peng, Ziyan Wu, and Jan Ernst. Zero-shot deep domain adaptation. In *European Conference on Computer Vision*, pages 793–810, 2018.

Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko. Visda: The visual domain adaptation challenge, 2017.

Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. *Dataset shift in machine learning*. The MIT Press, 2009.

Y. Roh, G. Heo, and S. E. Whang. A survey on data collection for machine learning: A big data - ai integration perspective. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2019. doi: 10.1109/TKDE.2019.2946162.

Bernardino Romera-Paredes and Philip H. S. Torr. An embarrassingly simple approach to zero-shot learning. In *International Conference on Machine Learning*, volume 37, pages 2152–2161, 2015.

Artem Rozantsev, Mathieu Salzmann, and Pascal Fua. Beyond sharing weights for deep domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 41(4):801–814, 2018.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *European Conference on Computer Vision*, pages 213–226, 2010.

Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. Asymmetric tri-training for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 2988–2997, 2017.

Kuniaki Saito, Shohei Yamamoto, Yoshitaka Ushiku, and Tatsuya Harada. Open set domain adaptation by backpropagation. In *European Conference on Computer Vision*, pages 153–168, 2018.

Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000.

Rui Shu, Hung Bui, Hirokazu Narui, and Stefano Ermon. A dirt-t approach to unsupervised domain adaptation. In *International Conference on Learning Representations*, 2018.

Amos J Storkey and Masashi Sugiyama. Mixture regression for covariate shift. In *Advances in Neural Information Processing Systems*, pages 1337–1344, 2007.

Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul V Buenau, and Motoaki Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in Neural Information Processing Systems*, pages 1433–1440, 2008.

Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. *Density ratio estimation in machine learning*. Cambridge University Press, 2012.

Baochen Sun, Jiashi Feng, and Kate Saenko. Correlation alignment for unsupervised domain adaptation. In *Domain Adaptation in Computer Vision Applications*, pages 153–171. Springer, 2017.

Takeshi Teshima, Issei Sato, and Masashi Sugiyama. Few-shot domain adaptation by causal mechanism transfer. In *International Conference on Machine Learning*, 2020.

Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2962–2971, 2017.

Roberto Vezzani and Rita Cucchiara. Video surveillance online repository (visor): an integrated framework. *Multimedia Tools and Applications*, 50(2):359–380, 2010.

Paul Voigt and Axel von dem Bussche. *The EU General Data Protection Regulation (GDPR): A Practical Guide*. Springer Publishing Company, Incorporated, 1st edition, 2017. ISBN 3319579584.

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

Ximei Wang, Ying Jin, Mingsheng Long, Jianmin Wang, and Michael I Jordan. Transferable normalization: Towards improving transferability of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 1953–1963, 2019.

Junfeng Wen, Russell Greiner, and Dale Schuurmans. Correcting covariate shift with the frank-wolfe algorithm. In *Twenty-fourth International Joint Conference on Artificial Intelligence*, 2015.

Garrett Wilson and Diane J Cook. A survey of unsupervised deep domain adaptation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(5):1–46, 2020.

Rui Xia, Zhenchun Pan, and Feng Xu. Instance weighting for domain adaptation via trading off sample selection bias and variance. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4489–4495. AAAI Press, 2018.

Xiang Xu, Xiong Zhou, Ragav Venkatesan, Gurumurthy Swaminathan, and Orchid Majumder. d-sne: Domain adaptation using stochastic neighborhood embedding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2497–2506, 2019.

Hongliang Yan, Yukang Ding, Peihua Li, Qilong Wang, Yong Xu, and Wangmeng Zuo. Mind the class weight bias: Weighted maximum mean discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2272–2281, 2017.

Yongxin Yang and Timothy Hospedales. Zero-shot domain adaptation via kernel regression on the grassmannian. In *International Workshop on Differential Geometry in Computer Vision for Analysis of Shapes, Images and Trajectories*, pages 1.1–1.12, 2015.

Kaichao You, Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Universal domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2720–2729, 2019.

Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3712–3722, 2018.

Jing Zhang, Zewei Ding, Wanqing Li, and Philip Ogunbona. Importance weighted adversarial nets for partial domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8156–8164, 2018.

Kun Zhang, Bernhard Schölkopf, Krikamol Muandet, and Zhikun Wang. Domain adaptation under target and conditional shift. In *International Conference on Machine Learning*, pages 819–827, 2013.

Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael Jordan. Bridging theory and algorithm for domain adaptation. In *International Conference on Machine Learning*, pages 7404–7413, 2019.

Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1452–1464, 2017.

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*, pages 2223–2232, 2017.