AN ADVERSARIAL APPROACH TO IMPORTANCE WEIGHTING FOR DOMAIN ADAPTATION

AN ADVERSARIAL APPROACH TO IMPORTANCE WEIGHTING FOR DOMAIN ADAPTATION

BY

NIMA MASHAYEKHI, B.Sc.

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTING AND SOFTWARE

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

Master of Science

Master of Science (2022) McMaster University

(computing and software) Hamilton, Ontario, Canada

TITLE:	An Adversarial Approach to Importance Weighting for Domain Adaptation
AUTHOR:	Nima Mashayekhi B.Sc., Electrical Engineering, Amirkabir University of Technology, Tehran, Iran
SUPERVISOR:	Dr. Hassan Ashtiani
NUMBER OF PAGES:	x, 47

Abstract

Most supervised learning methods assume that the training and test data points are generated from the same distribution. Therefore, they face major challenges when this assumption does not hold. Domain adaptation techniques aim to address this issue by adapting the learned models to new distributions. Covariate shift is a common assumption in domain adaptation, where the training and test distributions only differ in the marginal distributions. A common idea to tackle covariate shift is estimating the importance weights of the training data points using unlabeled data from the source and target distributions and then training the classifier using importance-weighted risk minimization. Existing methods for estimating the importance weights are kernel-based which scale poorly with dataset size and underperform on high-dimensional data. This work proposes a novel method for estimating the importance weights using generative adversarial networks. There are two neural networks used in this framework which we call them the weighting and the discriminator networks. These networks are jointly trained using an adversarial learning scheme. We designed a benchmark for assessing the performance of classification under various forms of distribution shift and evaluated our method in this framework. We observe that while our method is effective in estimating the importance weights, the improvements we get in the domain adaptation task depends on the discrepancy of the domains.

To my parents and my dear sister

Acknowledgements

Above all, I would like to express my appreciation and gratitude to my supervisor, Dr. Hassan Ashtiani. His continuous guidance, support, and mentorship have been exceptional. I am grateful that I was able to work under his supervision and learn so many valuable lessons.

I would like to thank the committee members, Dr. Rong Zheng and Dr. Fei Chiang for dedicating their time to this thesis.

I would like to thank my labmates, Qing Liu, Ishaq Aden-Ali, and Alireza Fatholl-ah Pour for their help and friendship. Their constant support helped me through my research process.

Last but not least, I would like to thank my parents for always encouraging and supporting me.

Contents

List of Figures

List of Tables

Notation, Definitions, and Abbreviations

Abbreviations

DA	Domain Adaptation
ERM	Empirical Risk Minimization
IW-ERM	Importance weighted Empirical risk minimization
GAN	Generative Adversarial Network

Chapter 1

Introduction

1.1 Motivation

Supervised Machine Learning algorithms have been highly successful recently, however their success relies on the fact that training and test data comes from the same distribution. A big challenge of machine learning algorithms is distribution shift (or dataset shift), where training and test data come from different distributions. Therefore, many *Domain Adaptation* methods have been proposed in the literature to reduce the harmful effect of distribution shift. An example of distribution shift is shown in figure 1.1.

1.2 Concrete Examples of Domain Adaptation

The distribution shift that we discussed above can have several forms. We study various types of distribution shifts that can be present in a classification problem. In this section we give concrete examples of the main types of shifts that we consider in

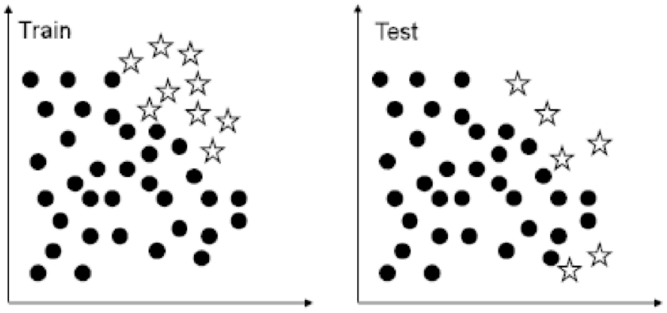


Figure 1.1: Distribution shift [52]

this thesis.

Assume that we have trained a machine learning algorithm which is a cancer predictor. The input of this predictor is a medical image (denoted by x) and the output is a binary label (denoted by $y \in \{0, 1\}$) which predicts whether a physician diagnoses the patient with cancer ($y = 1$) or not ($y = 0$). Assume that the model works well in practice. However, If we deploy this model in a different country with lower quality of life (e.g., more air pollution) we may notice a significant drop in the performance of the cancer predictor. This is due to the fact that the probability of having cancer is higher in the latter country (i.e., $p(y = 1)$ is higher in the latter country). However, the distribution of the healthy $p(x|y = 0)$ and cancerous $p(x|y = 1)$ images may be the same across the countries. We call this type of shift as *Label Shift*.

As an another form of the distribution shift, assume that we have trained a neural network for handwritten digit recognition using black and white images. Can we also use this model on colorful images of digits? In this case $p(x|y)$ is not fixed anymore unlike the previous setting. For instance, in figure 1.2, an example of this kind of shift is shown. The upper row contains instances from the MNIST data set [38] and the lower row instances are from the MNIST-M data set [18] which are obtained from

the BSDS500 data set [1]. There are many techniques for *adapting* our model to the a new domain which we call them *Manifestation Shift*.

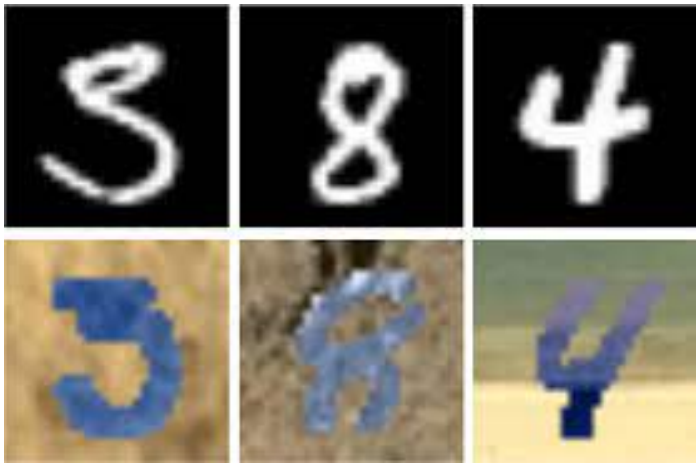


Figure 1.2: An example of manifestation shift [18]

There are also some other forms of distribution shift. Consider a convolutional neural network which is supposed to classify pictures of cats versus dogs. Assume there are two datasets for this model and both of them have the same ratio of cats to dogs. Therefore, there is no "label shift" between the two datasets. Also, assume the pictures of the two datasets "look alike", hence we do not have manifestation shift. However, different breeds of cats and dogs are significantly over/under-represented across these two datasets. This shift which we call *subpopulation shift* can also decrease the accuracy of the classifier.

1.3 A theoretical view on Domain Adaptation

In order to have a better understanding of domain adaptation, it is useful to know how the target error is bounded. Ben-David *et al.* [7] have developed the following bound on the target domain performance of a classifier which is trained on the source

domain.

$$\epsilon_T(h) \leq \epsilon_S(h) + d(p,q) + min(\mathbb{E}_p[|f_S(x) - f_T(x)|], \mathbb{E}_q[|f_S(x) - f_T(x)|]) \quad (1.3.1)$$

In equation 1.3.1, p and q are the source and target distributions, respectively. $\epsilon_S(h)$ and $\epsilon_T(h)$ are source and target errors. Also, $f_S(x)$ and $f_T(x)$ denote source and target labelling functions.

In this equation, the third term is the difference of labelling functions between the source and target domains and it is expected to be small. However, the problem is the second term which is the divergence between source and target domain. The difference between distributions is usually measured by variation divergence (equation 1.3.2). But the use of variation divergence has two disadvantages. First, when we are given finite samples of some arbitrary distributions, this measure cannot be accurately estimated[35, 6]. Second, it unnecessarily inflated the bound. Therefore, other measures are used such as $\mathcal{H}$-divergence [7], which addresses these issues.

$$d(p,q) = 2 \sup_{B \in \mathcal{B}} |Pr_p[B] - Pr_q[B]| \quad (1.3.2)$$

1.4 Domain Adaptation and other related research areas

There are several research fields closely related to domain adaptation, which will be introduced in this section.

Transfer learning is a quite similar area to domain adaptation. In transfer

learning the goal is to improve the performance of a trained model on a different but related domain or task [59]. The key difference between domain adaptation and transfer learning is that in transfer learning the target labels may be different from source labels (in a classification setting), but in domain adaptation the labels are the same and only the distributions differ. A common approach for transfer learning is reweighting some data of the source domain for use in the target domain [62, 48, 29, 8]. There is another approach for transfer learning which is also used similarly in domain adaptation and it is finding a good feature representation such that it reduces the discrepancy between the source and target [49, 2, 3, 41, 33].

Multi-task learning is another related field which optimizes a model on several related tasks. This will also make the model generalize better on the original task, however multi-task learning's goal is not improving the generalization to new tasks ot unseen data [12]. **Multi-domain learning** is similar to multi-task learning and it trains a model on various related domains [66].

Meta learning, aims to observe how different machine learning models perform on various learning tasks, and then learn from this experiences, in order to learn new tasks much faster and more efficient. [57, 27, 56]. There is an application of meta learning in domain adaptation which tries to *meta-learn* the initial conditions of existing DA algorithms [42].

Lifelong learning aims to learn from multiple sequential tasks or domains. In this approach, the model learns over time by gaining new knowledge while retraining previously learned experiences. [9, 47]

Zero-shot learning is another related research area and it aims at learning models from seen classes and classify samples whose categories are unseen in training

[60, 34].

1.5 Domain Adaptation using Importance Sampling

In Domain Adaptation, when source and target data share support, a common idea is to use importance-weighted empirical risk minimization(IW-ERM)[11]. The idea is to correct the loss function by assigning proper weights to target data, or equivalently upweighting/downweighting the data regarding their "importance" to the target domain. Existing methods for IW-ERM are mostly kernel-based which scale poorly with dataset size and underperform on high-dimensional data [22]. The goal of this work is to use neural networks to estimate these weights. Neural networks have shown promising results in many applications and the use of neural networks to estimate weights in IW-ERM is not studied before. Training the neural network used in this work is inspired by Generative Adversarial Networks (GANs) and is explained in detail in the next chapters.

1.6 Contributions

In this thesis, we make the following contributions:

- The problems in Domain Adaptation are usually divided into two groups: Label Shift and Covariate Shift. The assumptions used in each of these group are different to each other and they require different methods to tackle them. We further define two separate groups in Covariate Shift problems: Manifestation

Shift and Population shift. The methods used in each of these groups are also different to one another. In the next chapter we introduce some techniques used to solve the problems in each group.

- We have designed a benchmark using the CIFAR-10 dataset to simulate distribution shifts. These shifts can be in the shape of label shift, subpopulation shift, etc. Having these shifts in our training and test datasets, we can then try various methods including importance weighting techniques and evaluate them.

- We have proposed a new framework based on importance-weighted empirical risk minimization. The core of the framework is weight learning and the goal is to assign proper weights to the instances from the source domain. For example, the instances that are underrepresented in the source domain (compared to the target domain) should get relatively higher weights. There is an adversarial learning scheme in our framework using neural networks and it is inspired from Generative Adversarial Networks. The use of neural networks for learning weights was not studied in previous works.

- The proposed framework is implemented in Python using the PyTorch library. We have done several experiments in different scenarios in order to evaluate our framework. There are two main settings of source and target domains. In the first setting, the domains are relatively separable, and in the second setting they are not separable. We can also adjust the intensity of the shift between domains, using our benchmark. At the end, we compare three cases where is no weighting used, our weighting method is used, and where "ideal weights" are used.

1.7 Organization

The rest of this thesis is organized as follows: The second chapter will be focused on the related work. The third chapter will present the problem formulation and our approach. The fourth will discuss the experimental settings, implementation details and hyperparameter choices. The last chapter will be dedicated to the conclusion and the directions for future work.

Chapter 2

Related Work

In the previous chapter we briefly discussed the problem of Domain Adaptation. In this chapter, we will explain some relevant methods to solve this problem. Domain Adaptation is almost impossible to solve unless we make some assumptions which will be explained in the following. We denote the distribution of the data on which the model is trained (*source* distribution) as p, and the distribution of the data of interest (*target* distribution) as q. We consider the task of classification where the input variable is denoted by x and output variable is denoted by y. It is also assumed that we have labeled data from the source distribution and unlabeled data from the target distribution. These are the two main settings that we consider for domain adaptation:

- **label shift** (also known as *target shift*): it is assumed that the conditional input distribution is invariant (i.e,. $p(x|y) = q(x|y)$), and $p(y)$ changes from source to target (i.e,. $p(y) \neq q(y)$) [5].

- **covariate shift** (also known as *sample selection bias*): in this setting it is

assumed that the conditional output distribution is invariant between source and target distributions ($p(y|x) = q(y|x)$), but the marginal distributions differ ($p(x) \neq q(x)$) [5].

Table 2.1: Two settings of Domain Adaptation

Covariate Shift	Label Shift
$p(x) \neq q(x)$	$p(y) \neq q(y)$
$p(y|x) = q(y|x)$	$p(x|y) = q(x|y)$

2.1 Handling Label Shift

When facing label shift in classification, the most common approach is to estimate the weights $w(y) = q(y)/p(y)$ for every class $y \in \mathcal{Y}$ with training data, and then using the importance-weighted ERM (Empirical Risk Minimization) framework, which solves $\min_{\Theta} \sum_{i=1}^{n} w_i \ell(y_i, x_i)$ where Θ is the set of parameters of the model and ℓ is the loss function. Moreover, the weight of the i-th data point is $w_i = q(x_i, y_i)/p(x_i, y_i)$. Note that because of the label shift assumption, it is equal to define $w_i = q(y_i)/p(y_i)$ [5, 16, 45].

In [45], authors have proposed Black Box Shift Estimation (BBSE) which can detect and quantify the shift, and correct the classifiers without test set labels, using the inverse of the confusion matrix. One issue with this approach is that it is not promising in the low sample setting, because when there are very few samples available, the weight estimation will have a high variance which is not desirable.

Later, the authors of [5] proposed an efficient weight estimator which can yield good statictical guarantees without a requirement on minimum sample complexity

as necessaery for BBSL. Moreover, a novel method is proposed by Sugiyama *et al.* [16] to estimate the class ratio in the test dataset using Expectation Maximization algorithm and f-divergence–approximation. While most of the papers of this are of research are focused on classification, Guo *et al.* [23] have introduced methods for continuous target shift adaptation in regression and conditional density estimation using importance weight estimator for continuous targets.

2.2 Handling Covariate Shift

We can identify two types of covariate shifts which have distinct natures and it is reasonable to use different approaches for handling them.

2.2.1 Manifestation Shift

Sometimes an instance of the data in one domain may have a different representation in other domains. For example, consider a situation where the source domain consists of colorful images but the target domain has black-and-white images. Intuitively, in this example it makes sense to try make the source images black-and-white so that they are consistent with the target domain.

More generally, in manifestation shift we assume that there is a shared representation over which the source and the target domains have similar distributions. Therefore, the main task becomes finding the appropriate shared representation.

2.2.2 Population Shift

In this case, the marginal distribution of source and target are different. More specifically, the source distribution overrepresents or underrepresents some parts of the target distribution. For example, it may be the case that the images of women are more frequent in the source domain, whereas men are more common in the target domain. Intuitively, one could try to tackle this type of shift by *reweigting* the source samples and then train a classifier on the weighted source samples.

2.3 Techniques used for Domain Adaptation

In this part, we introduce general methods which are used for tackling various forms of distribution shift.

2.3.1 Handling Manifestation Shift

The main challenge in manifestation shift is fining a shared representation between the source and the target distribution. To achieve this goal, we can either find a mapping from one domain to another [26, 67], or extract domain-invariant features over which the distributions of the source and target are similar [18].

But how can we find such a mapping (or mappings)? Intuitively the distribution of source and target samples should be similar over the shared space. One of the most promising approaches to find a shared representation is adversarial learning. On a high level, we find a space over which the samples of the source and target are "indistinguishable". More specifically, an *adversary* (a classifier) tries distinguish between source and target samples. If the adversary is unsuccessful, then we have a

domain-invariant representation.

One of the simplest yet promising methods in this area is Adversarial Discriminative Domain Adaptation (ADDA) [54] which is inspired from Generative Adversarial Networks (GAN) [21] and consists of two steps: first learning a discriminative representation using the labels in the source domain and then a separate encoding that maps the target data to the same space using an asymmetric mapping learned through a domain-adversarial loss.

One of the most well known methods for handling manifestation shift is domain-adversarial training of neural networks [16] and its overall sketch is depicted in figure 2.1. This approach promotes the emergence of the features that are (i) discriminative for the main learning task on the source domain (blue part of the figure 2.1) and (ii) indiscriminative with respect to the shift between the domains.

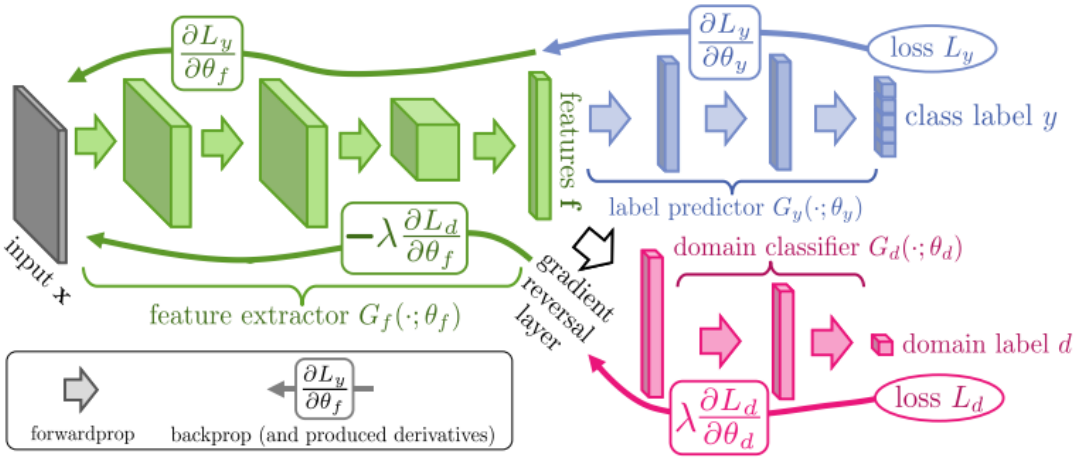


Figure 2.1: domain-adversarial training [16]
The objective of this method is to learn features which are not only discriminative enough to classify the data but also are indiscriminative across various domains.

Most of the authors in this area assume the only shift they are faced with is the manifestation shift, but the authors of [43] consider the case when we are faced

with both manifestation shift and label shift. They have proposed the method called Domain Adversarial nets for Target Shift (DATS) to address label shift while learning a domain invariant representation. As we can see in figure 2.2, except the bottom red part, the rest of the framework resembles the previous method.

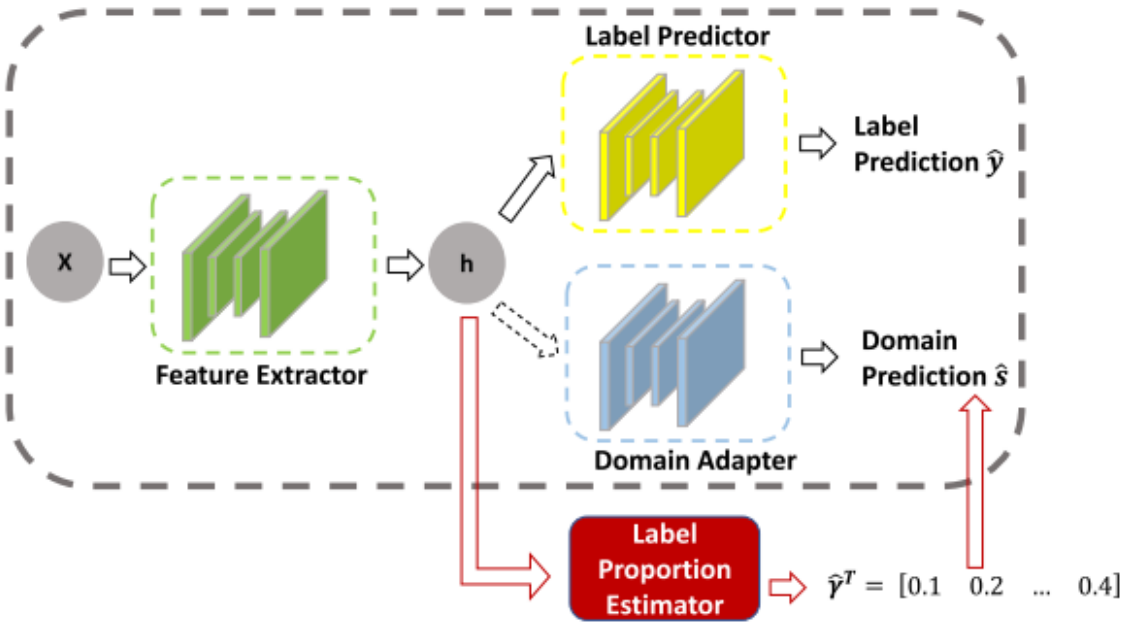


Figure 2.2: DATS framework [43]
This method is similar to the previous method (domain-adversarial training). The first part of the model (Label predictor) is to extract disciminative features. The second part (Domain Adapter) tries to make the features more indiscriminative across domains, and the last part (Label Proportion Estimator) assigns appropriate weights to each class in order to deal with label shift.

2.3.2 Handling Population Shift

We can tackle this issue by *reweigting* the source samples. One of the most famous reweighting schemes is kernel mean-matching [22] which tries to re-weight the source data such that its distribution more closely matches the distribution of the target

data. Another well-known approach is *importance sampling* and some examples of this method are importance-weighted risk minimization [51] and KL importance estimation [53]. However, reweighting and importance sampling methods can be generally unsuccessful if the source and target distributions are far apart. Also, kernel methods do not work well for high dimensional data, reweighting using kernels may not be ideal.

2.3.3 Reweighting using neural networks

Importance-weighed risk minimization is a key part in many machine learning algorithm including deep learning. In [11] the effect of importance weighting in deep neural networks has been studied and it is shown that it can be helpful while facing distribution shift. In [50] a meta-learning algorithm is proposed that learns to assign weights to training examples based on their gradient direction, but the drawback of this approach is that it requires a clean and unbiased validation dataset.

A technique often used when handling distribution shift is first weight estimation, and second weighted classification. In [17] this technique is used and weights are calculated by kernel mean matching [22]. Actually, there is no methods for reweighting using neural networks in the literature and it is a motivation to propose an algorithm using these tools.

2.3.4 Bayesian Priors and Fine Tuning

A popular heuristic for domain adaptation and more generally transfer learning is finding a solution with respect to the source only, and then using this solution to narrow down the search for the solution of the target domain. One example of this

approach is training a neural network with the data from the source domain, and then "fine tuning" the weights of this network using the samples from the target domain [64, 31]. A related Bayesian approach is using the source solution (posterior) as a prior for the target task [15, 19]. A drawback of these approaches is that in order to succeed, they require the solution to the first phase to be close to the solution for the target domain. Moreover, these methods need labeled data from the target distribution which can be considered as a limitation.

2.3.5 Data Augmentation

Data augmentation is a heuristic for domain adaptation which is also used in order to increase the robustness of learning systems. The main idea is to augment the source data set with new instances that are often generated by adding structured noise to the source instances. For example, when the source domain of a natural language processing task is a news data set and the target is the social media, it would make sense to inject the type of noise common in social media to the source instances [55]. However, unlike previous methods, these methods rely highly on domain knowledge (i.e,. images, text, ...) and they are not intented for adapting to a specific domain.

2.4 Generative Adversarial Networks

Most of the Machine Learning models can be classified into discriminative or generative models. Recently, with the use of neural networks, there is a new research area called Generative Adversarial Networks (GANs)[21]. Theoretically, a GAN does an unsupervised task of observing some data and trying to produce synthetic data as

similar as possible to them. There are two neural networks in GANs. One of them is called the generator network, and the other one is called the discriminator network. The discriminator is a binary classifier which tries to distinguish real data from the synthetic (or fake) data. On the contrary, the generator tries to confuse discriminator by generating realistic data. These two networks are trained simultaneously and they challenge each other. At the end, if trained properly, generator generates realistic data which resembles the original data, and discriminator becomes good at predicting the fake ones.

Practically, GANs have been successful in various applications including but not limited to character generation [36, 24], image blending [61, 13], speech synthesis [44, 28] and music generation [65].

Moreover, image-to-image translation is an active line of research which is highly related to our problem. Here, the aim of the networks is to learn a mapping from one domain to another domains [58, 63, 14, 46, 30, 39, 40].

Despite all the successes of the GANs, it is worth to mention that training GANs is a relatively difficult task. The most common challenges are mode collapse [4], vanishing gradient [20], internal covariate shift [32] and lack of proper evaluation metrics [10].

Chapter 3

Approach

In this chapter, we first focus on the formal definition of the problem and then present the approach for solving the problem. Our goal is to introduce a reweighting method using neural networks which can be used in settings including label shift and population shift.

3.1 Problem Formulation

Suppose we want to perform a task of classification and we are facing distribution shift between the source and target distributions. Note that we have access to labelled data from source, and unlabelled data from target distribution and the goal is to classify the target data. We denote the distribution of source as p and the distribution of target as q.

As discussed in the previous chapter, a common approach is to use Weighted ERM (Empirical Risk Minimization), instead of the ERM, which solves $\min_\Theta \sum_{i=1}^{n} w_i \ell(y_i, f_\Theta(x_i))$ where f is the predictor (e.g. a neural network) parameterized by θ and ℓ is the loss

function.

If the distribution shift which we are dealing with is label shift, then the ideal weights (i.e. w_i s) are $w(y) = q(y)/p(y)$ for every class $y \in \mathcal{Y}$ with training data. However, the challenge is that it is assumed that the target data are without labels and it is not straightforward to calculate the weights. A more general and harder case is population shift, where the instances with the same label does not necessarily share the same weight.

Generally, the goal of our method is to provide proper weight for each instance in the source dataset. In order to adapt the model to the target domain, instances that are underrepresented in the source domain (comparing to the target domain) should get relatively higher weights. Similarly, instances that are overrepresented in the source domain should get lower weights.

After learning the weights, they are used for for weighted classification.

3.2 Overall Approach

Inspired by the idea of Generative Adversarial Networks (GAN), we use two neural networks. We call them weighting and discriminator network. (The role of former is similar to the generator in the GANs, and the latter resembles the GAN's discriminator.) The main idea in GANs is to learn to create high quality fake images out of noise to the point that the discriminator cannot distinguish fake and real images and is "fooled".

Similar to the main idea of the GANs, we aim to feed the discriminator with the data from source and target and upweight/downweight the source data by the weighting network in order to "fool" the discriminator. This is equivalent to say that

the weighted source distribution is relatively similar to the target distribution.

Concretely, assume f_{θ_d} is the discriminator network parameterized by θ_d, its inputs are the data (e.g. images of the dataset) and its outputs are labels of a binary classification. Also g_{θ_g} is the weighting network parameterized by θ_g (We have denoted the weighting network as g to not confuse it with the weights, also its role is similar to the generator in GANs). g_{θ_g}'s inputs are the source data, and its output are the corresponding weights which are explained in the following. We also denote the input variables as x.

Assume $B = (x_i)_{i=1}^{b}$ is a batch of unlabelled data from the source. We aim to assign suitable weight to each of the instances. The weighting network does so and $W = g_{\theta_g}(B)$ where:

- $W = (w_i)_{i=1}^{b}$
- $w_i > 0$
- $\sum_{i=1}^{b} w_i = b$

Let $\mathbf{Cat}(\mathbf{B}, \mathbf{W})$ be a categorical distribution over the samples in the batch (B), where the probability for the outcome x_i is $\frac{w_i}{b}$. We then generate b samples independently from the $\mathbf{Cat}(\mathbf{B}, \mathbf{W})$ distribution and denote it as $B' = \left(x'_j\right)_{j=1}^{b}$. We call B' as a reweighted source batch and it is supposed to look like a batch from the target. Finally, the discriminator, f_{θ_d}, is a binary classifier and it is supposed to differentiate target samples from "reweighted" source samples.

However, one challenge is that if we use sampling, it would not be easy to compute the gradient. Therefore, instead of sampling, we multiply each sample's weight to its corresponding loss which is theoretically equivalent with sampling, according to the

equation 3.2.1. A high-level overview of the framework is shown in figure 3.3.

$$\mathbb{E}_{x\sim Cat(B,W)}\nabla_{\theta_g} L_D(x) = \nabla_{\theta_g}\mathbb{E}_{x\sim Cat(B,W)} L_D(x) = \nabla \sum_{i=1}^{n} w_i \ell_D(x_i) \tag{3.2.1}$$

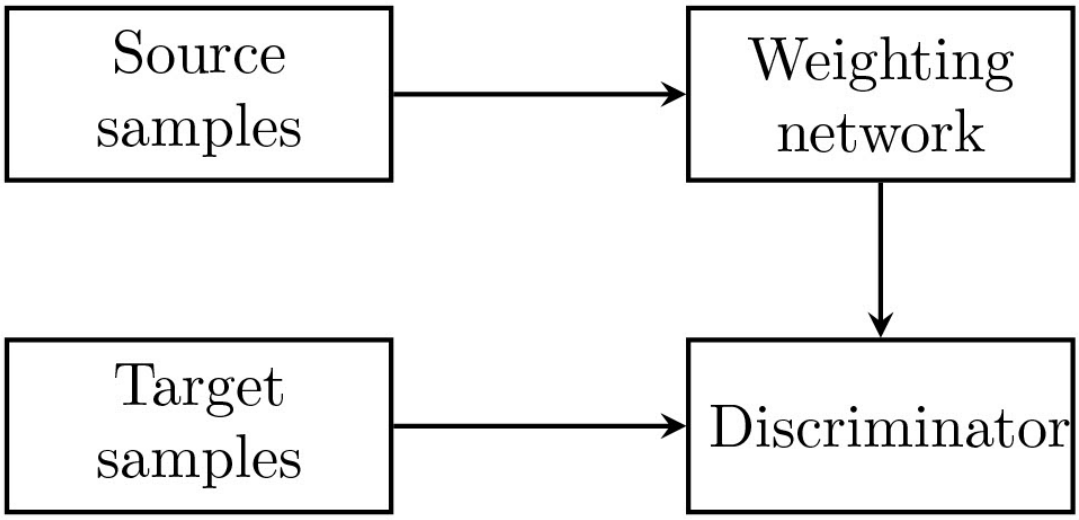


Figure 3.3: An overview of the proposed framework

3.3 Training

In this section we explain the training of the proposed method. Our domain adaptation method is comprised of two parts. First, we compute the suitable weight of each of the instances in the source domain, using the discriminator and weighting networks. Secondly, these weights are used in a weighted ERM learning algorithm (e.g. a convolutional neural network).

We want to train the discriminator and weighting networks using the min-max problem in equation 3.3.1. Note that B_s and B_t represent batches of source and target

respectively. Also D is the binary classification network (discriminator).

$$\max_{\theta_g} \min_{\theta_d} \sum_{B_t} \left(-\frac{1}{|B_t|} \sum_{x \in B_t} \log(D(x)) \right) - \sum_{B_s} \left(\frac{1}{|B_s|} \sum_{x \in B_s} w(x) \log(1 - D(x)) \right) \quad (3.3.1)$$

However, this approach does not work well in practice and a better solution is to define different loss functions for discriminator and weighting networks which are given in the following. We then update the parameters iteratively using stochastic gradient descent.

$$L_D(B_s, B_t) = \sum_{B_t} \left(-\frac{1}{|B_t|} \sum_{x \in B_t} \log(D(x)) \right) - \sum_{B_s} \left(\frac{1}{|B_s|} \sum_{x \in B_s} w(x) \log(1 - D(x)) \right) \quad (3.3.2)$$

$$L_W(B_s) = -\sum_{B_s} \left(\frac{1}{|B_s|} \sum_{x \in B_s} w(x) \log(1 - D(x)) \right) \quad (3.3.3)$$

An overview of the training of these two networks are shown in the algorithm 1.

3.3.1 Training the discriminator

The aim of training the discriminator is to maximize the probability of correctly classifying the domain in which the instances belong to. This is done in two steps. First, a batch of target data is forward passed through the discriminator, then the first term of loss in eq. 3.3.2 is calculated and then the gradients in backward pass are calculated. Secondly a batch of source data is given to the weighting network to obtain the corresponding weights. Then it is passed through the discriminator and the second term in eq. 3.3.2 is calculated. Finally, after accumulating the gradients

with a backward pass, the discriminator's optimizer is called.

3.3.2 Training the weighting network

We aim to train the weighing network in an effort to match the weighted source distribution with the target distribution. Intuitively, the weighting network assigns higher weights to the instances which are more likely to appear in the target domain, and similarly lower weights for those which are unlikely to be in the target domain.

Note that when a batch of source data is passed through the weighting network, a normalization is done on the weights, therefore the mean of weights in each batch is equal to 1.

3.3.3 Using the weights in a class-weighted classification

After learning the weights, we use them for weighted classification. Suppose there are N samples in each batch (equation 3.3.4), the loss of each sample is calculated by equation 3.3.5. C is the number of classes, and w_c is the corresponding weight of each class.

$$\ell(x, y) = \frac{\sum_{n=1}^{N} l_n}{N} \tag{3.3.4}$$

$$l_n = -\sum_{c=1}^{C} w_c log \frac{exp(x_{n,c})}{\sum_{i=1}^{C} exp(x_{n,i})} y_{n,c} \tag{3.3.5}$$

Algorithm 1 Joint training of weighting and discriminator networks

Input:

B_s: batches from source domain

B_t: batches from target domain

T: number of iterations

η_g, η_d: Step sizes

Output:

W: estimated importance weights

for $t = 1, ..., T$ **do**

Pass B_s to the discriminator.

$W \leftarrow$ Pass B_s to the weighting network.

Compute first part of $L_D(B_s, B_t)$ by equation 3.3.2.

Pass B_t to the discriminator.

Compute second part of $L_D(B_s, B_t)$ by equation 3.3.2.

$\theta_d^t \leftarrow \theta_d^{t-1} - \eta_g \nabla L_D(B_s, B_t)$.

Compute $L_W(B_s)$ by equation 3.3.3.

$\theta_g^t \leftarrow \theta_g^{t-1} - \eta_g \nabla L_W(B_s)$.

end for

use W for weighted classification by equations 3.3.4 and 3.3.5

Chapter 4

Experiments

In the experiments, we consider several scenarios, explain them and see the results.

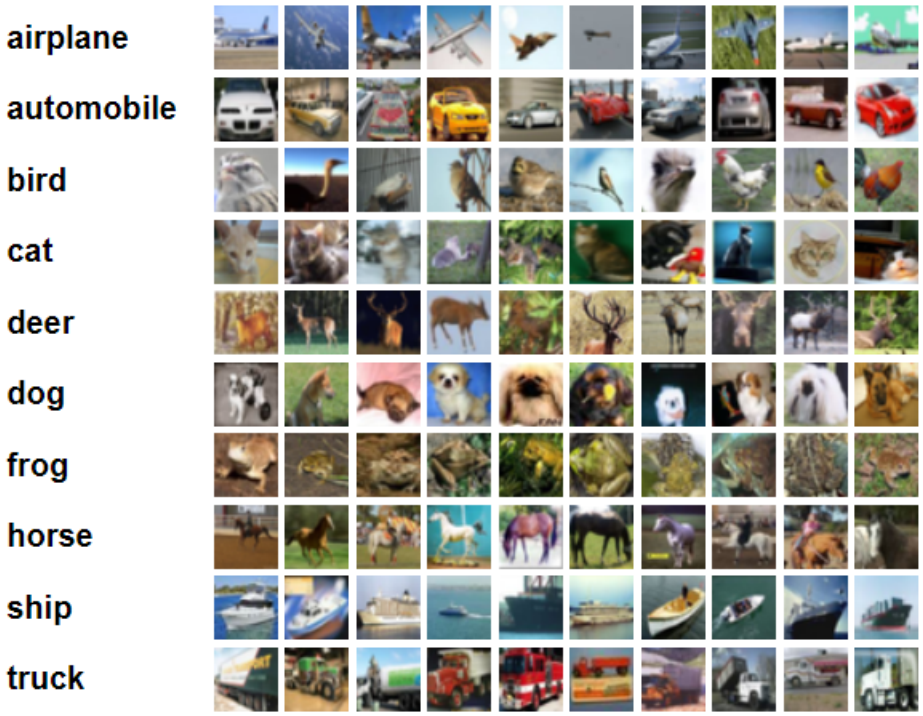
4.1 The setup

As discussed in the previous chapter, our proposed method can be able to handle label shift. In order to simulate the label shift, we have created a custom dataset based on the CIFAR-10 dataset [37]. The shift which we encounter in label shift is the case where the proportions of the number of instances for given classes is different in the source and target domains (e.g. same number of cats and dogs in the source, but more pictures of cats in the target). We can enter the number of instances in each class as an input to the custom dataset and thus the resulted dataset can be used for checking the label shift. We aim to compare the use of ideal weights and the usual case (no weighting) in different situations, form the case where there is minor shift between the domains to the case where the shift is extreme. To do so, we use the variable *alpha*, which has a value from 0 to 1. In this experiment, we choose a

group of classes in the source. For each class, we only keep *alpha* times the size of class number of instances and remove the rest (using the custom dataset). Also, we choose the rest of classes in the source and do the same with the given alpha. For example, if there are 5000 pictures in each class of the training set (or the source domain) and 1000 pictures in each class of the test set (or the target domain), We may have $5000 \times alpha$ pictures of cats and 5000 pictures of dogs in the training set. While, there are 1000 pictures of cats and $1000 \times alpha$ pictures of dogs in the test set..

Therefore, we can compare the accuracies of classification using the ideal weights and the usual case at different values of *alpha*. An important result is that the gaps between the accuracy of weighted classification using ideal weights and usual classification highly depends on how we group the classes into source and target and then apply the *alpha*. We have studied two cases: (1) the classes which *alpha* is applied on them in the source domain, are the first five classes, and the rest of classes are customized in the target. (2) The first group (for source) are the animals (consisting of the labels of bird, cat, deer, dog, frog and horse), and the target group is the machines (consisting of airplane, automobile, ship and truck).

After learning the weights, we have used them for weighted classification and the classifier is ResNet-18[25].



4.2 Experimental results

4.2.1 Learning the Weights

In this section, we study how the learned weights converge to the ideal weights during training. To do so, we compute the Euclidean norm of the difference of the learned weight and ideal weight and plot it. There are two set of experiments which will be explained in the following.

Animals vs. Machines

In this experiment, the source domain consists of a majority of animals pictures, while the target domain has a majority of machine pictures. To be more precise, $alpha = 0.1$. As we can see, the learned weights have converged to the ideal weight

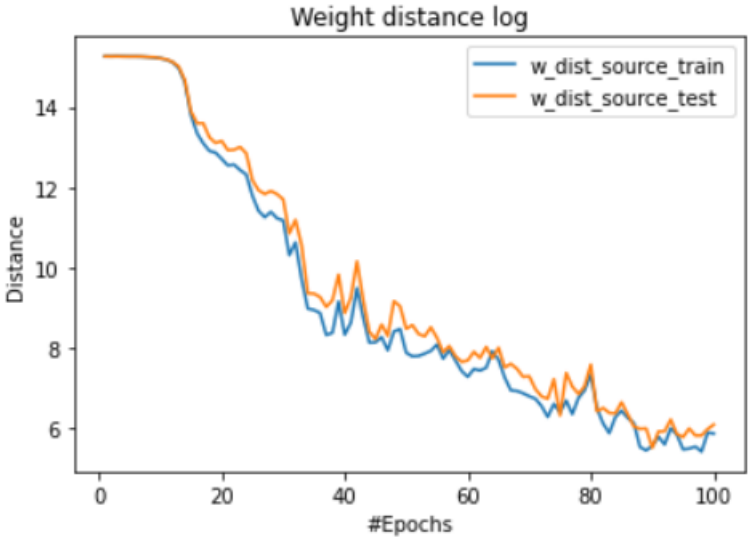


Figure 4.4: Results of learning the weights for the Animals vs. Machines
In this experiment, the learned weights by the proposed algorithm converged well to the ideal weights and there is no overfitting.

fairly well. We have also plotted the weights of both train and test sets and it shows that there is no overfitting.

First five vs. Last five

In this part, we have repeated the experiment with different domains. Here, the source domain has a majority of pictures of the first five classes (airplane, automobile, bird, cat, deer), and the target domain has a majority of pictures of the last five classes (dog, frog, horse, ship, truck). We have kept all the parameters of the neural network (including number of epoches, learning rate, etc.) fixed. As we can see the model has poorer performance.

These experiments show that our model is dependent on the data. In order to have a better understanding of this problem, we have done another set of experiments which will be explained in the next section.

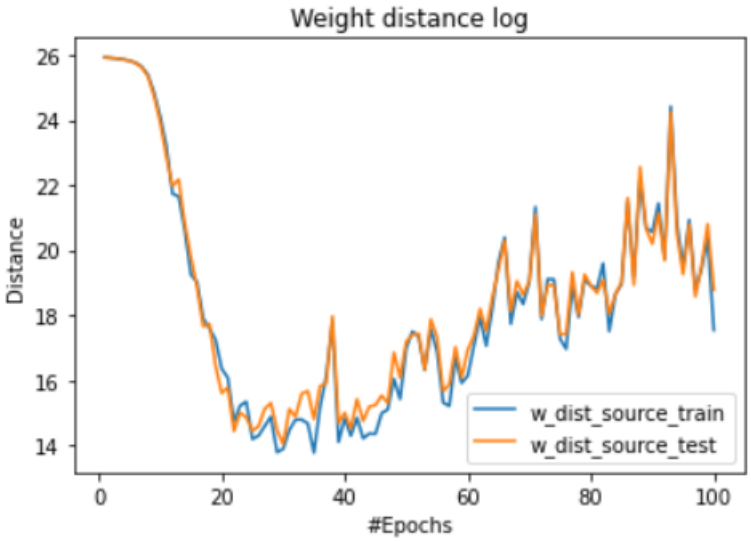


Figure 4.5: Results of learning the weights for the First five classes vs. Last five classes
In this experiment, the proposed algorithm performs poorer with a different set of domains, although all the parameters of the network are fixed.

After learning the weights, they are used for for weighted classification.

4.2.2 Binary Classification

In this section, we have modified the labels of the dataset. We have assigned the label of 1 to the source data points, and 0 to the target points. By source and target we mean the two formations of the previous section. We also try two values for *alpha*, which are 0.1 and 1. Hence, we can understand how hard is it to classify the points when we have a shift. The accuracy of the classifications are summarized in the following table.

The results show that the classification of first five vs. last five classes are much harder when there is an extreme shift. Note that an extreme shift means higher difference between the source and target domains. The results also indicate that Animal

Table 4.2: Classification of domains for two values of *alpha*

alpha	Animal vs. Machine	First five vs. Last five
0.1	92.91	45.63
1	97.18	89.31

vs. Machine case is easier to classify (or equivalently more separable). Therefore, our hypothesis is that in order to learn the weights properly, the model should be able to separate the data points at the first place.

4.2.3 Ideal Weights and Classification

In order to study the effect of our learned weights on the label shift, we should first compare the usual case with the case where "ideal weights" are used in a weighted classification scenario. By ideal weights (i.e. w_i s) we mean $w(y) = q(y)/p(y)$ for every class $y \in \mathcal{Y}$. Here p and q are denoted as the number of instances in the source and target respectively. Similar to previous experiments, we do this experiment in two different settings:

First five vs. Last five

In the figure 4.6, *alpha* shows the amount of shift between the source and target. Low value of *alpha* corresponds to more extreme shift between the domains. Full explanation of this variable is available in Section 4.1. The accuracies shown as "normal" indicate how the neural network performs without any weighting, and accuracies shown as "ideal weights" are for the case when ideal weights are used as a weighting in the loss function. As we expected, using ideal weights improves the performance of the neural network when there is an extreme shift (lower values of *alpha*) and this

improvement diminishes as we increase *alpha*. Because when the source and target are almost identical, the values of ideal weights converges to 1, and weighting doesn't have much effect on the performance of the network.

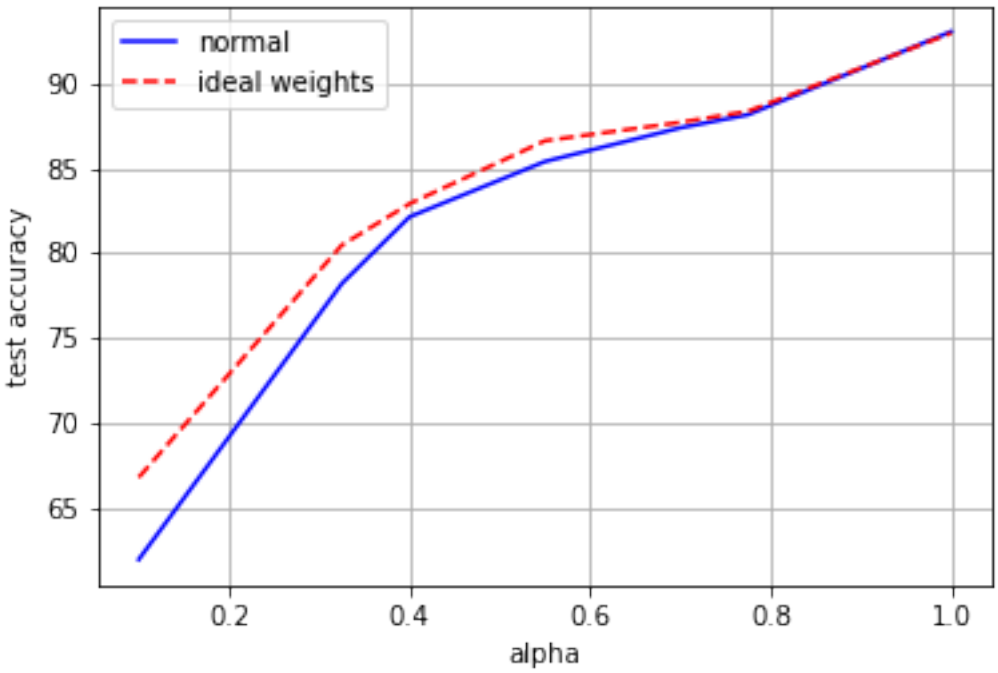


Figure 4.6: Results of using ideal weights for the First five classes vs. Last five classes

Animal vs. Machine

In this part, we have repeated the experiment on a different setting (Animal vs. Machine) and the results are shown in figure 4.7. As we see, using ideal weights results in a different situation here. When the shift is extreme, using ideal weights not only doesn't improve the accuracy, but also it decreases it. Also, similar to previous experiment, using ideal weights doesn't have much effect in higher values of

alpha.

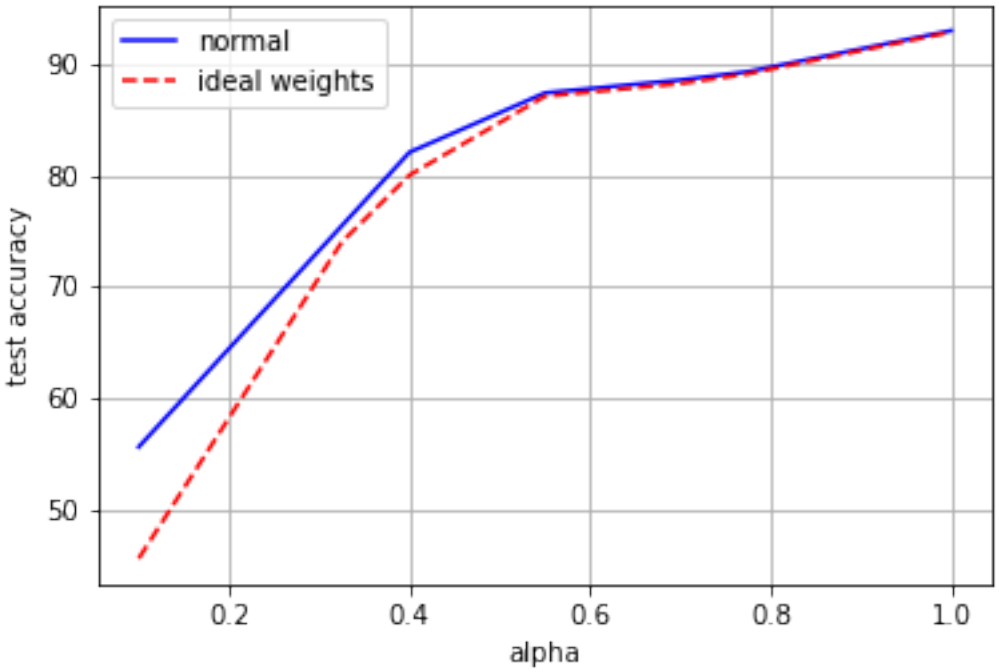


Figure 4.7: Results of using ideal weights for Animals vs. Machines

In order to justify this phenomenon, our hypothesis is that the effect of importance weighting depends on how separable are the domains. In the section 4.2.2, we showed that in a binary classification scenario, the "Animal vs. Machine" problem is considerably easier to classify comparing to "First five classes vs. Last five classes". Also, Lipton *et al.* [11] have shown that when the classes of data are separable, the effect of importance weighting diminishes over epoches in deep neural networks. In figure 4.8, there is an experiment with a synthetic two-dimensional linearly-separable dataset. Points are colored according to their true labels. The background shades show the decision surface of an multi-layer perceptron with a hidden layer of size

64. The red line shows a decision boundary by a logistic regression classifier and the dotted black line shows the max-margin separator. There are different weightings in each row and as we can see, after sufficient epoches, the final decision boundary is almost similar regardless of the weighting.

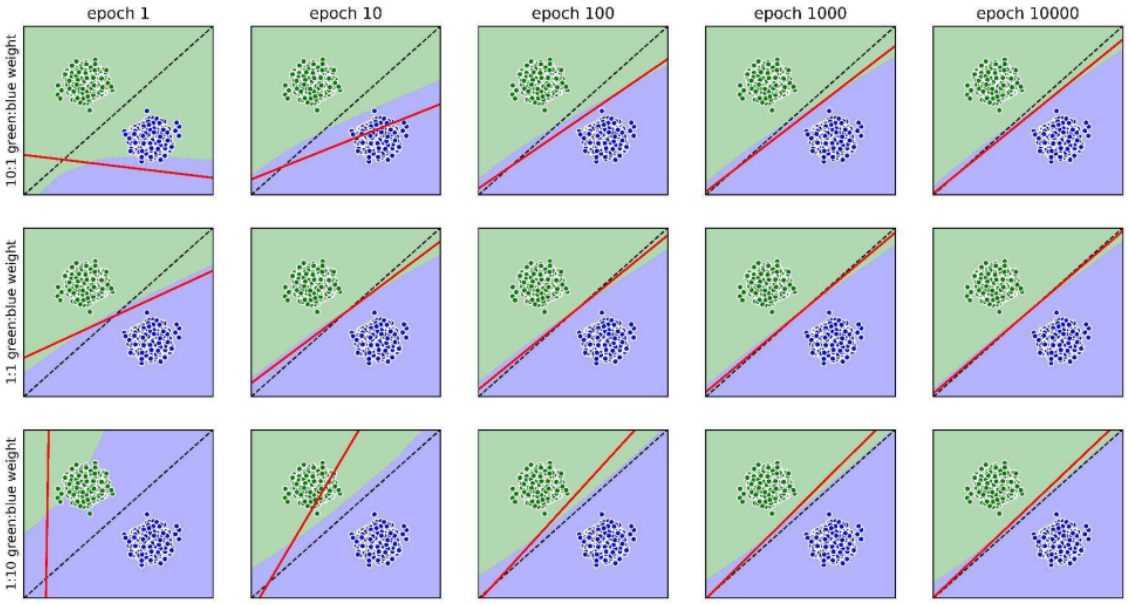


Figure 4.8: Convergence of decision boundaries over epochs of training with different importance weights [11]

Therefore, we believe when the classes are almost separable, using importance weighting may not help the classifier.

4.2.4 Learned Weights and Classification

In this part, we study the effect of the learned weights on the accuracy of classification. In this experiment, we have set the value of *alpha* equal to 0.1, which means that the shift between the source and target domains are at the highest level. Having an extreme shift between the domains helps us better to see the effect of using weight in

classification. In the following table, the first row shows the accuracy when the usual classifier is used. The second row is for the case when we use the learned weights of our method weighted classification. And the last row shows the use of ideal weights in weighted classification.

Table 4.3: Comparison of different methods of weighting

Accuracy	Animal vs. Machine	First five vs. Last five
without weights	57.96	62.01
learned weights	56.85	67.25
ideal weights	45.62	66.83

As we see, using the learned weights has improved the classification accuracy in the First five vs. Last five case. However, learned weights are not effective in the other case (Animal vs. Machine). This result of this experiment is similar to the last section's result and it seems importance weighting does not necessarily improve the classifier.

A similar phenomenon is observed by Lipton *et al.*[11] and they believe weighting the loss function of deep networks fails to correct distribution shift. They suggest rethinking the standard application of importance weighting in combination with deep learning.

4.3 Summary

The experiments done in this part show that the ability of our proposed method on learning the weight relies on the separability of the domains. When the domains are relatively separable, learning the weights is done properly. However, when the domains are not separable, learning the weights becomes challenging.

Our experiments show that the effect of learned weights also depends on the separability of the domains. When the difference between the domains is less, learned weights can improve the classification accuracy but they fail to improve the accuracy when the domains are relatively separable.

Chapter 5

Conclusion

This work presents a novel idea for importance weighting which is a common approach for domain adaptation. Previous methods of importance weighting are mostly kernel-based and their downside is that they scale poorly and underperform on high-dimensional data.

The main contribution of this thesis is proposing a new framework for learning weights used in importance weighting. After estimating these weights, they are used for weighted classification. Unlike existing approaches for this problem, our approach is based on neural networks. Two neural networks are used in this framework and they are jointly trained using an adversarial learning scheme. We also designed a benchmark which can simulate distribution shift and the intensity of the shift can be controlled. We explored different choices of parameters and their effects on performance. We empirically evaluated our framework and showed the weighted learned by our method can converge to the desired weights. However, we also noticed the challenges in learning the weights. Our experiments show when the domains are relatively separable, learning the weights is an easier task. But in this case using importance

weighting may not help the classifier. On the other hand, when the domains are not separable at all, learning the weights becomes challenging and the learned weights improve the classification accuracy.

5.1 Future Work

In this part, we have introduced possible directions for future work.

5.1.1 Extension of this work

As discussed earlier, the ability of our proposed method of learning the weights relies on the separability of the domains. Therefore, an important question is how we can modify this method to work efficiently even when the domains are not much separable.

One other important issue is to examine the effect of weighting in deep neural networks. Our experiments show that when the domains are separable, weighting fails to improve the classifier. This problem needs to be addresses in future works.

5.1.2 Manifestation shift

We worked on a certain type of shift in this thesis which is population shift. There are other types of shift in domain adaptation problems such as manifestation shift. There are some works which assume these two types of shift are present in the data, but they have used both domain-adversarial neural networks and kernel methods in their approach. Hence, a potential future work is to examine whether it is possible to solve this problem using solely neural networks or not.

5.1.3 Subpopulation shift

The focus of this thesis is on population shift which is a from of covariate shift. The problem of population shift can be extended to subpoulation shift. In subpopulation shift, the difference of source and target domains is how each subpopulation is represented. Therefore, it will be interesting to investigate if importance weighting using neural networks can help this problem.

Bibliography

[1] Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. (2010). Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, **33**(5), 898–916.

[2] Argyriou, A., Evgeniou, T., and Pontil, M. (2006). Multi-task feature learning. *Advances in neural information processing systems*, **19**.

[3] Argyriou, A., Pontil, M., Ying, Y., and Micchelli, C. (2007). A spectral regularization framework for multi-task structure learning. *Advances in neural information processing systems*, **20**.

[4] Arora, S., Ge, R., Liang, Y., Ma, T., and Zhang, Y. (2017). Generalization and equilibrium in generative adversarial nets (gans). In *International Conference on Machine Learning*, pages 224–232. PMLR.

[5] Azizzadenesheli, K., Liu, A., Yang, F., and Anandkumar, A. (2019). Regularized learning for domain adaptation under label shifts. *arXiv preprint arXiv:1903.09734*.

[6] Batu, T., Fortnow, L., Rubinfeld, R., Smith, W. D., and White, P. (2000). Testing that distributions are close. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 259–269. IEEE.

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

[8] Bickel, S., Brückner, M., and Scheffer, T. (2007). Discriminative learning for differing training and test distributions. In *Proceedings of the 24th international conference on Machine learning*, pages 81–88.

[9] Biesialska, M., Biesialska, K., and Costa-jussà, M. R. (2020). Continual lifelong learning in natural language processing: A survey. *arXiv preprint arXiv:2012.09823*.

[10] Borji, A. (2019). Pros and cons of gan evaluation measures. *Computer Vision and Image Understanding*, **179**, 41–65.

[11] Byrd, J. and Lipton, Z. (2019). What is the effect of importance weighting in deep learning? In *International Conference on Machine Learning*, pages 872–881. PMLR.

[12] Caruana, R. (1997). Multitask learning. *Machine learning*, **28**(1), 41–75.

[13] Chen, B.-C. and Kae, A. (2019). Toward realistic image compositing with adversarial learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8415–8424.

[14] Choi, Y., Choi, M., Kim, M., Ha, J.-W., Kim, S., and Choo, J. (2018). Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8789–8797.

[15] Daumé III, H. (2009). Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815*.

[16] Du Plessis, M. C. and Sugiyama, M. (2014). Semi-supervised learning of class balance under class-prior change by distribution matching. *Neural Networks*, **50**, 110–119.

[17] Fang, T., Lu, N., Niu, G., and Sugiyama, M. (2020). Rethinking importance weighting for deep learning under distribution shift. *Advances in Neural Information Processing Systems*, **33**, 11996–12007.

[18] Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. (2016). Domain-adversarial training of neural networks. *The journal of machine learning research*, **17**(1), 2096–2030.

[19] Germain, P., Habrard, A., Laviolette, F., and Morvant, E. (2013). A pac-bayesian approach for domain adaptation with specialization to linear classifiers. In *International conference on machine learning*, pages 738–746. PMLR.

[20] Goodfellow, I. (2016). Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*.

[21] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, **27**.

[22] Gretton, A., Smola, A., Huang, J., Schmittfull, M., Borgwardt, K., and Schölkopf, B. (2009). Covariate shift by kernel mean matching. *Dataset shift in machine learning*, **3**(4), 5.

[23] Guo, J., Gong, M., Liu, T., Zhang, K., and Tao, D. (2020). Ltf: A label transformation framework for correcting label shift. In *International Conference on Machine Learning*, pages 3843–3853. PMLR.

[24] Hamada, K., Tachibana, K., Li, T., Honda, H., and Uchida, Y. (2018). Full-body high-resolution anime generation with progressive structure-conditional generative adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0.

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

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

[27] Hospedales, T., Antoniou, A., Micaelli, P., and Storkey, A. (2020). Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*.

[28] Hsu, C.-C., Hwang, H.-T., Wu, Y.-C., Tsao, Y., and Wang, H.-M. (2017). Voice conversion from unaligned corpora using variational autoencoding wasserstein generative adversarial networks. *arXiv preprint arXiv:1704.00849*.

[29] Huang, J., Gretton, A., Borgwardt, K., Schölkopf, B., and Smola, A. (2006). Correcting sample selection bias by unlabeled data. *Advances in neural information processing systems*, **19**.

[30] Huang, X., Liu, M.-Y., Belongie, S., and Kautz, J. (2018). Multimodal unsupervised image-to-image translation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 172–189.

[31] Huh, M., Agrawal, P., and Efros, A. A. (2016). What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*.

[32] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR.

[33] Jebara, T. (2004). Multi-task feature and kernel selection for svms. In *Proceedings of the twenty-first international conference on Machine learning*, page 55.

[34] JI, Z., WANG, H., YU, Y., and PANG, Y. (2019). A decadal survey of zero-shot image classification. *SCIENTIA SINICA Informationis*, **49**(10), 1299–1320.

[35] Kifer, D., Ben-David, S., and Gehrke, J. (2004). Detecting change in data streams. In *VLDB*, volume 4, pages 180–191. Toronto, Canada.

[36] Kodali, N., Abernethy, J., Hays, J., and Kira, Z. (2017). How to train your dragan. *arXiv preprint arXiv:1705.07215*, **2**(4).

[37] Krizhevsky, A., Hinton, G., *et al.* (2009). Learning multiple layers of features from tiny images.

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

[39] Lee, H.-Y., Tseng, H.-Y., Huang, J.-B., Singh, M., and Yang, M.-H. (2018). Diverse image-to-image translation via disentangled representations. In *Proceedings of the European conference on computer vision (ECCV)*, pages 35–51.

[40] Lee, H.-Y., Tseng, H.-Y., Mao, Q., Huang, J.-B., Lu, Y.-D., Singh, M., and Yang, M.-H. (2020). Drit++: Diverse image-to-image translation via disentangled representations. *International Journal of Computer Vision*, **128**(10), 2402–2417.

[41] Lee, S.-I., Chatalbashev, V., Vickrey, D., and Koller, D. (2007). Learning a meta-level prior for feature relevance from multiple related tasks. In *Proceedings of the 24th international conference on Machine learning*, pages 489–496.

[42] Li, D. and Hospedales, T. (2020). Online meta-learning for multi-source and semi-supervised domain adaptation. In *European Conference on Computer Vision*, pages 382–403. Springer.

[43] Li, Y., Murias, M., Major, S., Dawson, G., and Carlson, D. (2019). On target shift in adversarial domain adaptation. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 616–625. PMLR.

[44] Lin, K., Li, D., He, X., Zhang, Z., and Sun, M.-T. (2017). Adversarial ranking for language generation. *Advances in neural information processing systems*, **30**.

[45] Lipton, Z., Wang, Y.-X., and Smola, A. (2018). Detecting and correcting for label shift with black box predictors. In *International conference on machine learning*, pages 3122–3130. PMLR.

[46] Liu, M.-Y., Breuel, T., and Kautz, J. (2017). Unsupervised image-to-image translation networks. *Advances in neural information processing systems*, **30**.

[47] Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. (2017). Variational continual learning. *arXiv preprint arXiv:1710.10628*.

[48] Quionero-Candela, J., Sugiyama, M., Schwaighofer, A., and Lawrence, N. D. (2009). Dataset shift in machine learning— the mit press.

[49] Raina, R., Battle, A., Lee, H., Packer, B., and Ng, A. Y. (2007). Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning*, pages 759–766.

[50] Ren, M., Zeng, W., Yang, B., and Urtasun, R. (2018). Learning to reweight examples for robust deep learning. In *International conference on machine learning*, pages 4334–4343. PMLR.

[51] Shimodaira, H. (2000). Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, **90**(2), 227–244.

[52] Stewart, M. (2019). Understanding dataset shift. `https://towardsdatascience.com/understanding-dataset-shift-f2a5a262a766`.

[53] Sugiyama, M., Nakajima, S., Kashima, H., Buenau, P., and Kawanabe, M. (2007). Direct importance estimation with model selection and its application to covariate shift adaptation. *Advances in neural information processing systems*, **20**.

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

[55] Vaibhav, V., Singh, S., Stewart, C., and Neubig, G. (2019). Improving robustness of machine translation with synthetic noise. *arXiv preprint arXiv:1902.09508*.

[56] Vanschoren, J. (2018). Meta-learning: A survey. *arXiv preprint arXiv:1810.03548*.

[57] Vilalta, R. and Drissi, Y. (2002). A perspective view and survey of meta-learning. *Artificial intelligence review*, **18**(2), 77–95.

[58] Wang, C., Xu, C., Wang, C., and Tao, D. (2018). Perceptual adversarial networks for image-to-image transformation. *IEEE Transactions on Image Processing*, **27**(8), 4066–4079.

[59] Wang, J., Lan, C., Liu, C., Ouyang, Y., Zeng, W., and Qin, T. (2021). Generalizing to unseen domains: A survey on domain generalization. *arXiv preprint arXiv:2103.03097*.

[60] Wang, W., Zheng, V. W., Yu, H., and Miao, C. (2019). A survey of zero-shot learning: Settings, methods, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, **10**(2), 1–37.

[61] Wu, H., Zheng, S., Zhang, J., and Huang, K. (2019). Gp-gan: Towards realistic high-resolution image blending. In *Proceedings of the 27th ACM international conference on multimedia*, pages 2487–2495.

[62] Yao, Y. and Doretto, G. (2010). Boosting for transfer learning with multiple sources. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 1855–1862. IEEE.

[63] Yi, Z., Zhang, H., Tan, P., and Gong, M. (2017). Dualgan: Unsupervised dual learning for image-to-image translation. In *Proceedings of the IEEE international conference on computer vision*, pages 2849–2857.

[64] Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? *Advances in neural information processing systems*, **27**.

[65] Yu, L., Zhang, W., Wang, J., and Yu, Y. (2017). Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31.

[66] Zhou, K., Yang, Y., Qiao, Y., and Xiang, T. (2021). Domain adaptive ensemble learning. *IEEE Transactions on Image Processing*, **30**, 8008–8018.

[67] Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232.