ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
“ΘΕΩΡΗΤΙΚΗ ΠΛΗΡΟΦΟΡΙΚΗ”

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Γενικεύοντας την Προσαρμογή Πεδίου: Χαλάρωση των Υποθέσεων & μία Εναλλακτική Συνάρτηση Κόστους για Μεθόδους με Αντιμαχόμενα Δίκτυα

Γιώργος Α. Πικραμένος

Επιβλέπων: Σ. Περαντώνης

ΑΘΗΝΑ

SEPTEMBER 2019

NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

PROGRAM OF POSTGRADUATE STUDIES
“THEORETICAL COMPUTER SCIENCE”

MASTERS THESIS

Generalizing Domain Adaptation: Relaxing Task Assumptions & an Alternative Cost Function for Adversarial Methods

George A. Pikramenos

Supervisor: S. Perantonis
ATHENS

ΣΕΠΤΕΜΒΡΙΟΣ 2019

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Γενικεύοντας την Προσαρμογή Πεδίου: Χαλάρωση των Υποθέσεων & μία Εναλλακτική Συνάρτηση Κόστους για Μεθόδους με Αντιμαχόμενα Δίκτυα

Γιώργος Α. Πικραμένος

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: Σ. Περαντώνης, Διευθυντής Έρευνας Ινστιτούτο Πληροφορικής και Τηλεπικοινωνιών, ΕΚΕΦΕ Δημόκριτος

Ημερομηνία Εξέτασης: 30 Σεπτεμβίου 2019

MASTERS THESIS

Generalizing Domain Adaptation: Relaxing Task Assumptions & an Alternative Cost Function for Adversarial Methods

George A. Pikramenos

SUPERVISOR: S. Perantonis, Research Director, Institute of Informatics and Telecommunications, NSCR Demokritos

Examination Date: September 30, 2019

ΠΕΡΙΛΗΨΗ

Οι τεχνικές επιβλεπόμενης μάθησης τυπικά λειτουργούν με την υπόθεση πως το σύνολο εκπαίδευσης και δοκιμής έχουν ληφθεί από μία κοινή κατανομή. Για αυτόν τον λόγο, η εκπαίδευση χρήσιμων μοντέλων με συμβατικές μεθόδους για επιβλεπόμενη μάθηση απαιτούν τουλάχιστον κάποια δεδομένα με ετικέτες από το πρόβλημα που μας ενδιαφέρει. Η εκτεταμένη έρευνα στο πεδίο της μάθησης μηχανής, και ειδικότερα στην βαθιά μάθηση, έχουν αποφέρει πολύ ισχυρές μεθόδους για την αντιμετώπιση προβλημάτων επιβλεπόμενης μάθησης, ενώ οι εξελίξεις σε συστήματα διαχείρισης μεγάλων δεδομένων μας επιτρέπουν να συλλέγουμε και να οργανώνουμε ακατέργαστα δεδομένα με ταχύτατους ρυθμούς. Αυτή η πρόοδος έχει σε μεγάλο βαθμό δει το “κώλυμα” της διαδικασίας της μάθησης να μεταφέρεται από την μοντελοποίηση και την εκπαίδευση, στην συλλογή ετικετών. Υποστηρίζουμε πως για την πλήρη εκμετάλλευση της τεχνολογίας μας, πρέπει να ξεπεράσουμε αυτό το “κώλυμα” και να αναπτύξουμε μοντέλα που είναι ανθεκτικά σε διαφορές μεταξύ των κατανομών των συνόλων εκπαίδευσης και δοκιμής. Η προσαρμογή πεδίου είναι ένα πλαίσιο το οποίο απευθύνεται στα παραπάνω θέματα και κάτω από ορισμένες υποθέσεις προσφέρει εργαλεία για την επίλυσή τους. Σε αυτήν την εργασία, θα συζητήσουμε βελτιώσεις σε υπάρχουσες τεχνικές για την προσαρμογή πεδίου που στηρίζονται σε αντιμαχόμενα νευρωνικά δίκτυα. Εισάγουμε μία νέα συνάρτηση κόστους εμπνευσμένη από την πρόοδο στα generative adversarial networks (GANs) και το πεδίο της βέλτιστης μεταφοράς. Τέλος, προτείνουμε ένα νέο πλαίσιο, το οποίο καλούμε Αμφίδρομη Μερική Προσαρμογή Πεδίου, στο οποίο χαλαρώνουμε τις συνήθεις υποθέσεις που γίνονται στην απλή προσαρμογή πεδίου και παρουσιάζουμε έναν αλγόριθμο για την αντιμετώπιση προβλημάτων σε αυτό το νεό πλαίσιο.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Μάθηση Μηχανής

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Μεταφερόμενη Γνώση, Προσαρμογή Πεδίου, Αντιμαχόμενα Νευρωνικά Δίκτυα

ABSTRACT

Supervised learning techniques typically work under the assumption that train and test datasets are drawn from the same distribution. As such, training useful models with conventional techniques for a supervised learning task, requires us to obtain at least some labeled data for our problem of interest. Extensive research efforts in the fields of *ML* and in particular *DL* have yielded powerful methods for tackling supervised problems and developments in big data systems have made it possible to gather raw data at unprecedented rates. These advances have seen, to a large extent, the bottleneck of the learning procedure shift from modelling/training to obtaining labels for training data. We argue that in order to fully utilize our technologies, we need to bypass this bottleneck by creating models that are robust under train and test data distribution discrepancies. *DA* is a framework that addresses the aforementioned issues and under certain assumptions, provides tools to resolve them. In this thesis we discuss improvements on current techniques for *DA* that rely on adversarial neural networks. We introduce a new cost function for such methods inspired by progress in generative adversarial networks (GANs) and the field of optimal transport. Finally, we propose a novel problem setup, termed two-way partial domain adaptation, which relaxes the assumptions made in traditional *DA* and we present a first algorithm to tackle problems in this setup.

SUBJECT AREA: Machine Learning

KEYWORDS: Transfer Learning, Domain Adaptation, Adversarial Neural Networks

To my father and mother

ACKNOWLEDGEMENTS

Firstly, I would like to thank my supervisor Stavros Perantonis for all his guidance and support without which this thesis could not have been completed. I also want to thank the people at the computational intelligence lab of NCSR Demokritos for providing their feedback at various stages of this project and helping me complete my experiments by granting me access to their hardware.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

Code for the experiments and methods described in this project can be found in the following repository `https://github.com/GPikra/Adapy`.

An important limitation of "traditional" *ML* algorithms is that they work under the assumption that training and test data share the same distribution (and/or feature space). *TL* is essentially a framework where this assumption is relaxed, allowing us to tackle *ML* problems by incorporating knowledge from another distinct but related *ML* problem[1]. Effective transfer learning is practically relevant and highly desirable [1]. In particular, an important direct application of *TL*, which will constitute our problem of interest in this project, is obtaining labels for an unlabeled dataset from a model trained on some other dataset with a related label space and without necessarily the same feature-space distribution. We will refer to this problem as *label transfer*.

Past studies on *TL* have yielded numerous "shallow" methods[2] [2] for performing transfer, but recently there has been an increasing interest for exploiting deep neural networks in this field. Earlier approaches for "deep" *TL* attempted to capture transferable features from the lower layers of neural networks trained on source data and fine-tune a network with these features on target data. For this reason, these methods are refered to as fine-tuning methods. Later on, training networks with deep architectures specific to targeting *TL* emerged as a fruitful practice and inspired many state of the art algorithms. In this thesis, we focus on an important class of such algorithms, which work within an adversarial framework.

In label transfer, the instances of the underlying datasets may or may not lie in the same feature space and (if they do) we assume that they are distributed differently. This last assumption is not necessary but it is generally the case and if it doesn't hold, label transfer reduces to classification. A special case of label transfer is the following: Given two datasets with the *same label space* such that exactly one of them is labelled -the source dataset-, can we retrieve the labels for the other -the target dataset- ?. As we will see later, this setup is widely known as domain adaptation and there is a vast arsenal of algorithms for tackling this kind of problem. In particular, adversarial domain adaptation algorithms are quite flexible and effective in this setup. Nevertheless, both theory and practice suggest that in certain cases these algorithms can fail and we will explore how to fix this by introducing ideas from the field of optimal transport.

DA is very appealing theoretically, but the shared label space assumption is quite restrictive for practical applications. This is because in practice, we ideally want to extract labels

[1] This is discussed further below

[2] Without the use of deep neural networks

from large “general-purpose” datasets with much larger label spaces than any specific-task-dataset. For example, given an unlabelled dataset of natural images of domestic animals, we want to be able to use eg ImageNet to extract labels. In pursuit of this goal, the partial domain adaptation framework was recently introduced in which algorithms work under the assumption that labels for target data are contained in the larger label space of source data. Even though we believe the introduction of *PDA* is a step in the right direction, we remark that creating such general-purpose source datasets, which document the entire label space of any given specific task is unrealistic. Take for example the task of classifying natural images or the task of classifying human activity in video data. It is kind of hard to imagine a dataset in which every possible label for these tasks is well represented! To address this issue we introduce a new more general framework which we term *two-way partial domain adaptation*.

Given a source and a target dataset, our aim is to automatically extract precisely the information from the source data relevant to the target data and simultaneously to identify which target instances are relevant to the given source. That is, we simply assume some arbitrary non-empty intersection between target and source label spaces, and we want to identify those instances of the target data that can benefit from the source dataset. We refer to such instances as *transfer relevant*. Additionally, we are interested in identifying precisely those instances of the source dataset that can benefit the target dataset (hence “two-way”). Note that we can not hope to predict the label of a target instance belonging to a class not represented in the source label space but the advantage of identifying which instances belong to such classes is two-fold: Firstly, it allows us to minimize negative transfer during the adaptation procedure and secondly, it allows us to identify the subset of target instances that can be reliably labelled using this source.

In this first chapter, we present an overview of the general framework we will be using and describe the transfer learning and domain adaptation problems. In addition, *PAC* theoretic bounds will be discussed for domain adaptation. We also briefly introduce the reader to earlier methods for *DA* as they provide usefull insights and a more complete picture of the field. Specifically, we discuss an important algorithm for shallow feature-representation transfer and devote a section for fine-tuning methods. Last but not least, we describe the adversarial *ANN* framework and how it can be used for *DA*.

In the second chapter, we discuss the Wasserstein distance, a metric for measuring distribution similarity first introduced in the field of optimal transport. We motivate its use in the feature-representation approach for domain adaptation and we argue, through a series of theoretical results, that it is quite natural to build an adversarial model around the Wasserstein distance. Furthermore, we present a novel adversarial algorithm for *DA* using the ideas from this section and present experimental evidence of its superiority on a series of interesting *DA* tasks. These include image obstruction, displacement, rotation and rescaling.

In the third chapter, we present the *PDA* problem. We consider the complications that

arise in *PDA* and present an overview of the main algorithmic schemes that can be found in the bibliography for tackling this problem.

In the final chapter we motivate and discuss our new adaptation framework. We present a novel algorithm to tackle it and provide some theoretical support of its effectiveness. We conduct experiments to illustrate that previously proposed algorithms fail in our new setup, while our new method works.

1.1 Preliminaries

In the *TL* framework, the description of an *ML* problem is typically broken down into a domain and a task[3]. A *domain* $\mathcal{D}$ is a tuple $(\mathcal{X}, X)$, where $\mathcal{X}$ is some feature space and X is some random variable mapping instances of the underlying "experiment" to elements of $\mathcal{X}$. A *task* $\mathcal{T}$ over a domain $\mathcal{D} = (\mathcal{X}, X)$ is a tuple $(\mathcal{Y}, Y|X)$, where $\mathcal{Y}$ is some label space and $Y|X$ is a conditional random variable mapping the results of the underlying experiment to elements of $\mathcal{Y}$. Because we are usually interested in $P(X)$ and $P(Y|X)$ (rather than X and $Y|X$) we often write $(\mathcal{X}, P(X))$ for domains and $(\mathcal{Y}, P(Y|X))$ for tasks.

The objective of the problem $\{\mathcal{D}, \mathcal{T}\}$ is to learn from the available data a function $f : \mathcal{X} \to \mathcal{Y}$, to predict the value of Y given an observation of X. We call f a *predictor* for $\mathcal{T}$. This needs to be done in an optimal way with respect to some criterion we adopt. For instance, in a classification task where we are interested in maximum accuracy, our objective would be to find an f such that $\forall x \in \mathcal{X}$, $f(x) \approx \arg\max_y Pr(Y = y|X = x)$. Similarly, in a regression task where we intend to minimize the *MSE* of f, we would need to have $\forall x \in \mathcal{X}$, $f(x) \approx E[Y|X = x]$.

For example, if we were interested in classifying images in those that contain cats and those that do not, our domain could be $\mathcal{D} = (\mathbb{R}, X)$, where X takes as input an image A and outputs A's average pixel intensity (here A is an instance of the underlying "experiment"; a natural image). Define $Y(A) = 1$ if A is an image of a cat and $Y(A) = 0$ otherwise. Our task may then take the form $\mathcal{T} = (\{0, 1\}, Y|X)$ and our objective is to learn the most probable label $y \in \{0, 1\}$ given an observed average pixel intensity $x \in \mathbb{R}$.

Describing *ML* problems as above, we may phrase the goal of *TL* as follows. Given a source problem $\{\mathcal{D}_S, \mathcal{T}_S\}$ and a target problem $\{\mathcal{D}_T, \mathcal{T}_T\}$, we want to improve the predictor $f_T(\cdot)$ for $\mathcal{T}_T$ by incorporating information from $\{\mathcal{D}_S, \mathcal{T}_S\}$ and $f_S(\cdot)$. In general, we may even have more than one source problems. If $\{\mathcal{D}_T, \mathcal{T}_T\} = \{\mathcal{D}_S, \mathcal{T}_S\}$, then we recover the traditional *ML* setting.

There are three common metrics for measuring the performance of a *TL* method.

[3]Notations and definitions are adapted from [1],[2]

1. The difference in final performance of $f_T(\cdot)$ with and without transfer
2. The difference in steps before convergence of the algorithm with and without transfer
3. The initial performance of the algorithm (before additional training on the target task) with and without transfer

Note that 2 and 3 are only relevant for iterative algorithms, eg for methods using neural networks.

If information from the source problem reduces the performance of $f_T(\cdot)$ we say that *negative transfer* has occured. It is a major challenge in current *TL* research to reduce negative transfer automatically. In particular, recent research has tried to identify ways to measure relatedness between tasks which could potentially allow us to transfer the "right" amount of knowledge between them [1].

1.2 Transfer Learning Scenarios

Depending on the differences between $\{\mathcal{D}_T, \mathcal{T}_T\}$ and $\{\mathcal{D}_S, \mathcal{T}_S\}$ a number of different *TL* frameworks arise which need to be studied separately[4]. In this project our focus is mainly on *DA*, in which $\mathcal{T}_T = \mathcal{T}_S$ and $\mathcal{D}_T \neq \mathcal{D}_S$. *DA* problems are classified as *homogeneous* or *heterogeneous* depending (respectively) on whether $\mathcal{X}_S = \mathcal{X}_T$ or $\mathcal{X}_S \neq \mathcal{X}_T$. Under the assumption $\mathcal{T}_T = \mathcal{T}_S$ where the common label space is $\mathcal{Y}$, one might be tempted to think that a model for $P(Y|X_S)$ will work well for $P(Y|X_T)$ without further amendment (since the underlying distribution $P(Y|X)$ is the same in both cases), however, this is not the case in practice because $P(X_s) \neq P(X_T)$, which we refer to as *dataset bias* or *covariate shift*. A detailed argument and example for this can be found in [3].

In addition, in the context of *DA*, our focus is on problems where data from the source domain is much more abundant than data from the target domain and both source and target data are available during training. Furthermore, source data is assumed to be at least partially labeled, while target data may be unlabeled[5]. This framework is refered to as *TTL* and in this project we treat *DA* as part of *TTL*.

There are two main approaches to tackling *DA*, namely the *instance-based* approach and the *feature-representation* approach. The former is inspired by importance sampling[6], while the latter assumes the existence of a shared near-optimal inductive bias between tasks and attempts to find it. In particular, feature representation methods look for a transformation that maps both domains in an "intermediate" space in which their marginal distributions are similar. This procedure mitigates the effects of covariate shift.

[4] A description of the different *TL* scenarios that may arise is given in [1],[2]
[5] At least some labeled data may be needed as a validation set
[6] See [1], page 8 for a detailed argument

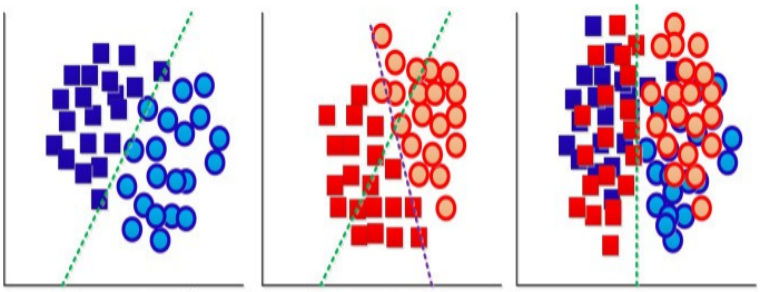

Figure 1.1: Illustration of covariate shift and its problems. On the left the source domain and a decision boundary for a classifier trained on the source domain. In the middle, the target domain juxtaposed with the source decision boundary and a decision boundary obtained by supervised learning on the target domain. On the right the two domains mapped to some latent space through domain adaptation. (Image taken from https://medium.com/deep-learning-domain-adaptation-on-image-segmentat/introduction-2b44dd49ea05, 10/4/2019)

The following informal discussion is aimed at motivating the use of feature representation transfer for *DA* further. Intuitively, if two domains are related we expect that there exist shared hidden variables which majorly affect the data in both domains. That is, there should be some shared latent space in which the distributions of the data from the two domains are very similar and which accounts for most of the variance in the common label space. In fact, we could informally think of how related two domains are, as the extent to which we can find a representation that allows us to perform well on both tasks simultaneously.

If we could discover the aforementioned latent space we could use it to train an algorithm on data from either domain to use on either task. This latent space is probably lower dimensional than the original domains because we expect the later to include other variables that "separate" the two tasks. Thus, most approaches to representation transfer are to perform some form of dimensionality reduction on the two domains, ensuring that the marginal distributions on the resulting space are similar. Note also that the resulting space must be "rich" enough to make the prediction of label variables easy.

1.3 Learning Bounds for Domain Adaptation

In this section we will present some theoretical guarantees for the accuracy of *DA*. Recall that for typical classification (ie under the assumption of identical marginals between training and test sets), the *VC* bounds from *PAC* learning theory give

$$\epsilon_{Test}(h) \leq \epsilon_{Train} + \mathcal{O}\Big(\sqrt{\frac{VC(H)}{N}\log(\frac{N}{VC(H)}) - \frac{\log(\delta)}{N}}\Big)$$

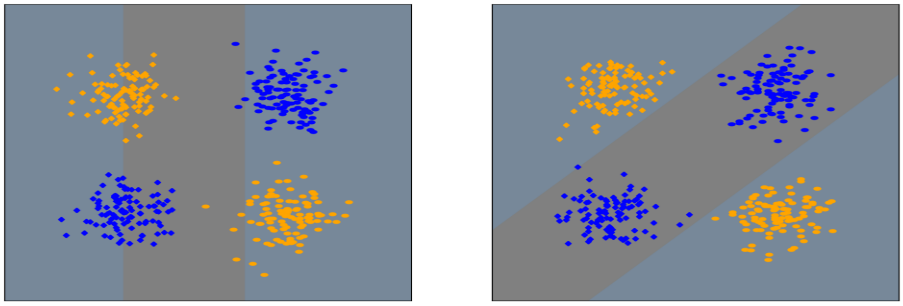

Figure 1.2: An example illustrating a good and a bad hypothesis space. In both pictures the blue points represent the source domain and the orange points the target domain. The circles represent points in class A and the rombuses represent points in class B. On the left the hypothesis space includes one parameter models specifying a vertical line. On the right our hypothesis space includes arbitrary linear models specified by two parameters. On the left H is not strong enough to separate the domains but it can easily separate the classes. This is not true on the right.

where H is our hypothesis space, $VC(H)$ is the VC-dimension and δ is the probability that the above inequallity will hold. When we relax the assumption of identical marginals, results established in [32] give the following bound

$$\epsilon_{Target}(h) \leq \epsilon_{Source}(h) + \mathcal{O}\Big(d_{H\Delta H}(D_S, D_T) + \sqrt{\frac{VC(H)}{N}\log(\frac{N}{VC(H)}) - \frac{\log(\delta)}{N}}\ \Big) + \lambda$$

where $\lambda = \min_{h \in H} \epsilon_{Target}(h) + \epsilon_{Source}(h)$. The expression $d_{H\Delta H}$ is defined as

$$d_{H\Delta H}(\pi_1, \pi_2) = \sup_{A \in \mathcal{A}_{H\Delta H}} |\pi_1(A) - \pi_2(A)|$$

where $H\Delta H = \{g \mid \exists h, h' \in H,\ \text{st}\ \forall x,\ g(x) = XOR(h(x), h'(x)\}$ is the *symmetric difference hypothesis space* and $\mathcal{A}_{H\Delta H} = \{A \mid \exists g \in H\Delta H,\ \text{st}\ g(x) = 1,\ iff\ x \in A\}$. In words, $d_{H\Delta H}$ resembles the total variation distance but with the supremum taken over all sets where two hypotheses in H disagree. By inspecting the above inequality we note two things. Firstly, to be able to guarantee some success in the adaptation procedure, it is necessary that there is a common near optimal inductive bias $h^* \in H$; otherwise the term λ may be too large. Secondly, we want our hypothesis space to be complex enough to discriminate between classes but not complex enough to discriminate between domains. It is readilly seen that a task is well suited for adaptation only if we can represent our instances in such a way, such that discriminating between classes is easier than discriminating between domains.

1.4 Shallow Representation Domain Adaptation

It is evident from the above discussion, that any algorithm for representation-based transfer should inccorporate some measure of distribution similarity. Usual choices involve the *KL* divergence, cross entropy and *JSD*. However, in most settings the use of these measures will require an intermediate density estimation step[7]. An alternative that allows us to skip this step is the *MMD* criterion [4].

In this section we will discuss the algorithm introduced in [5] for representation transfer. It is based on maximum variance unfolding [6] as a dimensionality reduction technique and it achieves distribution similarity by minimizing the *MMD* between domains in the new space. Some brief remarks from the theory of *RKHS* are necesary for this discussion and are presented below.

1.4.1 Note on kernels

A RKHS $\mathcal{H}$ over a domain $\mathcal{D}$ with kernel function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$, is a Hilbert space of real functions $f : \mathcal{X} \rightarrow \mathbb{R}$ with dot product $\langle \cdot, \cdot \rangle$ satisfying

$$\langle f, k(x, \cdot) \rangle = f(x) \tag{1.1}$$

and $\overline{\text{span}}\{k(x, \cdot) : x \in \mathcal{X}\} = \mathcal{H}$. (1.1) is refered to as the reproducing property and it implies that k is symmetric and positive semi-definite. The later means that for any finite sequence of points $X = \{x_1, ..., x_n\}$, the kernel matrix of k over X defined as

$$K_{ij} = k(x_i, x_j)$$

is positive semi-definite. Furthermore, for every symmetric positive-definite k there is an *RKHS* with kernel k. We call $\overline{\text{span}}(k(x, \cdot) : x \in S)$ a section of the kernel on S and $k(x, \cdot) = \phi(x)$ a *feature map* of k. A kernel is universal, if for any compact subset $Z \subseteq \mathcal{X}$, any continous real-valued function over Z can be approximated arbitrarily close by an element of the section of k on Z [7]. A *RKHS* is universal if its reproducing kernel is universal.

Given data in two domains $X = \{x_1, ..., x_{n_1}\}$ and $X' = \{x'_1, ..., x'_{n_2}\}$ the *MMD* empirical estimate for their distribution similarity is given by

$$\text{MMD}(X, X') = ||\frac{1}{n_1} \sum_{i=1}^{n_1} k(x_i, \cdot) - \frac{1}{n_2} \sum_{i=1}^{n_2} k(x'_i, \cdot)||_{\mathcal{H}} \tag{1.2}$$

In [4] and [7] detailed motivation for this measure is given. The *MMD* criterion only has the desired properties when the underlying kernel is universal. For this reason it is important to ensure that our kernel is universal in the discussion to come.

[7] This is not a problem in the case of neural networks, as this step comes essentially for free

It can be shown that if for any set X the induced kernel matrix is positive definite, then the kernel is universal [5]. The *MMD* criterion can be restated in terms of kernel matrices as follows

$$\text{MMD}(X, X') = \text{trace}(KL)$$

where,

$$K = \begin{bmatrix} K_X & K_{X,X'} \\ K_{X,X'}^T & K_{X'} \end{bmatrix}, \text{ and } L_{ij} = \begin{cases} \frac{1}{n_1^2}, \text{ if } i,j \leq n_1 \\ \frac{1}{n_2^2}, \text{ if } i,j > n_1 \\ -\frac{1}{n_1 n_2}, \ otherwise \end{cases}$$

Here $K_{X,X'}$ is the matrix with entries $K_{ij} = k(x_i, x'_j)$ for $x_i \in X$ and $x'j \in X'$. Finally, note that if a kernel can be written as

$$K = \tilde{K} + \epsilon I \tag{1.3}$$

with $\epsilon > 0$ and $\tilde{K} \succeq 0$, it is positive definite and hence universal.

1.4.2 Maximum Mean Discrepancy Embedding

The *MMDE* [7] is a dimensionality reduction technique, that ensures that the distributions of source and target domains in the resulting space are similar in the *MMD* sense. Given two domains X_S, X_T we want to find a mapping ψ such that it minimizes MMD$\big(\psi(X_S), \psi(X_T)\big)$. If $\phi(x)$ is the feature map of a universal kernel then for any arbitrary map ψ, $\phi \circ \psi(x)$ is the feature map of another universal kernel [5]. This effectively means that MMD$\big(\psi(X_S), \psi(X_T)\big)$ is equivalent to MMD(X_S, X_T) for some other universal kernel k'.

Now from what we discussed, the problem we need to solve looks like

$$\begin{aligned} \min_{K} \quad & \text{trace}(KL) \\ \text{subject to} \quad & K \succ 0 \end{aligned}$$

This however is not enough as we need the resulting space to be rich enough to allow good prediction. This is why the additional constraints below are added [6].

$$\begin{aligned} \min_{\tilde{K}} \quad & \text{trace}(\tilde{K}L) - \lambda \tilde{K} \\ \text{subject to} \quad & [1]\ \tilde{K}_{ii} + \tilde{K}_{jj} - 2\tilde{K}_{ij} + 2\epsilon = ||x_i - x_j||^2,\ \forall (i,j) \in \mathcal{N} \\ & [2]\ \tilde{K}\mathbf{1} = -\epsilon\mathbf{1} \\ & [3]\ \tilde{K} \succeq 0 \end{aligned}$$

Here, **1** is the vector of all ones and $\mathcal{N}$ is the set defined by: $(i,j) \in \mathcal{N}$ if x_i, x_j are δ-nearest neighbors of each other (for some predefined δ). The modifications to the problem essentialy mean that nearby points in the original space stay close after the transformation (constraint 1) and far away points get stretched maximally (-$\lambda\tilde{K}$ in objective). The second

constraint centers the data on the new space. Also, we change variables from $K \leftrightarrow \tilde{K}$, where $K = \tilde{K} + \epsilon I$ as this allows us to cast the problem as an *SDP* problem.

Now, having solved the *SDP* we can perform *PCA* on $\tilde{K}$ to obtain a low dimensional representation for our data. Since the data is centered, $\tilde{K}$ is related to the covariance matrix of our data in the new space and so this is justified. The obtained representation can be used to train a classifier on source data and it can then be used to solve the target task, as we no longer have the covariate shift problem and the latent space is rich enough to allow good prediction.

In practice, this algorithm does not scale up well. Firstly, even though *SDP* solvers have polynomial complexity, the fastest available ones have complexity worse than $\sim \mathcal{O}(n^3)$ [8]. This leads to an overall complexity of $\mathcal{O}((n_1 + n_2)^{6.5})$ for *MMDE* [9]. Furthermore, storing the involved matrices in memory for large problems may be infeasible. Overall, even though the above algorithm is theoretically appealing it is very computationally intensive and this hinders its scalability and practicality.

To elude the computational burden of *MMDE*, a modified version of this algorithm is proposed in [9] called Transfer Component Analysis, which also allows us to tackle partial domain adaptation problems.

1.5 Transfer via Fine-Tuning Neural Networks

1.5.1 Method Description

Fine-tuning methods for *DA* are amongst the most widely used ones.[8] The general procedure for these methods involves training a deep neural network on the source task (or assembly of source tasks) and using the learnt weights to initialize a network which is then trained ("fine-tuned") on the target task. Design choices for these algorithms are mainly concernced with the fine-tuning phase and examples include whether or not to freeze weights in lower layers and randomly initialize the remaining weights and whether to augment target data with source data. In the fine-tuning method framework, a "cut" on the source network at some layer l means that we are considering separately the layers from input up to and including l and the rest of the layers.

Informally, experiments suggest that features learned by lower layers (nearer to input) of the network are fairly general and can be transferred accross domains, while higher layer features are domain specific[9]. For example, deep neural networks trained on natural images tend to learn features similar to Gabor filters or color blobs irrespective of

[8] Strictly speaking these methods could be thought of as multi-task learning methods, however because the amount of labeled data may be very limited we consider it *DA*

[9] [10]

the objective function or the specific dataset used[10]. The experimental setup is described in detail in [10]. Note that for the above referenced experiments, the generality of a set of features for a task A is quantified only with respect to a second task B, so beneficial transfer is only guaranteed if the underlying datasets are related; it is not necesary that using low level features from an other task will be beneficial.

An interesting remark made in [10] is that even if the tasks between which we wish to transfer knowledge are related, choosing the number of layers to use from the source network is not very straightforward. Cutting a network of N layers at layer l and using layers $1 \dots, l$ to initialize the lower layers of a target network exhibits bad behaviour when $l \approx \frac{N}{2}$. In [10] it is argued that this observation is due to fragile co-adaptation developed between the network's intermediate layers, which can only be learned when layers are trained jointly (thus fine-tuning cannot fix this). Cutting the network near the middle breaks the co-adaptation between the two parts and thus should be avoided. On the other hand we want to incorporate as much "general" information from the source task as possible. So we need a way to deal with this dilemma.

For example, say we are using the first layer of the source network for transfer between two related datasets. It may be beneficial to use the second layer as well (as more information is adapted) but it might not be the case that using a third layer is beneficial; the third layer could be co-adapted with subsequent layers and provide bad information on its own. In addition, adding more and more layers (to incorporate co-adaptations) may lead to more domain-specific features, which is not desirable for beneficial transfer.

In practice these issues are dealt with using a validation set; We treat the cutting point l as a hyperparameter.

1.5.2 Extending these Ideas to Multi-Task Learning

MTL is part of the *ITL* framework, where we assume that there is available labelled data in both source and target problems, in similar volumes and both are available during trainining. Furthermore, we need not have shared domains or tasks between problems, eg $\mathcal{T}_S \neq \mathcal{T}_T$ and/or $\mathcal{D}_T \neq \mathcal{D}_S$. In contrast to *DA*, in *MTL* we do not distinguish between "source" and "target" problems, as this characterization is not really relevant, but instead index problems, *eg* $\{\mathcal{D}_i, \mathcal{T}_i\}_{i=1}^n$. *MTL* is *homogeneous* if all supervised tasks $\mathcal{T}_i$ are of the same type (for example classification/regression). In this section we consider homogeneous classification *MTL*. Similarly, when $\mathcal{D}_i$ share the same feature-space $\mathcal{X}$ we say we have *homogeneous-feature MTL*.

As mentioned earlier, fine-tuning methods are very closely linked with multi-task learning. In particular, when target and source data are both labelled we can extend the ideas from the previous section to develop a scheme for learning the tasks simultaneously. This

[10] ibid

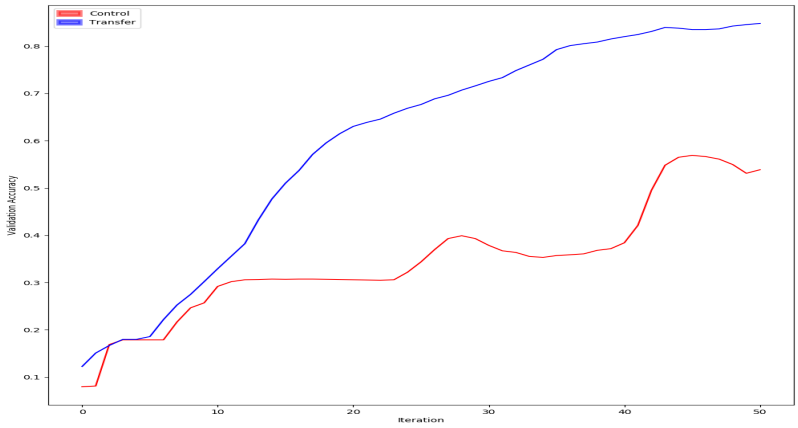

Figure 1.3: Tensorboard logs for training on 20 batches of 64 images from data consisting of 1000 random instances of the USPS dataset. A randomly initialized network (orange) vs a network with 2 lower layers pretrained on MNIST (blue). Networks have the same topology. It is clear from this example that initializing lower layers with weights obtained from a pretrained network results in faster learning and higher final accuracy. Accuracy after 20 epochs is $\approx 84\%$ with transfer and $\approx 53\%$ without.

serves as a form of regularization enabling the trained models to generalize better and additionally allows us to train more complex models on lesser data. The idea is straightforward: train two (or more) neural networks in an alternating fashion while a fixed number of their lower layers is shared between them. Although this method is not state-of-the-art in *MTL*, we present it here for completeness.

As discussed earlier, lower layers tend to be general and thus sharing them between models for related tasks is appropriate. The models can still become specific in their upper layers, although we cannot completely avoid task specific features creeping into the shared representation. As discussed in [12], it is possible to make multi-task learning more effective by slightly modifying this idea to incorporate adversarial methods. Here we present the benefits of multi-task learning experimentally.

For our experiment we trained four networks on MNIST and USPS, two of which are trained jointly (one on each task) while sharing their first layer and two of which are trained independently of each other (one on each task) which serve as controls. For each task we picked a subset of the labels such that the label spaces for each task slightly differ. Further, we used a small amount of data from each dataset. The results indicate that the classifier for task 2 performed significantly better than its control counterpart, while on task 1 there was little difference between the multitasking network and its control.

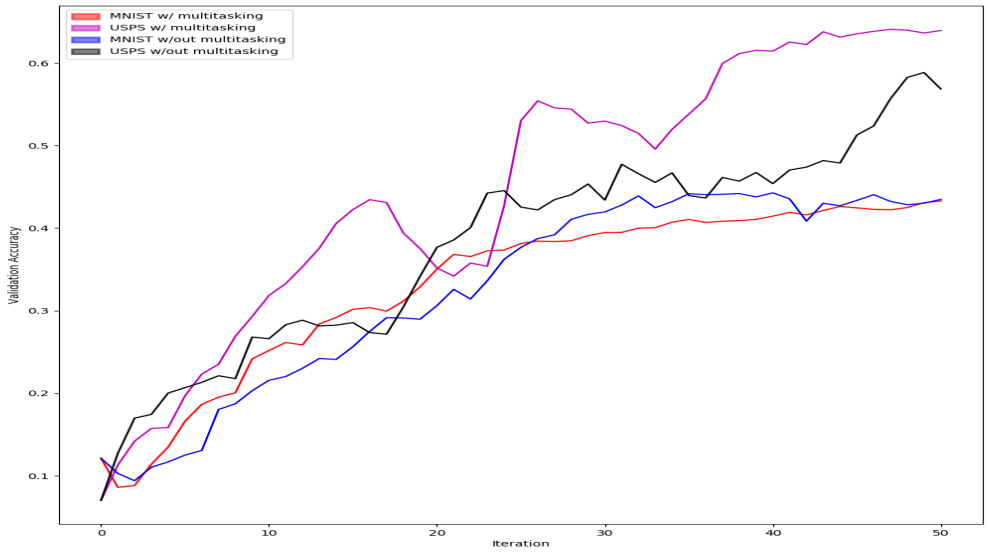

Figure 1.4: Tensorboard logs for the multi-task experiment.

1.6 Adversarial Domain Adaptation

In this section we describe adversarial methods for *DA*. The term adversarial methods broadly refers to a class of algorithms that train neural networks jointly with "competing" objectives. Adversarial networks for *DA* can be broken down as follows: There are two "representer" networks $\mathbf{M}_S$ and $\mathbf{M}_T$ mapping the source and target domains respetively to a common latent space and a domain discriminator network **D** mapping the latent space to the interval $[0, 1]$. Additionally, there is a classifier network **C** that maps the output of $\mathbf{M}_S$ to a distribution over the common label space. When $\mathbf{M}_S = \mathbf{M}_T$ we say that the method is *symmetric*. Symmetric methods are in general harder to train because they require more fine tuning and the classifier network should also be trained during transfer.

In the adaptation phase, $\mathbf{M}_S$ and $\mathbf{M}_T$ are trained so that **D** cannot discriminate between their outputs and so that **C** performs well on source data given input from $\mathbf{M}_S$. On the other hand, **D** is trained to distinguish the outputs of $\mathbf{M}_S$ and $\mathbf{M}_T$. As we will see later, under a suitable loss function, we can think of $\mathbf{D}(x)$'s as measuring the likelihood that x was generated by $\mathbf{M}_S$. We see thus that the representer networks and the discriminator are like adversaries. For asymmetric methods, the classifier is usually pretrained on source data and kept fixed during the adversarial training phase.

The aim is to reach an equillibrium (point of convergence) where **D** is identically 0.5 and where **C** performs well. This would imply that the distributions of $\mathbf{M}_S(x_s)_{x_S \sim P(X_S)}$ and

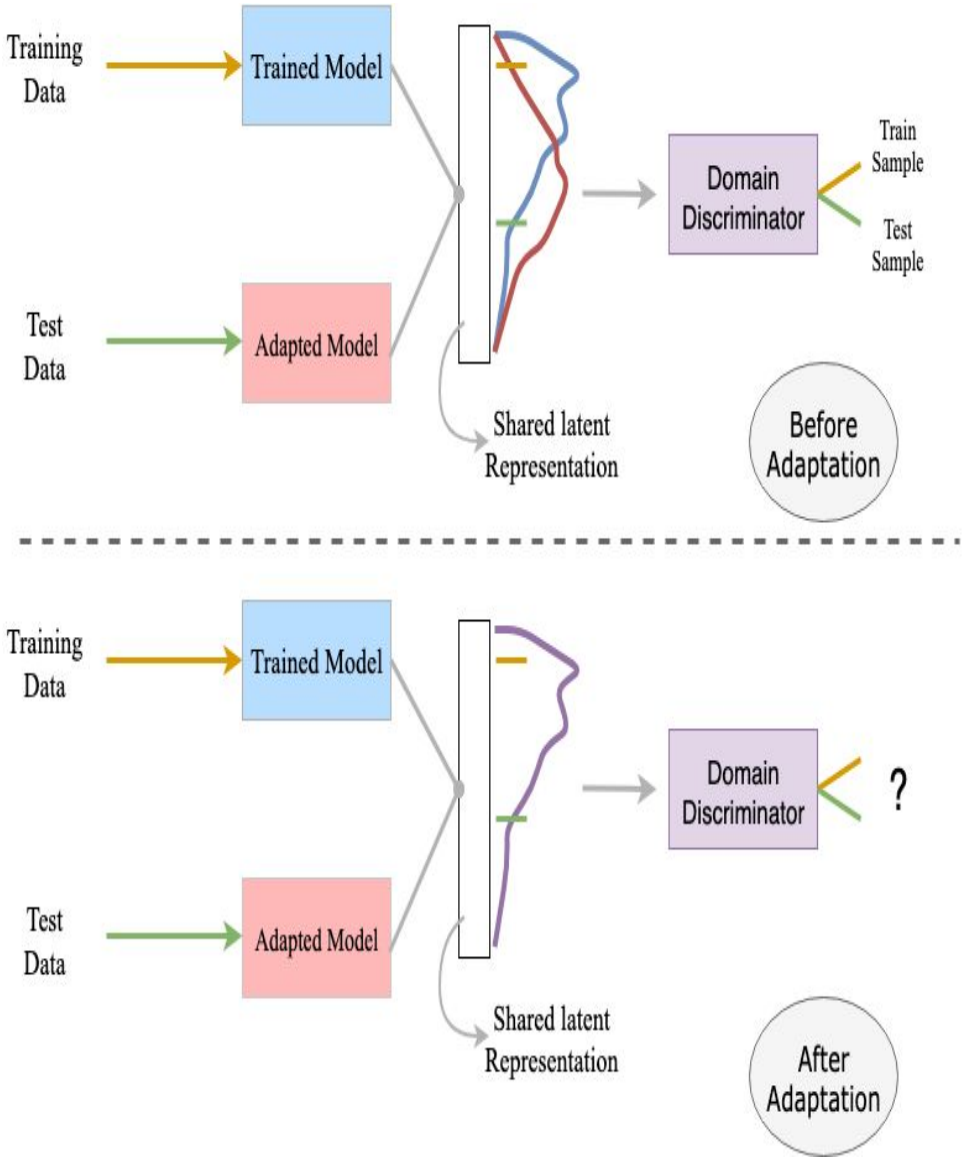

Figure 1.5: Illustration of the central idea in adversarial domain adaptation through the ADDA algorithm. Before adaptation, the latent distributions of source and target domains are different and the domain discriminator can distinguish them by selecting the most probable domain. When latent space distributions of source and target domains are aligned the domain discriminator cannot distinguish source from target instances.

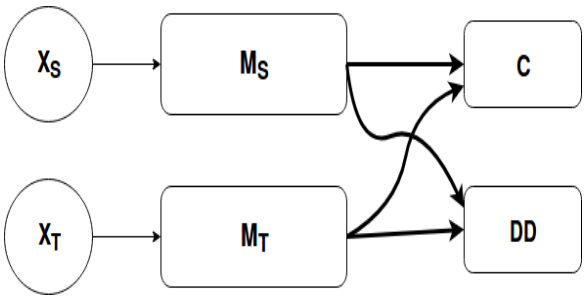

Figure 1.6: Diagram for general adversarial domain adaptation topology.

$\mathbf{M}_T(x_t)_{x_t \sim P(X_T)}$ are identical (or rather indistinguishable by an adversary with the complexity of **D**) and that $\mathbf{M}_S(x)$ is rich enough for $\mathcal{T}_S$. Because **C** provides a good model for $P(Y|\mathbf{M}_S(X_S))$, by the *DA* assumptions, it must also provide a good model for $P(Y|\mathbf{M}_T(X_T))$. So provided we can construct our network topologies and objective functions so that the above equillibrium is reachable, we can solve the covariate shift problem. In fact, here we have assumed $P(Y|\mathbf{M}_S(X)) \approx P(Y|\mathbf{M}_T(X))$ which is reasonable if $M_t \approx M_s$ or if good transfer has occured. Symmetric methods have the advantage that $P(Y|\mathbf{M}_S(X)) = P(Y|\mathbf{M}_T(X))$ comes for free.

In general adversarial methods for *DA* perform very well. However, there are some cases of problems where training the networks through the above procedure may not lead to a satisfying solution. For example, when the source and target domain distributions are multimodal, it may be hard to align them in latent space though gradient updates [13],[14]. That is, it is possible for **D** to be unable to discriminate between the two domains and $P(\mathbf{M}_S(x_s)_{x_S \sim P(X_S)}) \neq P(\mathbf{M}_T(x_t)_{x_t \sim P(X_T)})$. This is refered to as the mode collapse problem.

1.6.1 An Example: Adversarial Discriminative Domain Adaptation

One instantiation of the above setup is the *ADDA* algorithm [15]. *ADDA* uses loss functions

$$\mathcal{L}_C = -E_{x_s, y_s \sim X_S, Y_S}\big[\sum_{i=1}^{K} \mathbb{I}\{i = y_s\} \ \log C_i\big(\mathbf{M}_s(x_s)\big)\big]$$
$$\mathcal{L}_D = -E_{x_s \sim X_S}[\log(D\big(\mathbf{M}_s(x_s)\big))] - E_{x_t \sim X_T}[\log(1 - D\big(\mathbf{M}_t(x_t)\big))]$$
$$\mathcal{L}_M = -E_{x_t \sim X_T}[\log(D\big(\mathbf{M}_t(x_t)\big))]$$

where expetations are estimated as dataset averages. As a pre-training phase, the algorithm learns weights for $\mathbf{M}_s$ and **C** by minimizing

$$\min_{C, M_s} \mathcal{L}_C$$

over $\{\mathcal{D}_S, \mathcal{T}_S\}$. These weights are then frozen, ie they are not modified further for the rest of training. Note that the resulting **C** is essentially a model for $P(Y|\mathbf{M}_s(X_S))$, as $\mathcal{L}_C$ is the cross entropy between the empirical distribution for $P(Y|\mathbf{M}_s(X_S))$ and the inferred distribution $\mathbf{C}(\mathbf{M}(X_s))$. As this is minimized we have $\mathbf{C}(\mathbf{M}(X_s)) \approx P(Y|\mathbf{M}_s(X_S))$. The topology of the network $\mathbf{M}_t$ for the target domain is identical to $\mathbf{M}_s$ and $\mathbf{M}_t$ is initialized as $\mathbf{M}_s$ (with same weights) to avoid finding degenerate solutions. Then, adversarial training attempts to iteratively solve

$$\begin{aligned} &\min_{D} \mathcal{L}_D|_{\text{fixed } \mathbf{M}_t} \\ &\min_{\mathbf{M}_t} \mathcal{L}_M|_{\text{fixed } \mathbf{D}} \end{aligned} \tag{1.4}$$

Note that minimizing $\mathcal{L}_M$ is equivalent to maximizing $\mathcal{L}_D$ with respect to $\mathbf{M}_t$. The reason $\mathcal{L}_M \neq -\mathcal{L}_D$ is that the choice of $\mathcal{L}_M$ provides better gradient feedback early on during training and has the same stationary point properties as $-\mathcal{L}_D$ [14]. The loss function $\mathcal{L}_D$ is inspired from [13], and following similar arguments we can show it has the desired stationary point properties. In particular,

$$\begin{aligned} \mathcal{L}_D = &- \int_{\mathcal{X}_S} p_s(x_s) \log(\mathcal{D}(\mathbf{M}_s(x_s))) \, dx_s - \int_{\mathcal{X}_T} p_t(x_t) \log(\mathcal{D}(\mathbf{M}_t(x_t))) \, dx_t = \\ &- \int_{\mathcal{M}} p_{\mathbf{M}_s(X_S)}(m) \log(\mathcal{D}(m)) + p_{\mathbf{M}_t(X_t)}(m) \log(\mathcal{D}(m)) \, dm \end{aligned}$$

where $\mathcal{M}$ is the common latent space in which the image of $\mathbf{M}_s$ and $\mathbf{M}_t$ belong. This can be shown to have its optimal/minimum value (with respect to D and with fixed $\mathbf{M}_s$ and $\mathbf{M}_t$) at

$$D^*(m) = \frac{p_{\mathbf{M}_s(X_S)}(m)}{p_{\mathbf{M}_s(X_S)}(m) + p_{\mathbf{M}_t(X_T)}(m)} \tag{1.5}$$

This also justifies the intuition described earlier; that $D(\cdot)$ measures the likelihood that its input was generated from the source domain. Fixing $D(m) = D^*(m)$, $\mathcal{L}_D$ can be rewritten as (some more detail can be found in [13] and a similar proof is given for weighted distributions in Chapter 4 of this thesis)

$$\mathcal{L}_D = \log(4) - 2\text{JSD}(p_{\mathbf{M}_s(X_S)} || p_{\mathbf{M}_t(X_T)})$$

Thus, $\mathcal{L}_D$ is minimimized for fixed D when $p_{\mathbf{M}_s(X_S)} = p_{\mathbf{M}_t(X_T)}$. Thus, indeed the algorithm has an equillibrium with the desired properties, given the networks are complex enough to model these distributions.

1.6.2 Mitigating the Mode-Collapse Problem

As mentioned earlier an important limitation with classical adversarial methods is their ability to deal with the mode collapse problem. For this reason modifications such as *CADA*

[15] have been proposed.

CADA deals with the mode collapse problem, by conditioning training on **C**'s prediction. Label information can help reveal the multimodal structure of domain distributions making it easier to align distributions in latent space [15]. Specifically, we define a new mapping F with inputs the outputs of $\mathbf{M}_s$ and **C**. Then, **D** maps the output of F to $[0, 1]$ as before. A simple candidate for F is the vector concatenation function $F(x, y) = xy$.

We denote $h_s(x_s) = F\Big(\mathbf{M}_s(x_s), \mathbf{C}\big(\mathbf{M}_s(x_s)\big)\Big)$ and $h_t(x_t) = F\Big(\mathbf{M}_t(x_t), \mathbf{C}\big(\mathbf{M}_t(x_t)\big)\Big)$. The conditional objective for the network has the form

$$\mathcal{L}_D = -E_{x_s \sim X_S}[\log(D\big(h_s(x_s)\big))] - E_{x_t \sim X_T}[\log(1 - D\big(h_t(x_t)\big))]$$

and again expectations are estimated as dataset averages. This scheme may be adapted for different adversarial *DA* algorithms and depending on the specific algorithm used, different parameters of the network will be frozen during training. For example, we could use the above scheme as part of the *ADDA* algorithm described in the previous section, in which case **C** and $\mathbf{M}_s$ would be frozen during adversarial training.

Different mappings F represent different ways of incorporating label information to the discriminators input. In [15] it is argued that simple concatenation does not allow the model to fully capture interdependencies between the outputs of $\mathbf{M}_s$ and **C**. An alternative mapping is proposed based on estimating the tensor product of the two input vectors. Recall, the tensor product between two vectors is given by

$$v \otimes u = \begin{bmatrix} v_1 u_1 & v_1 u_2 & \dots & v_1 u_{n_u} \\ \vdots & & & \vdots \\ v_{n_v} u_1 & v_1 u_2 & \dots & v_{n_v} u_{n_u} \end{bmatrix}$$

of course the computation of $v \otimes u$ is expensive and in practice we cannot embed it in the adversarial network without dramatically increasing the number of parameters. The proposed solution is to estimate the tensor product using randomized algorithms [15].

WASSERSTEIN DOMAIN ADAPTATION

It should be clear from our discussion thus far that the problem of distribution alignment is of central importance to domain adaptation. We have also seen that this can be tackled by solving a minimization problem involving some "measure of distance" between distributions (eg *MMD*, *KL*-divergence etc..). These measures may be regarded as being better or worse depending on the task at hand. For example, we saw that using the *MMD* criterion in the *MMDE* algorithm allowed us to skip a density estimation step. Similarly, the *JSD* arises naturally under common objective functions for training adversarial networks. In this chapter, we will discuss a new measure of distance between distributions which has been succesfully used to tackle many hard problems in machine learning [16]; the Wasserstein distance. The reasons for this success will become apparent by the end of this chapter.

2.1 Wasserstein Distance

Formally, given two probability measures π_1, π_2 with finite moments of order p on $\mathbb{R}^d$ (endowed with the Borel σ-algebra on $\mathbb{R}^d$) we define the p-Wasserstein distance by

$$W_p(\pi_1, \pi_2) = \inf_{\pi} E[\ ||X - Y||^p\ ]^{\frac{1}{p}} = \inf_{\pi} \Big(\int_{\mathbb{R}^d \times \mathbb{R}^d} ||x - y||^p d\pi(x, y) \Big)^{\frac{1}{p}}$$

where the infimum is taken over all probabilistic couplings π of π_1 and π_2 and where X, Y are random vectors with marginal distributions π_1, π_2 respectively. Let $\mathcal{P}_p$, where $p \geq 1$ be the set of all probability measures on $\mathbb{R}^d$ that have finite moments of order p. Then it can be shown [17] that $W_p(X)$ is a metric on $\mathcal{P}_p$. This definition can be generalized to any separable, complete metric space. Informally, in the definition of the Wasserstein distance we are looking for a joint distribution (with marginals π_1, π_2) of X, Y so that under this joint distribution the expected value of $||X - Y||^p$ is minimized. For example, the coupling we need for $W_2(\pi_1, \pi_2)$ is the one that maximizes the correlation between X, Y [18]. In this project we will only be concerned with the $Wasserstein$-1 distance. For discrete distributions, $Wasserstein$-1 distance is also known as the *EM* distance.

2.1.1 Computing the Wasserstein Distance Between Discrete Distributions

To better understand the Wasserstein distance we present a method for computing it when the underlying marginals are discrete distributions. The minimization problem of computing $W(\pi_1, \pi_2)$ when π_1, π_2 are n-dimensional probability vectors can be cast as a linear program. To see this note that a probabilistic coupling π of π_1, π_2 can be represented as

an n^2 dimensional vector

$$\pi = \begin{pmatrix} \pi(x_1, y_1) \\ \pi(x_1, y_2) \\ \vdots \\ \pi(x_2, y_1) \\ \pi(x_2, y_2) \\ \vdots \\ \pi(x_n, y_n) \end{pmatrix}$$

Also define

$$C = (||x_1 - y_1||, \dots, ||x_1 - y_n||, ||x_2 - y_1||, \dots, ||x_2 - y_n||, ||x_n - y_1||, \dots, ||x_n - y_n||)$$

and a cost function

$$z = C \cdot \pi$$

we further need to introduce constraints to ensure that π is a valid probabilistic coupling of π_1, π_2. Fistly,

$$\pi \geq 0$$

then, define the concatenation $\pi_m = \begin{pmatrix} \pi_1 \\ \pi_2 \end{pmatrix}$ and the $2n \times n^2$ matrix

$$A = \begin{pmatrix} 1^n & 0^n & 0 \dots & \\ 0^n & 1^n & 0 \dots & \\ \hline 1, 0^{n-1} & 1, 0^{n-1} & 1, 0^{n-1} & \dots \\ 0, 1, 0^{n-2} & 0, 1, 0^{n-2} & 0, 1, 0^{n-2} & \dots \\ 0, 0, 1, 0^{n-3} & 0, 0, 1, 0^{n-3} & 0, 0, 1, 0^{n-3} & \dots \\ \vdots & & & \vdots \end{pmatrix}$$

we also need

$$A \cdot \pi = \pi_m$$

which ensures we have correct marginals. Overall, the linear program we need to solve is

$$\begin{aligned} \min_{\pi \in \mathbb{R}^{n^2}} \quad & z = C \cdot \pi \\ \text{subject to} \quad & A \cdot \pi = \pi_m \\ & \pi \geq 0 \end{aligned}$$

which can be solved by algorithms in the literature (eg simplex, interior point methods, ...). Figure 2.1, illustrates visually the solution to the above linear program for a toy example. Similar methods can be employed for higher order wasserstein distances in these discrete cases. However, the minimization problem when the underlying distributions are continous densities, is intractable. This is unfortunate, since, for domain adaptation, we are mainly interested in distributions with continuous support. This will be resolved by introducing the dual Kantorovich problem.

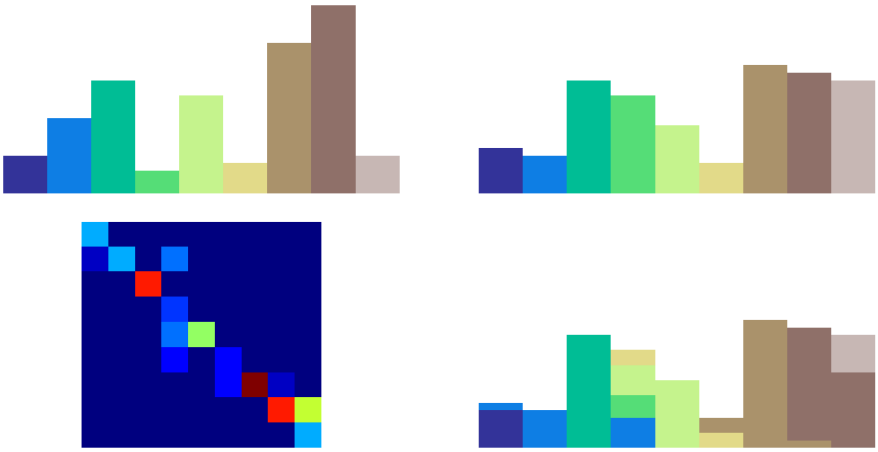

Figure 2.1: Visualization of an optimal transport plan with respect to the wasserstein distance between two discrete distributions. This was obtained by solving the aforementioned linear program. On the top left we see the source distribution and on the top right we see the target distribution. On the bottom left we see the optimal coupling encoded as a heatmap. On the bottom right we see, color coded, the optimal transport resulting from the optimal coupling.

2.1.2 Motivating the Use of the Wasserstein Distance

We now present an illustrative example[1] of why we would want to use an alternative to the *JSD* as a distribution similarity measure in adversatial *DA*.

Example (2.1.1) (Learning Parallel Lines)

Suppose $X \sim Unif(0,1)$. Let $g_\theta : \mathbb{R} \to \mathbb{R}^2 : x \to (\theta, x)$ and let π_0 be the distribution of $g_0(X)$ and π_θ the distribution of $g_\theta(X)$. The *JSD* between these two distributions is given by (for $\theta \neq 0$):

$$JS(\pi_0, \pi_\theta) = \frac{1}{2}\Big(\int I\{x=0\} \log\Big(\frac{2I\{x=0\}}{I\{x=0\}+I\{x=\theta\}}\Big)\, dx\, dy + \int I\{x=\theta\} \log\Big(\frac{2I\{x=\theta\}}{I\{x=0\}+I\{x=\theta\}}\Big)\, dx\, dy\Big) = log(2)$$

hence and since $JS(\pi_0, \pi_0) = 0$ we have,

$$JS(\pi_0, \pi_\theta) = \begin{cases} \log(2), \ if\ \theta \neq 0 \\ 0\ , \ if\ \theta = 0 \end{cases} \tag{2.1}$$

[1] Adapted from an example given in [16]

Note that if we wanted to adjust the parameter of g_θ so as to minimize $JS(\pi_0, \pi_\theta)$ we could not have used gradient methods as there are no usefull gradients in (2.1). This is a concern for us, since gradient feedback is essential to adversarial *DA*.

Note that the Wasserstein distance for these distributions is given by

$$W(\pi_0, \pi_\theta) = \inf_{\pi} E[\ ||g_0(X) - g_\theta(X)||\ ] = \inf_{\pi} \Big(\int\limits_{\mathbb{R}^d \times \mathbb{R}^d} ||g_0(X) - g_\theta(X)||\ d\pi \Big) =$$

$$= \inf_{\pi} \Big(\int\limits_{\mathbb{R}^d \times \mathbb{R}^d} ||(-\theta, 0)||\ d\pi \Big) \overset{(a)}{=} |\theta|$$

where (a) follows because π is a probability coupling. Thus we see that in this example the wasserstein distance gives us usefull gradient signals, which allow us to tune the parameter of g_θ. The behaviour noted in the above example can be justified theoretically and generalized as we will see later. We delve deeper in this topic in the following section.

2.2 Geometric Considerations for Convergence of Domain Adaptation

Following the discussion in [19], we start by introducing some useful concepts. Given two regular submanifolds M_1, M_2 of $\mathbb{R}^n$, we say that they *intersect transversally* at $x \in M_1 \cap M_2$ if $T_x M_1 + T_x M_2 = T_x \mathbb{R}^n$, where $T_x M$ denotes the tanget space at x of the manifold M. Otherwise, if M_1, M_2 do not intersect transversally at a point $x \in M_1 \cap M_2$, we say they *perfectly align* at x. We have the following theorem which explains the behaviour noted in example 2.1.1,

Theorem (2.2.1)

Let π_1, π_2 be continous and have supports contained in regular submanifolds M_1, M_2 of $\mathbb{R}^n$ respectively, such that M_1, M_2 have dimension $< n$ and do not perfectly align. Then

$$JS(\pi_1 || \pi_2) = log(2)$$

Proof

From the premise of the theorem, we know that at every point $x \in M_1 \cap M_2$ the manifolds intersect transversally. Let S_1, S_2 be the supports of π_1, π_2 respectively. We have

$$2JS(\pi_1, \pi_2) = \int\limits_{S_1} \pi_1(\vec{x}) \log(\frac{2\pi_1(\vec{x})}{\pi_1(\vec{x}) + \pi_2(\vec{x})}) d\vec{x} + \int\limits_{S_2} \pi_2(\vec{x}) \log(\frac{2\pi_2(\vec{x})}{\pi_1(\vec{x}) + \pi_2(\vec{x})}) d\vec{x}$$

Lemma: If M_1, M_2 are as in the premise of the theorem, then $M_1 \cap M_2$ has measure 0 in both M_1 and M_2.

Proof of Lemma:
If $M_1 \cap M_2 = \emptyset$ then the lemma is true. If not, then $M_1 \cap M_2 = M'$ must also be a regular submanifold of $\mathbb{R}^n$. It is enough to prove that M' is a strictly lower dimensional submanifold of both M_1 and M_2 since then it will have measure zero in both.

Suppose for a contradiction and without loss of generality, that $dim(M') = dim(M_1)$. Then, considering tangent spaces as embedded in $\mathbb{R}^n$, $T_xM' = T_xM_1$ for all x, since $T_xM' \subseteq T_xM$ and they have the same dimension. But then $T_xM_1 = T_xM' \subseteq T_xM_2$ (†). Since no perfect alignment occurs, $n = dim(T_xM_1 + T_xM_2)$ and because of (†), $dim(T_xM_1 + T_xM_2) = dim(T_xM_2) \Rightarrow dim(M_2) = n$. A contradiction since our assumption was that $dim(M_1), dim(M_2) < n$. □

We now have

$$\int_{S_1} \pi_1(\vec{x}) \log(\frac{2\pi_1(\vec{x})}{\pi_1(\vec{x}) + \pi_2(\vec{x})}) d\vec{x} =$$

$$\int_{S_1 \setminus S_2} \pi_1(\vec{x}) \log(\frac{2\pi_1(\vec{x})}{\pi_1(\vec{x}) + \pi_2(\vec{x})}) d\vec{x} + \int_{S_1 \cap S_2} \pi_1(\vec{x}) \log(\frac{2\pi_1(\vec{x})}{\pi_1(\vec{x}) + \pi_2(\vec{x})}) d\vec{x} \to 0$$

where the second term is zero since we integrate over a set of measure zero. In the first term the domain of integration does not intersect the support of π_2 and has full measure. Hence, we have

$$\int_{S_1} \pi_1(\vec{x}) \log(\frac{2\pi_1(\vec{x})}{\pi_1(\vec{x}) + \pi_2(\vec{x})}) d\vec{x} = \int_{S_1 \setminus S_2} \pi_1(\vec{x}) \log(\frac{2\pi_1(\vec{x})}{\pi_1(\vec{x})}) d\vec{x} = \log(2)$$

Similar arguments show

$$\int_{S_2} \pi_2(\vec{x}) \log(\frac{2\pi_1(\vec{x})}{\pi_1(\vec{x}) + \pi_2(\vec{x})}) d\vec{x} = \log(2)$$

and thus $JS(\pi_1 || \pi_2) = \log(2)$ □

The above result shows that there are cases where *JSD* will simply not work for our purposes. Naturally, the question arises of whether we should be worried about this in practice. The following result provides some evidence that the aforementioned behaviour should not be neglected

Theorem (2.2.2)

Let M_1, M_2 be regular submanifolds of $\mathbb{R}^n$ such that their respective dimensions are $< n$.

Take X, Y independent and continous random variables with image in $\mathbb{R}^n$. Let x and y be vectors drawn from the distributions of X and Y respectively and define $\tilde{M}_1 = M_1 + x$ and $\tilde{M}_2 = M_2 + y$. Then

$$Pr(\tilde{M}_1 \text{ perfectly aligns with } \tilde{M}_2) = 0$$

Proof:

See [19], Lemma 2. □

It is hypothesized that in general, the support of real world data distributions indeed lies on lower dimensional submanifolds of the respective feature space. This is known as the manifold hypothesis. Note however, that by mapping our data onto lower dimensional spaces (eg using a deep neural network to map data points to a lower dimensional latent space) we can make the supports of the transformed distributions overlap significantly. This may, of course, lead to loss of information (eg mapping everything to a single point) and even when the mapping is useful, points x for which $\frac{p_S(x)}{p_T(x)}$ is either too large or too small will not have significant contribution to the gradients in adversarial training [20].

2.3 Wasserstein Distance Works

We will now present some theorems and sketch proofs (adapted from [16]) in order to better understand the reasons that the Wasserstein distance is more suited for our purposes. For the rest of our discussion we will assume, unless otherwise stated, that all distributions have finite means.

Theorem (2.3.1)

Let π_Y be a probability distribution over some compact space $\mathcal{Y}$ and X a random variable over another space $\mathcal{X}$. Suppose that $g : \mathcal{X} \times \mathbb{R}^n \to \mathcal{Y} : \mathbb{R}^n \ni \theta \to g_\theta : \mathcal{X} \to \mathcal{Y}$ is a parametric function. Finally, let π_θ be the distribution of $g_\theta(X)$. Then,

1. If g is continous with respect to θ, then $W(\pi_Y, \pi_\theta)$ is continous with respect to θ.
2. If g is locally Lipschitz with local Lipschitz constants that have finite expectation with respect to the distribution of g's input, then $W(\pi_Y, \pi_\theta)$ is continous everywhere and differentiable almost everywhere.

Proof (Sketch)

Given two points θ, θ' in $\mathbb{R}^n$ we have $||g_\theta - g_{\theta'}|| \to 0$ as $\theta' \to \theta$. But the joint distribution of the vector $(g_\theta(x), g_{\theta'}(x))$ is a probabilty coupling with appropriate marginal distributions hence $0 \leq W(\pi_\theta, \pi_{\theta'}) \leq E[||g_\theta - g_{\theta'}||]$ (since W is has an inf over probability couplings) and thus $W(\pi_\theta, \pi_{\theta'}) \to 0$ as $\theta \to \theta'$. Since $W(\cdot, \cdot)$ is a metric we have (follows from the triangle

inequality)

$$0 \leq |W(\pi_Y, \pi_\theta) - W(\pi_y, \pi_{\theta'})| \leq W(\pi_\theta, \pi_{\theta'}) \to 0$$

Which gives claim 1.

Now if g is as in 2 we want to show $W(\pi_Y, \pi_\theta)$ has the same properties. For any point in g's domain $p = (x, \theta)$ there is an open neighborhood U of p and a local Lipschitz constant L_p, such that $\forall q = (x', \theta') \in U$ we have

$$||g(p) - g(p')|| \leq L_p ||p - p'||$$

As we let $x \to x'$ and if $(x, \theta') \in U$ $(\dagger)$, we get

$$E[||g(p) - g(p')||] \leq ||\theta - \theta'|| E[L_p]$$

By the previous argument this gives

$$|W(\pi_Y, \pi_\theta) - W(\pi_y, \pi_{\theta'})| \leq ||\theta - \theta'|| E[L_p]$$

Since the set U_θ of parameters θ that satisfy $(\dagger)$ is open in $\mathbb{R}^n$ (because U was open) and $E[L_p]$ is finite, we can set $L_\theta = E[L_p]$ and we have

$$|W(\pi_Y, \pi_\theta) - W(\pi_y, \pi_{\theta'})| \leq L_\theta ||\theta - \theta'||$$

and this is true for all parameters θ in U_θ. Hence, W is locally Lipschitz and thus continous everywhere and differentiable almost everywhere. □

Note that these nice properties of W do not hold in general for the *JSD*; indeed Example 2.1.1 demonstrates that the *JSD* can be discontinous.

Even though there seems to be reasons to prefer the Wasserstein distance over the *JSD* we have already seen that *JSD* works in practice for certain problems. It would be reassuring to know that whenever the *JSD* works so will the Wasserstein distance. Essentially, this is taken care of by the next theorem, but before we procceed we present a small bit of theory.

The *TV* distance between probability measures on the same measurable space (say with the Borel σ-algebra) $(\mathcal{X}, Borel(\mathcal{X}))$ is defined as

$$TV(\pi_i, \pi) = \sup_{A \in Borel(\mathcal{X})} |\pi_i(A) - \pi(A)|$$

The *TV* distance satisfies all properties of a metric and induces another way of comparing probability measures. It can be shown that it has similar properties to *JSD* [16]. For the

next proof we need an important result called *Pinsker's Inequality*. This states that

$$TV(\pi_1, \pi_2) \leq \sqrt{\frac{KL(\pi_1||\pi_2)}{2}} \tag{2.2}$$

where KL denotes the *KL* divergence. Recall also that by definition, we have

$$JS(\pi_1, \pi_2) = KL(\pi_1||\pi_m) + KL(\pi_2||\pi_m) \tag{2.3}$$

where we have that

$$\pi_m = \frac{\pi_1 + \pi_2}{2}$$

Theorem (2.3.2)

Suppose that π is a distribution on a separable complete metric space $\mathcal{X}$ and let $\{\pi_i\}_{i\in\mathbb{N}}$ be a sequence of probability distributions defined on $\mathcal{X}$. Then,

$$\lim_{i\to\infty} JS(\pi_i, \pi) \to 0 \Rightarrow \lim_{i\to\infty} W(\pi_i, \pi) \to 0$$

Proof (Sketch)

$Lemma$ $(\dagger)$

$$\lim_{i\to\infty} JS(\pi_i, \pi) \to 0 \Rightarrow \lim_{i\to\infty} TV(\pi_i, \pi) \to 0$$

Proof of $Lemma$:

$$TV(\pi_i, \pi) \underset{(a)}{\leq} TV(\pi_i, \pi_m) + TV(\pi_m, \pi) \underset{(b)}{\leq} \sqrt{\frac{KL(\pi_i||\pi_m)}{2}} + \sqrt{\frac{KL(\pi||\pi_m)}{2}} = Q$$

Clearly, since $KL(\cdot||\cdot) \geq 0$, if $JS(\pi_i, \pi) \to 0$ then $Q \to 0$ and so $TV(\pi_i, \pi) \to 0$.

From $(\dagger)$ and Theorem 6.15 in [21] the claim trivially follows. □

The two theorems presented in this section tell us that Wasserstein distance has better properties than the *JSD* and that, loosely speaking, we don't loose anything from using W over *JSD* (excluding computational considerations).

2.4 Wasserstein Adversarial Networks

By now, we have motivated the use of the Wasserstein distance and we now address issues of computing it. In general, computing W directly is made intractable by the infimum in its definition. Hence, we present an important result that is key for W's computation.

Theorem (2.2.1) [Kantorovich-Rubinstein Duality (Special Case)] [21]

Let $\mathcal{J}$ denote the space of all 1-Lipschitz functions on a separable complete metric space $\mathcal{X}$ (eg $\mathbb{R}$). Then,

$$W(\pi_1, \pi_2) = \sup_{||\psi|| \leq 1} E_{\pi_1}[\psi(x)] - E_{\pi_2}[\psi(x)] \tag{2.4}$$

as long as the supremum on the right hand side is finite. In the above equation $\psi \in \mathcal{J}$.

Proof:

See [21], Chapter 5. □

In particular [16], we could consider defining a family of parametric K-Lipschitz functions $\{f_w\}_{w \in W}$ parametrized by a parameter vector w and solving the following maximization problem

$$\max_{w} E_{\pi_1}[f_w(x)] - E_{\pi_2}[f_w(x)] \tag{2.5}$$

if (2.5) is solved and the supremum of (2.4) is reached then we have succesfully estimated $W(\pi_1, \pi_2)$ up to a multiplicative constant (since $||f_w||$ may be ≥ 1).

The following theorem gives some key insights into why adversarial networks are particularly well-suited for using the Wasserstein distance as a distribution alignment metric.

Theorem (2.2.2)

Suppose that $g : \mathcal{X} \times \mathbb{R}^n \to \mathcal{Y} : \mathbb{R}^n \ni \theta \to g_\theta : \mathcal{X} \to \mathcal{Y}$ is a parametric function. Let also, Z be a random variable such that $Z \sim p$ and $g_\theta(Z) \sim \pi_2$. Moreover, let g be locally Lipschitz with local Lipschitz constants that have finite expectation with respect to p. Let also π_1 be a second distribution. Then the supremum in equation (2.4) is attained, ie

$$\exists \psi = \arg \max_{||\psi|| \leq 1} E_{\pi_1}[\psi(x)] - E_{\pi_2}[\psi(x)]$$

and (if both terms are well defined)

$$\nabla_\theta W(\pi_1, \pi_2) = -E_p[\nabla_\theta \psi(g_\theta(z))]$$

Proof:

See [16], Appendix C. □

Informally, using the idea expressed earlier (2.5) for approximating the function ψ by a parametric function f_w (eg. a neural network) we get a well defined scheme for minimizing the wasserstein distance between a distribution π_1 and the output distribution of a parametric function g_θ. In particular, consider performing the following computations in an

alternating manner

$$\max_{w} E_{\pi_1}[f_w(x)] - E_p[f_w(g_\theta(z))] \mid_{\text{fixed } \theta}$$
$$\min_{\theta} E_{\pi_1}[f_w(x)] - E_p[f_w(g_\theta(z))] \mid_{\text{fixed } w}$$

for example, using a stochastic gradient descent scheme. At each step, we want to find optimal parameters w in order to best approximate the gradient of $W(\pi_1, \pi_2)$ and so the maximization process above should be repeated until convergence at each step. Under some mild assumptions (eg w lives in a compact space, f, g have sufficient capacity etc) this scheme can be shown to theoretically converge (at least to a local minimum of W). In practice, problems may arise due to limited capacity of the underlying models and approximation errors. Convergence of this scheme in the context of *WGAN* is studied in [22].

Suppose now, for this informal discussion, that our parametric models have enough capacity to approximate ψ and capture the distribution π_1 fully and that f_w is K-Lipschitz $\forall w$. Then training f_w to optimality for the current θ gives us the direction of the gradient of $W(\pi_1, \pi_2)$ with respect to θ which allows us to update θ so as to minimize $W(\pi_1, \pi_2)$. Of course, after a couple of steps we need to return and tune w since the second term $-E_p[f_w(g_\theta(z))]$ has changed (due to the change in θ). The optimal scenario is that this process will continue until π_1 and π_2 are aligned (ie $W(\pi_1, \pi_2)$ is minimized).

There are still some important points to be addressed. Firstly, we want to use neural networks as our parametric models for f_w and g_θ so as to provide our models with the necessary capacity. The first issue we should address, is ensuring that f_w is K-Lipschitz $\forall w$, which of course may not be true for a general neural network. However, by restricting the network's weights to lie in a compact space W (for example the interval $[-0.5, 0.5]$), f_w can be made to satisfy the Lipschitz condition [16][2]. This could be done for example through weight clipping or projecting the weights to a sphere [16]. Furthermore, we can also use a neural network for g_θ due to the following result.

Theorem (2.2.3)

Let g_θ be a feedforward neural network with smooth Lipschitz activation functions and let its input be distributed with finite mean. Let also, π_1 be any distribution with finite mean and π_2 the distribution of the output of g_θ. Then,

1. $W(\pi_1, \pi_2)$ is continous everywhere and differentiable almost everywhere
2. g is locally Lipschitz with local Lipschitz constants that have finite expectation with respect to the distribution of g's input

Proof:

[2] note that the weight clipping method is not endorsed by the authors

The first claim follows from the second claim and the premise of the theorem using theorem (2.3.1).

The proof for the second claim is found at [16], Appendix C. □

So we see that the Wasserstein distance is a natural metric for aligning distributions in adversarial neural networks. We proceed to discuss how this observation can help us tackle the domain adaptation problem more effectively.

2.4.1 On Enforcing the Lipschitz Condition on f_w

As we already saw in Wasserstein adversarial optimisation that we need the parametric approximator f_w to satisfy a Lipschitz condition. In [16], the proposed way to enforce this was through weight clipping, even though the authors themselves were reluctant to using this method. In [22] an alternative gradient penalty method is proposed.

2.5 Wasserstein Domain Adaptation

In this section we will present an algorithm that incorporates the Wasserstein distance in an adversarial architecture for domain adaptation. This algorithm is essentially an adaptation of *ADDA* to use a Wasserstein domain critic instead of a vanilla domain discriminator. We term this algorithm *WADDA*. *WADDA* is similar to the *WDGRL* scheme introduced in [20]. However, in *WADDA* we allow the target network to have its own set of parameters, that is $\mathbf{M}_S \neq \mathbf{M}_T$ and we keep $\mathbf{M}_S$ fixed during training (we also have a source task pretraining phase like in *ADDA*). On the other hand, in *WDGRL* the source and task feature extractors are tied and the classifier is trained in an alternating manner together with the representer network. Because the source representer and classifier are trained during the transfer, training may become unstable much more easilly.

The objective functions for our algorithm is as follows:

$$\mathcal{L}_D = E_{x_t \sim X_T}[D(\mathbf{M}_T(x_t))] - E_{x_s \sim X_S}[D(\mathbf{M}_S(x_s))]$$

$$\mathcal{L}_T = \lambda E[H(C(\mathbf{M}^T(x_t)))] - E_{x_t \sim X_T}[D(\mathbf{M}_T(x_t))]$$

and alternating stocahstic minimization is performed as in

$$\min_D \mathcal{L}_D|_{fixed\ \mathbf{M}_T}$$

$$\min_{\mathbf{M}_T} \mathcal{L}_T|_{fixed\ D}$$

Note that as suggested by the results we presented in this chapter, this alternating optimisation decreases the Wasserstein distance between the latent distributions $\mathbf{M}_T(X_T)$ and

$\mathbf{M}_S(X_S)$ of source and target datasets. Furthermore, we expect this scheme to have better convergence properties that the traditional one (that uses the *JSD*) due to the superior gradients of the Wasserstein distance. Note that in each iteration we train the discriminator critic multiple times. This is because we want to obtain optimal parameters for D given the fixed $\mathbf{M}_T$ in order to best approximate the gradient of the Wasserstein distance.

Note also that in the representer objective function we have added *entropy regularization*. This idea is discussed in [23] and it has been previously used succesfully in a partial domain adaptation algorithm which we will discuss shortly. As explained in [23], in statistical learning the information we extract from unlabelled data comes from their structure. For any unsupervised learning algorithm to work succesfully, it is of paramount importance that it takes advantage of the structure of its input. Furthermore, in any unsupervised task it is natural to assume that meaningful structure is present in our data; otherwise no information can be gained.

In particular, for our problem we can use the classifier's output to regularize representation learning so as to preserve the target data structure as much as possible. We add to our objective function a term

$$\lambda E[H(C(\mathbf{M}^T(x_t)))]$$

where H denotes the information entropy and λ is a trade-off parameter. Minimizing this quantity with respect to the representer network parameters encourages the target latent representation to keep classes well separated, thus preserving the original structure.

EXPERIMENTS: FULL DOMAIN ADAPTATION

In this chapter we will list a number of domain adaptation tasks that will be of interest to us and we will conduct experiments to determine how effectively we can solve them. In particular, we will be interested in the following

1. Image obstruction
2. Image displacement
3. Image rotation
4. Image rescaling

We will use the MNIST and USPS datasets for our experiments. These are widely used datasets of hand-written digits consisting of gray-scale 28×28-pixel images. Specifically, we train a source model on the MNIST dataset and then perform the above operations to different extents on the USPS dataset and attempt transfer. Below we present our results. We will compare *ADDA* with *WADDA* and at the end of this chapter we present a "beat-the-benchmark" table where we tune *WADDA* to obtain very good performance on two tasks. These are marked with an * in the comparative tables. For the comparison we will set the trade-off parameter λ to zero as we are interested in comparing the wasserstein gradients to the *JSD* gradients alone. For beating the benchmark in the end, λ we is tuned.

We start with image obstruction where we put a "solid block" of $n \times n$ pixels in a location of the image (ie we set all pixel values in the block to 1's). We test different values of n. Next we test the displacement task. We displace the digits by (i, j) pixels to the horizontal and vertical directions respectively. We then test the rotation task, where we rotate the digits by an angle θ anti-clockwise. Finally, we test the rescaling task where we rescale the digits by a factor (i, j) along the horizontal and vertical axis respectively. For all our experiments, unless otherwise specified, the discriminator is trained on 10 batches for each batch the representer is trained on.

Furthermore, during the first epoch we allow the discriminator to train for 25 iterations in order to give it a "head-start". This strategy is intuitive and we empirically verified that this works nicely. All our experiments were performed on a Tesla K40c NVIDIA GPU and we performed each test 5 times and present the average accuracy. The Adam optimizer was used for all trials and the clip parameter for the weights in *WADDA* was set to 0.05.

We note that for *ADDA* too many iterations for the discriminator per iteration of the representer (depending on the task at hand) may lead to a perfect discriminator, in which case we have zero feedback to train the representer. This however is not a problem in *WADDA*, where the domain critic may be trained as long as we want. Furthermore, we observed

that *WADDA* with weight clipping suffers from vanishing/exploding gradients and we found that normalizing the gradients during training is always beneficial. In particular, we normalized gradients using the L_2 norm.

In summary, *ADDA* performs better with relatively balanced training in discriminator and representer, while *WADDA* performs better for more iterations of the critic and small learning rates for the representer. For this reason, in this comparative experiments we used the best hyperparameters we found for each method (including iterations D/R and learning rates). The most important parameter related to the networks' topology is the dimension of the latent space, ie the size of the output of the domain representer networks. This conclusion is drawn from what we have seen in Chapter 2 relating to *JSD* and *Wassertein* distance. Indeed, whene the dimension of the latent space is too large we note that *ADDA* does not perform well (there is slightly any change in the representer parameters). This problem was not observed for *WADDA*. For this reason we compare the methods with a network topology with relatively few latent dimensions (64).

For beating the benchmark we changed the topology of the discriminator from a simple multi-layer perceptron to a residual network. This further amends the vanishing/exploding gradient problem observed when weight clipping is used.

For the obstructed image task, both methods yield a significant improvement compared to the source network. *WADDA* consistently outperforms *ADDA* by a small margin. We make similar observations for the displacement task. In the rotation task we observe a sharp drop in the relative performance of *WADDA* with respect to *ADDA* which is indeed very interesting and should be investigated further. Finally, for the rescaling task, both methods perform quite well but *WADDA* offers significant improvement in the harder trials (ie with very small rescaling factors).

Table 3.1: Results for obstruction task

# of blocked pixels (n)	ADDA	WADDA-H	Source Network
5	85.15%	**86.45%**	69.51%
7	76.04%	**77.69%**	62.28%
14	74.47%	**79.82%**	48.87%
15	63.42%	**65.01%**	34.68%

Comparison of ADDA and WADDA-H on the obstructed image task. We let the networks train for 100 iterations. For this task, in WADDA-H we used $\lambda = 0$.

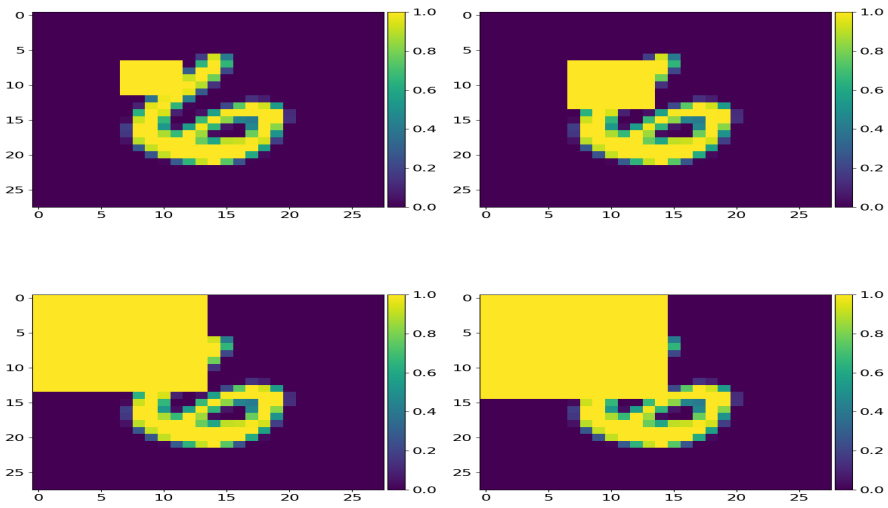

Figure 3.1: Sample from the obstructed USPS dataset for different values of n

Table 3.2: Results for displacement task

(i, j)	ADDA	WADDA-H	Source Network
(3,2)	65.71%	**76.62%**	58.59%
(7,0)	88.78%	**88.85%**	65.87%
(0,3)	68.83%	**73.98%**	57.19%
(5,3)	45.49%	**47.63%**	43.54%

Comparison of ADDA and WADDA-H on the displaced image task. We let the networks train for 100 iterations. Red indicates the occurence of negative transfer. For this task, in WADDA-H we used $\lambda = 0$

Table 3.3: Results for rotation task

θ **in degrees**	ADDA	WADDA-H	Source Network
15	93.82%	92.58%	93.67%
-30	87.74%	85.03%	74.73%
40	75.23%	74.91%	66.11%
60	45.39%	42.05%	41.90%

Comparison of ADDA and WADDA-H on the displaced image task. We let the networks train for 100 iterations.

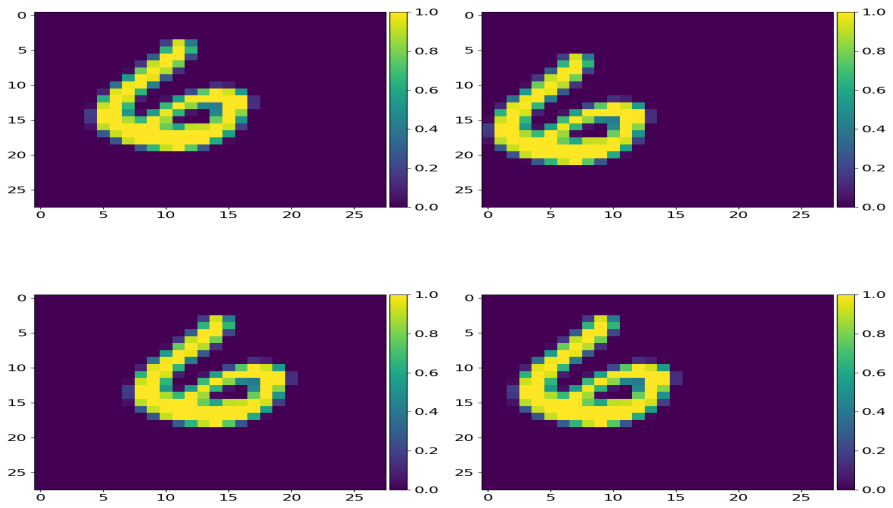

Figure 3.2: Sample from the displaced USPS dataset for different values of (i, j)

Table 3.4: Results for rescaling task

(i, j)	ADDA	WADDA-H	Source Network
(1.3,0.7)	78.97%	79.13% **	58.20%
(0.8,0.7)	73.59%	73.69%	54.11%
(1,1.5)	92.57%	**94.93%****	91.87%
(0.6,0.6)	55.51%	**63.71%**	45.24%

Comparison of ADDA and WADDA-H on the displaced image task. We let the networks train for 100 iterations. For ** the number of discriminator iterations per representer iterations (D/R) for WAADDA-H were increased to 25 to obtain increased performance. The same change for ADDA decreases its performance so we used 10D/1R

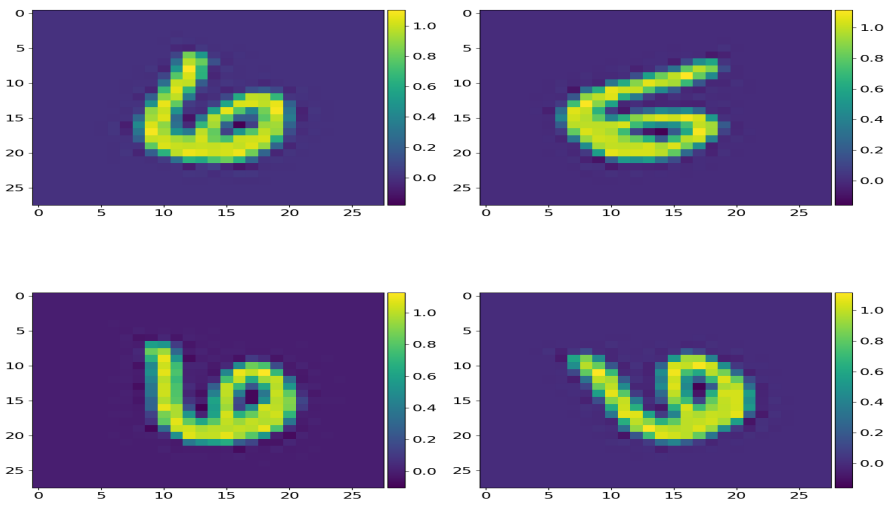

Figure 3.3: Sample from the rotated USPS dataset for different values of θ

Table 3.5: Results for beat-the-benchmark

Task	Parameters	iterations	accuracy	source accuracy
obstruction, $n = 15$	$D/R = 50$	400	**74.39%**	34.68%
displacement, $(i, j) = (5, 3)$	$D/R = 50$	500	**65.22%**	43.54%

D/R denotes the number of discriminator iterations per representer iteration. Learning rates were 5×10^{-5} and 10^{-5} for the discriminator and representer respectively.

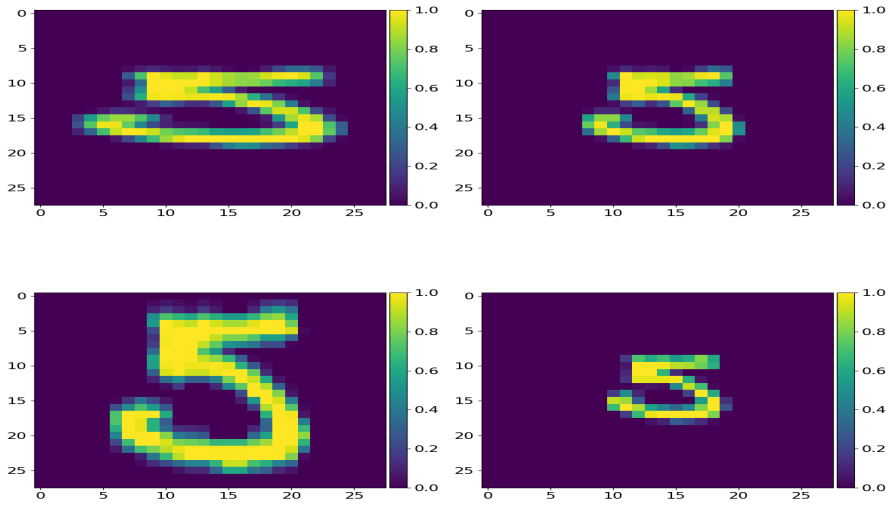

Figure 3.4: Sample from the rescaled USPS dataset for different values of (i, j)

PARTIAL DOMAIN ADAPTATION

4.1 Problem Description

In the first chapter, we discussed some important methods for domain adaptation and ran our own experiments to validate their effectiveness. As we saw, *DA* allows one to transfer labels from one dataset to another (unlabelled) dataset by aligning their feature distributions. But even though *DA* is an interesting theoretical problem its usefulness in real world applications is limited. The methods explored thus far only work under the restrictive assumption that $\mathcal{T}_T = \mathcal{T}_S$. In practice, we are usually interested in transferring labels from a large dataset (containing many classes) to a smaller (unlabelled) dataset with only a subset of these classes [24]. This fact motivates the more general framework of *PDA* introduced in [25] in which, the assumption $\mathcal{Y}_S = \mathcal{Y}_T$ is relaxed to $\mathcal{Y}_T \subseteq \mathcal{Y}_S$ and $Pr(Y_S = y|X, y \in \mathcal{Y}_T) = Pr(Y_T = y|X)$.

Ideally, we would like to identify a subset of the source data $X'_S (\subset X_S)$ such that $Pr(y \in \mathcal{Y}_T|X'_S) = 1$. If this is achieved we retrieve the full *DA* scenario and we simply need to align the marginal distributions $P(X'_S)$ and $P(X_T)$. However, it is important to note that we do not know which labels in $\mathcal{Y}_S$ correspond to $\mathcal{Y}_T$ and this makes our task much harder. As pointed out in [24], source data points with labels in $X_S \backslash X'_S$ are responsible for negative transfer and we want to eliminate (or decrease as much as possible) their contribution to training. To sum up, we can analyze the partial domain adaptation task into two objectives: minimize the influence of outlier source points[1] and maximally match the distributions between target domain and source domain with labels in $\mathcal{Y}_T$.

A naive approach to tackling the partial domain adaptation problem would be to augment the target data with a copy of the source data (ignoring labels). This certainly recovers the full *DA* problem but it does not achieve much. In general we are interested in cases where the source data is much more abundant than the target data and with many more classes. In these cases the contribution of target data under the naive approach may be minimal.

4.2 Adversarial Partial Domain Adaptation

Since we have already seen that using adversarial neural networks has been widely succesful for *DA* tasks it should come as no surprise that most methods for *PDA* are also built within this framework. Next, we will discuss some state-of-the-art methods for tackling the *PDA* problem within an adversarial framework.

[1] with labels in $X_S \backslash X'_s$

4.2.1 Selective Adversarial Network

The first method we will discuss is *SAN* which is introduced in [24] [2]. In this model, the domain discriminator used in typical adversarial *DA* models is replaced with $|\mathcal{Y}_S|$ discriminative networks D_i for $i = 1, \ldots, |\mathcal{Y}_S|$ (one for each source task). Informally, the idea behind this modification is that the network can now behave in different ways for different classes, which is desirable as we want maximal alignement for certain classes and no alignement for others. To decide which discriminator to use for each instance of training[3] a natural idea is to use the output of the source classifier **C** to make a soft decision. Recall that $\mathbf{C}\big(\mathbf{M}_s(x_s)\big)$ is a model for $P(Y|\mathbf{M}_s\big(X_S\big))$ and we can use it as a model for $P(Y|\mathbf{M}_t\big(X_T\big))$. Obviously, this is not necessarily a good model for the target task; this is the reason we are adapting domains in the first place.

Overall, for each instance $\mathbf{M}_t\big(x\big)$ or $\mathbf{M}_s\big(x\big)$ we use **C** to estimate the probability that x belongs to each of the source classes (using $\mathbf{C}\big(\mathbf{M}_t(x)\big)$ / $\mathbf{C}\big(\mathbf{M}_s(x)\big)$ respectively). Each domain discriminator D_i is associated with a cross entropy function

$$l_{D_i}(x) = -\frac{\mathbb{I}(d(x) = s)}{|X_s|} \log(D_i\big(\mathbf{M}_s(x)\big)) - \frac{\mathbb{I}(d(x) = t)}{|X_t|} \log(1 - D_i\big(\mathbf{M}_t(x)\big))$$

where $d(x)$ is the domain label of x and $\mathbb{I}$ is the indicator function. Note that

$$\sum_{x \in D_S \cup D_T} l_{D_i}(x) = \mathcal{L}_{D_i}$$

where $\mathcal{L}_{D_i}$ is the empirical estimate of the discriminator loss function that we discussed in classical adversarial *DA* and D_S, D_T are source and target datasets. We define the probability-weighted discriminator loss as

$$\mathcal{L}'_D = \sum_{k=1}^{|\mathcal{Y}_S|} \sum_{x \in D_S \cup D_T} \mathbb{I}(d(x) = i)\ \mathbf{C}^k\big(\mathbf{M}_i(x)\big)\ l_{D_k}(x)$$

where $\mathbf{C}^k$ denotes the k^{th} component of $\mathbf{C}(\cdot)$. In line with what we did for *ADDA*, we also define

$$\overset{M}{}_{D_i}(x) = -\frac{1}{|X_t|} \log(D_i\big(\mathbf{M}_t(x)\big))$$

and

$$\mathcal{L}'_M = \sum_{k=1}^{|\mathcal{Y}_S|} \sum_{x \in D_T} \mathbf{C}^k\big(\mathbf{M}_t(x)\big)\ l^M_{D_k}(x)$$

[2] There are some minor differences between what is described here and [24] because some adjustments were made for this project to be more coherent in terms of notation

[3] As discussed earlier we dont know target labels hence its not possible to determine this without some work

An additional weight term is included in the final model, which weighs down classes that are unlikely to be part of the target label space. In particular

$$w_k = \frac{1}{|X_T|} \sum_{x \in D_T} \mathbf{C}^k\big(\mathbf{M}_t(x)\big)$$

is used. It is evident that if most points score low for a particular class it is more unlikely that this class is part of the target label space. However, if a class has very few instances relative to other classes in the target data the model will likely fail to do effective transfer for this class since it will always get down-weighed (due to the low number of summands). This is not addressed in [24] but it is pointed out in [26]. The final loss functions for *SAN* are given by

$$\mathcal{L}_D^{SAN} = \sum_{k=1}^{|\mathcal{Y}_S|} w_k \sum_{x \in D_S \cup D_T} \mathbb{I}(d(x) = i)\ \mathbf{C}^k\big(\mathbf{M}_i(x)\big)\ l_{D_k}(x)$$

$$\mathcal{L}_M^{SAN} = \sum_{k=1}^{|\mathcal{Y}_S|} w_k \sum_{x \in D_T} \mathbf{C}^k\big(\mathbf{M}_t(x)\big)\ l_{D_k}^M(x)$$

and training takes the form

$$\forall k,\ \min_{D_k} \mathcal{L}_D^{SAN}|_{\text{fixed } \mathbf{M}_t, D_i\ \forall i \neq k}$$

$$\min_{\mathbf{M}_t} \mathcal{L}_M^{SAN}|_{\text{fixed } \mathbf{D_i},\ \forall i}$$

In general, because *SAN* uses one set of discriminator parameters for every class in the source task it does not scale well to problems with large source label spaces.

4.2.2 Importance Weighted Partial Domain Adaptation

Next, we discuss a different approach for partial domain adaptation introduced in [26]. The idea is similar to *SAN*; we want to reweight the influence of instances that are likely to be from outlier classes. However, instead of using the source classifier to do this, in this approach information about the relevance of an instance is derived from a domain discriminator. In particular, suppose that the domain discriminator $D(m)$ has reached its optimal value for some fixed representation networks $\mathbf{M}_S, \mathbf{M}_T$ (as defined in section $1.5.1$). Then, we can informally think of the output value of D as estimating the likelihood that D's input came from the source domain. Under this line of reasoning, if $D(\mathbf{M}_S(x)) \approx 1$ it is more probable that x's label is not part of the target label space and similarly if $D(\mathbf{M}_S(x)) \approx 0$ x's label is probably part of $\mathcal{Y}_T$. So instead of using $\mathbf{C}$ to calculate weights, in *IWAN* we use instance-based weights

$$\tilde{w}(m_x) = 1 - D^*(m_x)$$

derived from the optimal domain discriminator D^* (of section 1.5.1) for fixed representation networks. In the above, $m_x = \mathbf{M}_s(x)$. Since we are only interested in the relative importance of instances and to avoid having all weights near 1 or 0 the weights are normalized. Thus we get,

$$w(m_x) = \frac{\tilde{w}(m_x)}{E[\tilde{w}(m_x)]}$$

where the expectation is over the distribution of latent space instances occuring from the source domain. In practice the normalizing factor is estimated as the average of $\tilde{w}(m_x)$ for all x in D_S, or rather on the current minibatch of source data.

Given weights $w(m_x)$ for each instance x, a naive way to define the loss function for our task is

$$\min_{D} \max_{\mathbf{M}_t} \; - E_{x_s \sim X_S}[w(m_{x_s}) \log(D\big(\mathbf{M}_s(x_s)\big))] - E_{x_t \sim X_T}[\log(1 - D\big(\mathbf{M}_t(x_t)\big))] \tag{4.1}$$

But because we used D to derive the weights (ie the weights are a function of D) the optimal discriminator is not given by (as we might have hoped)

$$D^*(m_x) = \frac{w(m_x)p(m_x)}{w(m_x)p(m_x) + p(\mathbf{M}_t(x))}$$

Therefore, when solving the game (1.4), we dont get the theoretical guarantees of minimizing the *JSD* between the weighted source density and the target density. To fix this issue, a second domain discriminator D_0 (independent of D) is introduced. The game described in (1.4) becomes

$$\min_{D_0} \max_{\mathbf{M}_t} \; - E_{x_s \sim X_S}[w(m_{x_s}) \log(D_0\big(\mathbf{M}_s(x_s)\big))] - E_{x_t \sim X_T}[\log(1 - D_0\big(\mathbf{M}_t(x_t)\big))] \tag{4.2}$$

where now the weights are independent of D_0 and thus the optimal domain discriminator is given by

$$D_0^*(m_x) = \frac{w(m_x)p(m_x)}{w(m_x)p(m_x) + p(\mathbf{M}_t(x))}$$

Note also that by definition $E[w(m_x)] = \int \; w(m_x)p(m_x) \; dm_x = 1$ and clearly $w(m_x) \geq 0$ implying that $w(m_x)p(m_x)$ is a valid probability density function. Hence from the analysis of section $(1.6.1)$ solving (1.4) minimizes the *JSD* between the weighted source probability density and the target density. This is precisely what we want given that the weights actually do weight down the outlier classes and promote the target classes. To finish describing this method we also need to specify the loss functions for the classifier and the first discriminator.

Another important insight given in [26] is that the entropy minimization principle described in [23] can be used to make sure that the target classes are well separated by **C** under a transformation through $\mathbf{M}_t$. As stated in [23] and as we already discussed for *WADDA*, the information content of unlabeled data is reduced as classes overlap increases and it

is reasonable to assume that classes are well separated if unlabeled data are going to be useful to us. In particular, the loss function is augmented as

$$\mathcal{L} = -\lambda E_{x\sim X_t}[H\Big(\mathbf{C}\big(\mathbf{M}_t(x)\big)\Big)] + \mathcal{L}'$$

and again we only maximize with respect to $\mathbf{M}_t$ and minimize with respect to D. In the above

$$\mathcal{L}' = -E_{x_s\sim X_S}[w(m_{x_s})\,\mathsf{log}(D_0\big(\mathbf{M}_s(x_s)\big))] - E_{x_t\sim X_T}[\mathsf{log}(1 - D_0\big(\mathbf{M}_t(x_t)\big))]$$

and $\lambda > 0$ is a trade-off parameter as before. Empirically, we found that good values for λ for MNIST/USPS are between 3 and 7. Finally, H is the information entropy function. Informally, we can think of minimizing $E_{x\sim X_t}[H\Big(\mathbf{C}\big(\mathbf{M}_t(x)\big)\Big)]$ as pushing $\mathbf{C}\big(\mathbf{M}_t(x)\big)$ as far away from the uniform distribution as possible. The full optimization problem is given by the following equations:

$$\min_{D_0}\max_{\mathbf{M}_t}\mathcal{L}$$

$$\min_{D}\ -E_{x_s\sim X_S}[\mathsf{log}(D\big(\mathbf{M}_s(x_s)\big))] - E_{x_t\sim X_T}[\mathsf{log}(1 - D\big(\mathbf{M}_t(x_t)\big))]$$

$$\min_{C,\mathbf{M}_t} - E_{x_s,y_s\sim X_S,Y_S}\big[\sum_{i=1}^{K}\mathbb{I}\{i = y_s\}\ \mathsf{log}\,C_i\big(\mathbf{M}_s(x_s)\big)\big]$$

TWO-WAY PARTIAL DOMAIN ADAPTATION

5.1 Problem Description & Motivation

We discussed how reweighting the source distribution during training can minimize negative transfer in partial domain adaptation. But still, a natural question arises: Are the assumptions in partial domain adaptation general enough? We will argue that they are not.

Here is a quick recap. The upshot of developing transfer learning schemes is that we want to be able to train models on large labeled datasets (presumably hard to collect) with many classes and use them to train models for easy to obtain unlabelled datasets. We saw the theory behind domain adaptation and how this can be applied when the target and source labels are the same. In [24] and [26] a big leap forward was made for practical applications with the introduction of partial domain adaptation. This addresses the much more realistic scenario where the target label space is only a subset of the source label space. But in the general label transfer problem an even more realistic assumption is $\mathcal{Y}_S \cap \mathcal{Y}_T \neq \emptyset$, which we introduce here and term *two-way partial domain adaptation*. That is, to simply assume that there are some shared labels between tasks.

Two-way partial domain adaptation captures those situations where in addition to the assumptions of *PDA*, some new unseen classes are part of the target dataset, which were not present in the source data. Such scenarios arise naturally when the underlying task involves many classes and the source label space is small (for example due to cost of collecting a larger labelled dataset). Clearly, for a target instance with label $y \in \mathcal{Y}_T \backslash \mathcal{Y}_S$, we cannot hope to say much since our source classifier only outputs a distribution over source labels. However, the existence of such instances is a source of negative transfer during training. During training the partial domain adaptation model will falsely try to align the entire target distribution to the source distribution. To make matters worse, if we are using *IWPDA*, because source instances get reweighted, target outlier instances will tend to be aligned to source instances with labels in $\mathcal{Y}_T$. This hinders training. In more detail, the underlying assumption is further relaxed from $\mathcal{Y}_T \subseteq \mathcal{Y}_S$ to $\mathcal{Y}_T \cap \mathcal{Y}_S \neq \emptyset$ and $P(Y_S = y|X, y \in \mathcal{Y}_T \cap \mathcal{Y}_S) = P(Y_T = y|X, y \in \mathcal{Y}_T \cap \mathcal{Y}_S)$. The non-empty intersection is needed so that there is at least some information to be gained from the source model. According to our discussion thus far it should be clear that our goal is to find appropriate parameters so that $P(\mathbf{M}_S(X_S^{\cap})) = P(\mathbf{M}_T(X_T^{\cap}))$ where $X_i^{\cap}$ denotes the subset of X_i with labels in $\mathcal{Y}_T \cap \mathcal{Y}_S$. We will call the set $X_T^{\cap}$ the *transfer-relevant* instances.

We are also interested in developing a method which will allow us to identify which target instances are transfer-relevant. This is a very important component of the solution to the two-way partial domain adaptation problem, since without it, our obtained model for the

target task is essentialy unusable. Of course identifying $X_T^{\cap}$ without the target labels is extremely hard (if not impossible!) to do and hence we will settle for a solution that provides instances that are *probably transfer-relevant* with respect to some confidence measure that is related to $Pr(x \in X_T^{\cap})$.

In the following sections we propose a solution that both minimizes negative transfer due to outlier target labels and provides (essentially for free) a way to identify probably transfer-relevant instances. We will then provide experimental evidence for the effectiveness of our proposed method.

5.2 Doubly Importance Weighted Adversarial Network

We will introduce an algorithm inspired by the reweighting done in *IWPDA* for tackling two-way partial domain adaptation. Like in *IWPDA* we use two domain classifiers D and D_0 to align weighted latent distributions but unlike *IWPDA* we use the first classifier to assign weights to target instances as well. The idea behind this is similar to what we saw earlier for *IWPDA*. Denote $\mathbf{M}_T(x_t)$ as m_x^t for simplicity. Recall that for a fixed target representer network the optimal domain discriminator is given by (1.5), which may be rewritten as

$$D^*(x) = \frac{1}{1 + \frac{p_T(x)}{p_s(x)}} \tag{5.1}$$

which scales like $\frac{p_S(x)}{p_T(x)}$. When the domain discriminator is confident that a target instance lies in the target domain, $ie\ D(m_x^t) \approx 0$, it is unlikely that this instance has a label common to the source task and so it should be down weighted. Similarly, if $D(m_x^t) \approx 1$ it is much more likely that the label of this instance is shared in the source label space and so we should promote transfer. We have accordngly,

$$\tilde{w}^T(m_x^t) = D(m_x^t)$$

and normalizing as before

$$w^T(m_x^t) = \frac{\tilde{w}^T(m_x^t)}{E[\tilde{w}^T(m_x^t)]}$$

For the source domain nothing changes so we have,

$$\tilde{w}^S(m_x) = 1 - D^*(m_x)$$
$$w^S(m_x) = \frac{\tilde{w}^S(m_x)}{E[\tilde{w}^S(m_x)]}$$

Now let $P^{w^T}(\mathbf{M}_T(x_t)) = w^T(m_x^t)p_T(m_x^t)$, where p_T is the pdf of the target latent space. It is evident that P^{w^T} defines a probability density function which is the desired weighted density for the target task. Similarly, $P^{w^S} = w^S(m_x)p_S(m_x)$ forms the weighted density for the source task. The idea now is to use a separate domain discriminator to align these

weighted densities. The objective function looks like,

$$\mathcal{L}' = -E_{x_s \sim X_S}[w^S(m_{x_s}) \log(D_0(\mathbf{M}_s(x_s)))] - E_{x_t \sim X_T}[w^T(m_x^t) \log(1 - D_0(m_x^t))]$$

and as before, we can optimize with respect to D_0 for fixed weights and representer networks, using the calculus of variations. Denote the latent space Z, we have

$$\begin{aligned}
\mathcal{L}' &= \int_X F[D(x), D'(x), x]\, dx \\
&- \int_X [w^S(m_x) p_S(x) log(D_0(m_x)) + w^T(m_x^t) p_T(x) log(1 - D_0(m_x^t))]\, dx \\
&\underset{(a)}{=} - \int_Z [w^S(z) p_S(z) log(D_0(z)) + w^T(z) p_T(z) log(1 - D_0(z))]\, dz
\end{aligned}$$

where (a) follows from the change of variable formula and density transformation. We solve for D the equation

$$\begin{aligned}
\frac{\partial \mathcal{L}'}{\partial D}[D] = \frac{\partial F}{\partial D} - \frac{d}{dx}\frac{\partial F}{\partial D'} = 0 \Rightarrow \\
\frac{\partial}{\partial D}[w^S(z) p_S(z) log(D_0^*(z))] = -\frac{\partial}{\partial D}[w^T(z) p_T(z) log(1 - D_0^*(z))] \Rightarrow \\
\frac{w^S(z) p_S(z)}{D_0^*(z)} = \frac{w^T(z) p_T(z)}{1 - D_0^*(z)} \\
D_0^*(z) = \frac{w^s(z) p_S(z)}{w^s(z) p_S(z) + w^T(z) p_T(z)}
\end{aligned}$$

and in turn fixing the weights and the current optimal discriminator, we have

$$\begin{aligned}
\max_D \; -\mathcal{L}' = \\
= E_{x_s \sim X_S}[w^S(m_{x_s}) \log(\frac{w^s(m_x) p_S(m_x)}{w^s(m_x) p_S(m_x) + w^T(m_x) p_T(m_x)})] \\
+ E_{x_t \sim X_T}[w^T(m_x^t) \log(\frac{w^T(m_x^t) p_T(m_x^t)}{w^s(m_x^t) p_S(m_x^t) + w^T(m_x^t) p_T(m_x^t)})] = \\
= E_{z \sim P^{w^S}}[\log(\frac{P^{w^S}(z)}{P^{w^S}(z) + P^{w^T}(z)})] + E_{z \sim P^{w^T}}[\log(\frac{P^{w^T}(z)}{P^{w^S}(z) + P^{w^T}(z)})] = \\
= -\log(4) + JS(P^{w^S} || P^{w^T})
\end{aligned}$$

and it is promptly seen that this has a global minimum at $P^{w^T} = P^{w^S}$. Thus, as before, we have a way of calculating the gradient of the *JSD* between our distributions of interest using alternating optimization.

A final point to make is that we can use the scoring discriminator, which is obtained for free after runing *DIWAN*, to obtain a measure of the likelihood that each target instance belongs to an outlier target label. If the algorithm converges succesfully the distributions

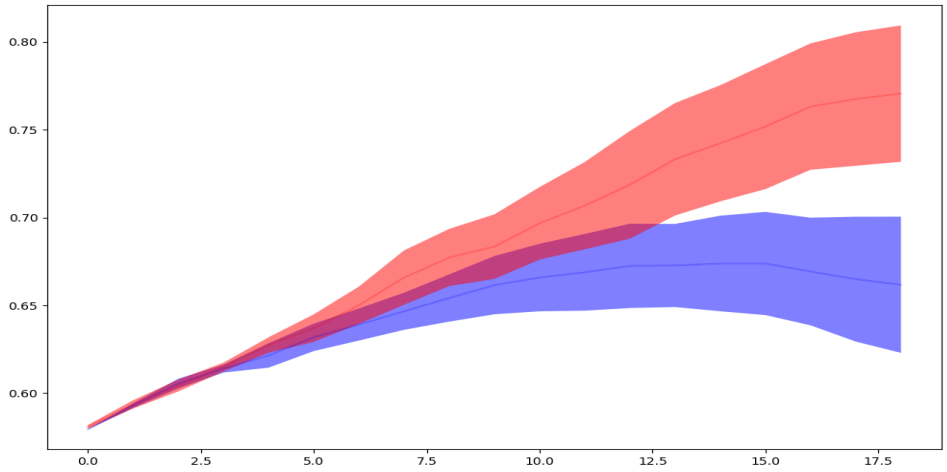

Figure 5.1: Experimental evidence for negative transfer due to oulier classes in target domain for obstruction task ($n = 15$). In red 0 outlier labels and in blue 5 outlier labels. We plot two standard deviation intervals from mean for 25 simulations per case.

of weighted source and weighted target data in the latent space are aligned. This means that the scoring discriminator (with appropriate capacity) when trained to optimality will give low scores for data that were involved heavily in the transfer and high scores for data that were not. Hence, the target weights described above are a natural confidence measure that allows us to identify transfer-relevant instances.

5.3 Experimental Evidence for Negative Transfer in the two-way Setting

We start by running experiments to observe how outlier target labels affect the effectiveness of *ADDA*. Our experimental setup is identical to that in Chapter 3. The same four tasks are considered. We start by running *ADDA* 25 times on a full domain adaptation scenario with 5 source labels and for 20 epochs. We then repeat for a scenario where there are 10 target labels and 5 source labels. The source network, labels and all hyperparameters are kept the same in both scenaria. We perform this experiment for all 4 tasks and plot average accuracy $\pm 2\sigma$ vs #-iterations for all tasks.

The results are as expected. The outlier target labels hinder the adaptation procedure notably. It is interesting that the negative effect is more severe for some tasks. For example, in the obstruction task the accuracy loss is on average $10\% <$, while for the rotation task it is close to 1%. For the rescaling task the accuracy loss is on average 2.5%, while on the displacement task it is 4.3%.

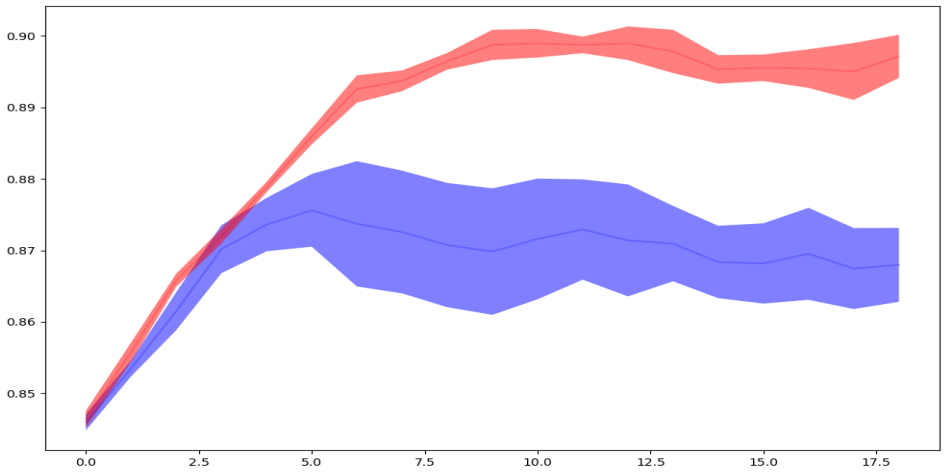

Figure 5.2: Experimental evidence for negative transfer due to oulier classes in target domain for rescaling task $(1.3, 0.7)$. In red 0 outlier labels and in blue 5 outlier labels. We plot two standard deviation intervals from mean for 25 simulations per case.

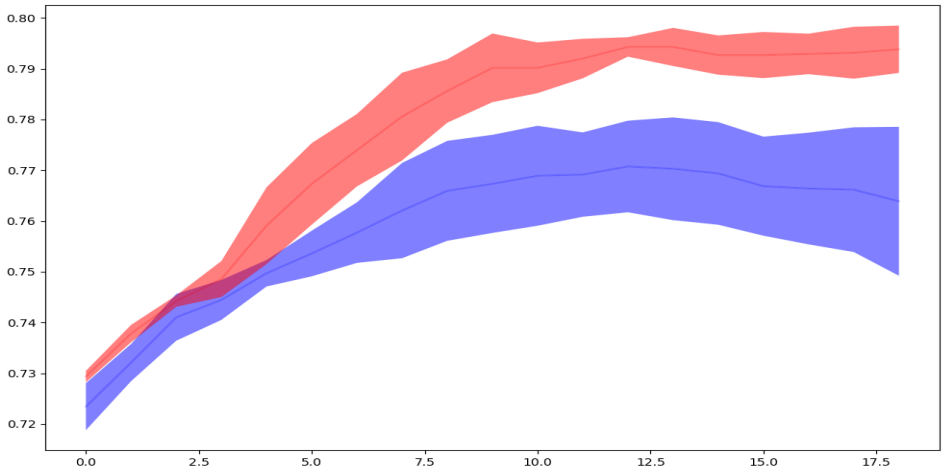

Figure 5.3: Experimental evidence for negative transfer due to oulier classes in target domain for displacement task $((5, 3)$. In red 0 outlier labels and in blue 5 outlier labels. We plot two standard deviation intervals from mean for 25 simulations per case.

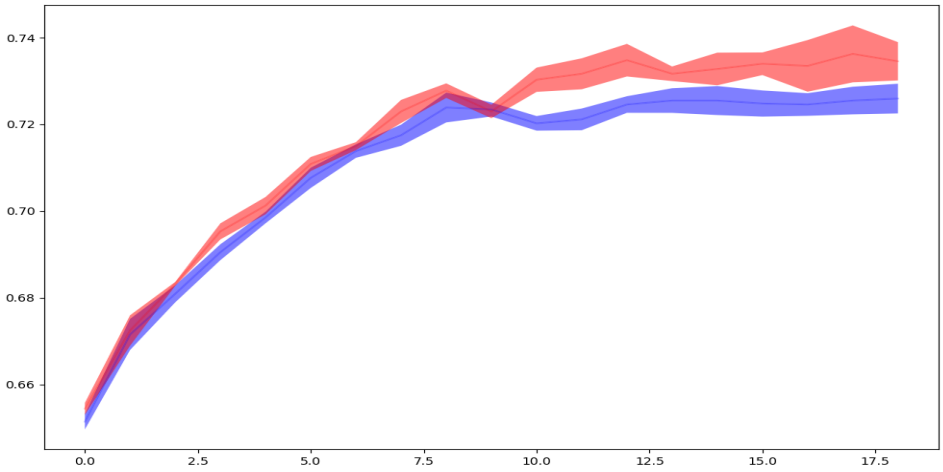

Figure 5.4: Experimental evidence for negative transfer due to oulier classes in target domain for rotation task $\theta = -30$. In red 0 outlier labels and in blue 5 outlier labels. We plot two standard deviation intervals from mean for 25 simulations per case.

5.4 Experimental Evidence for the effectiveness of DIWAN

In this section we break down *DIWAN* into the component mitigating negative transfer and the component identifying the transfer-relevant instances. Experiments for each of these are presented separately in the following two subsections.

5.4.1 Mitigating Negative Transfer

Below we present experimental results that illustrate the benefits of our method. We trained a source network and classifier on MNIST using only five random digit labels. We then perform transfer with different number of target labels and common labels. We compare *IWPDA* from Chapter 4 with our *DIWAN*. To identify negative transfer we only measure accuracy on common test labels. Note that in practice we do not know which instances correspond to the common test labels and hence the experiment below is "unrealistic". That is, the recorded accuracies could not be measured/achieved in a real world problem without further work. However, by comparing test accuracies on the common label instances we provide evidence for the negative transfer occuring when we ignore outlier target labels. We will repeat the experiment for all 4 tasks of interest. Each experiment is performed 5 times and average accuracies are reported.

All experiments in this section are performed on an NVIDIA GeForce 840M, except when marked with a $(*)$ in which case they are ran on a Tesla K40c. On the Tesla a different

source network and source labels are used as well.

In general, it is clear from the results that *DIWAN* mitigates the negative transfer due to outlier source and target labels in all 4 tasks. Furthermore, we found that *DIWAN* learns faster compared to *ADDA* and *IWPDA* and allows for usage of greater learning rates. We empirically found that in order to achieve "good" results (as the ones listed in our experiments), *ADDA* needs $5\times$-$10\times$ smaller learning rates. Appropriate tuning needs to be made for the discriminator as well and even with such low learning rates the algorithm often converges to much lower accuracy levels than *DIWAN*. When *ADDA* was run with larger learning rates there was rapid divergence. Finally, we observe that while *DIWAN* handles any combination of source and target outlier instances, when target outlier labels are present together with source outlier labels, *IWPDA* is unable to impove transfer quality over *ADDA*. This is characteristic in the rotation and displacement task experiments.

Table 5.1: Results for Obstruction ($n = 15$)

# target labels	# common labels	ADDA	IWPDA	2-way IWPDA	source network
2	2	3.77%	**89.56 %**	86.38%	7.83%
3	3	90.11%	**93.27%**	91.43%	34.85%
4	4	80.09%	**82.97%**	78.56%	53.19%
4	2	57.36%	85.59%	**87.76%**	57.35%
5	2	3.45%	70.72%	**88.44%**	7.83%
6	2	60.89%	78.49%	**80.05%**	49.16%
4	3	33.59%	**63.88%**	**64.48%**	29.32%
5	3	38.90%	**74.91%**	**74.84 %**	41.47%
6	3	67.65%	**69.54%**	**70.87%**	61.62%
7	3	70.22%	74.56%	**89.95%**	68.64%
5	4	75.21%	77.30%	**85.61%**	53.19%
6	4	74.41%	77.62%	**83.86%**	48.49%
7	4	79.91%	77.54%	**81.76%**	69.71%
8	5	72.69%	73.20%	**77.98%**	57.03%
10	5	63.09%	67.31%	**76.22%**	57.03%

Compare full domain adaptation with partial domain adaptation and two-way partial domain adaptation on different number of target labels on the obstruction task. 20 iterations, $(5scor, 10dd) : 1M_T$ iterations.

Table 5.2: Results for Rotation ($\theta = 40$)

# target labels	# common labels	ADDA	IWPDA	2-way IWPDA	source network
2	2	64.04%	**67.89%**	67.54%	64.94%
3	3	**66.90%**	66.78%	64.76%	62.15%
4	4	73.26%	**74.89%**	73.91%	68.82%
4	2	**70.01%**	65.14%	68.86%	64.95%
5	2	73.99%	73.81%	**74.49%**	72.77%
6	2	66.05%	60.13%	**67.16%**	49.28%
4	3	76.30%	76.98%	**77.83%**	73.59%
5	3	**71.23%**	68.40%	**71.67%**	62.15%
6	3	84.85%	83.24%	**84.92%**	73.30%
7	3	75.36%	77.21%	**81.50%**	62.83%
5	4	70.21%	70.43%	70.76%	69.08%
6	4	70.07%	70.04%	**70.26%**	69.08%
7	4	75.57%	77.03%	**78.86%**	68.82%
8	5	63.37%	64.21%	**67.39%**	65.83%
10	5	61.31%	65.83%	**67.02%**	65.83%

Compare full domain adaptation with partial domain adaptation and two-way partial domain adaptation on different number of target labels on the rotation task. 20 iterations, $(1scor, 10dd) : 1M_T$ iterations.

Table 5.3: Results for Displacement ($(i, j) = 5, 3$)

# target labels	# common labels	ADDA	IWPDA	2-way IWPDA	source network
2	2	60.00%	**78.89%**	**78.83%**	72.22%
3	3	59.77%%	**85.66%**	84.58%	80.65%
4	4	74.02%	**85.32%**	**85.05%**	66.07%
4	2	60.22%	60.01%	**65.21%**	56.42%
5	2	55.33%	60.07%	**78.88%**	72.22%
6	2	78.97%	79.24%	**90.46%**	59.23%
4	3	83.55%	88.31%	**92.66%**	69.90%
5	3	34.62%	35.17%	**55.76%**	35.31%*
6	3	38.98%	41.36%	**59.83%**	38.99%*
7	3	34.00%	36.13%	**42.62%**	35.31%*
5	4	79.21%	81.36%	**82.04%**	74.44%
6	4	45.50%	53.31%	**61.23%**	44.42%*
7	4	39.69%	41.55%	**66.80%**	38.31%*
8	5	73.25%	73.67%	**85.03%**	70.74%
10	5	66.56%	73.10%	**82.32%**	70.74%

Compare full domain adaptation with partial domain adaptation and two-way partial domain adaptation on different number of target labels on the displacement task. 20 iterations, $(1scor, 10dd) : 1M_T$ iterations.

Table 5.4: Results for Rescaling ($(i, j) = 0.6, 0.6$)

# target labels	# common labels	ADDA	IWPDA	2-way IWPDA	source network
2	2	60.32%	**87.60%**	85.92%	83.80%
3	3	77.28%	**81.32%**	80.97%	60.56%
4	4	77.33%	**79.41%**	77.90%	62.50%
4	2	55.84%	67.51%	**82.65%**	69.09%
5	2	80.98%	74.48%	**85.30%**	65.13%
6	2	72.31%	83.71%	**94.56%**	83.39%
4	3	66.02%	85.21%	**90.01%**	79.22%
5	3	59.03%	87.88%	**93.78%**	79.22%
6	3	44.67%	46.98%	**53.25%**	46.47%*
7	3	49.14%	56.39%	**58.12%**	54.11%*
5	4	52.98%	54.98%	56.76%	55.17%*
6	4	80.57%	81.92%	**87.61%**	80.87%*
7	4	70.07%	76.17%	**82.98%**	74.61%
8	5	59.65%	54.78%	**62.54%**	57.48%*
10	5	74.17%	77.83%	**84.54%**	68.57%

Compare full domain adaptation with partial domain adaptation and two-way partial domain adaptation on different number of target labels on the rescaling task. 20 iterations, $(1scor, 10dd) : 1M_T$ iterations.

5.4.2 DIWAN for Identifying Probably Transfer-Relevant Instances

In this experiment we investigate how the scoring discriminator weights vary between target instances. We start by randomly selecting the number of target labels and common labels we will use for each task. We then train our algorithm and plot histograms of the weights. Histograms for oulier labels are plotted with blue, while transfer relevant labels are plotted with red. We then proceed to measure the accuracy of our model on filtered instances with *DIWAN* score above different thresholds. We measure the total number of target instances after applying the filter, the percentage of total target relevant instances that go through the filter and the percentage of transfer relevant instances in the filtered dataset.

In general it is seen that the weight distribution for transfer relevant features for all four tasks is right-skewed and most of the transfer relevant instances have scores higher than

the mean (which corresponds to 1 on all diagrams). Furthermore, we find that for all tasks there is a cut-off threshold for which the filtered data is mostly transfer relevant (with percentage above 95%). For the obstruction task 32.39% of the dataset passed the threshold, while for other tasks the corresponding percentage is as low as 10%. We note however that apart from the rotation task, those high scoring transfer relevant instances are more likely to be classified correctly compared to lower scoring transfer relevant instances. For example, the accuracy on all the transfer relevant instances for the obstruction task was 78.91%, while for the high-scoring filtered data (99.22% of which is transfer relevant) the accuracy was 98.43%. For the rotation task, this is not true and this requires further investigation.

An important point to make here is that we have not given a principled way of selecting the threshold; we simply provided some empirical evidence to support that there is a "good" threshold. In practice, the choice of threshold needs to be chosen using a validation procedure. We still note that for all our experiments, keeping only instances with scores $1.5\times$mean results in a filtered dataset that mostly consists of transfer relevant data.

Table 5.5: Result for Obstruction experiment ($n = 14$) (Accuracy on transfer relevant instances, 56.08% $\rightarrow$ 78.91%)

Cut-off threshold	% of total transfer relevant instances considered	% transfer relevant instances in considered data	**# of target instances considered**	**Accuracy**
0	100.00	47.10	4142	36.12%
0.5	100.00	62.53	3120	47.95%
1	97.54	80.13	2375	62.95%
1.5	32.39	99.22	637	98.43%

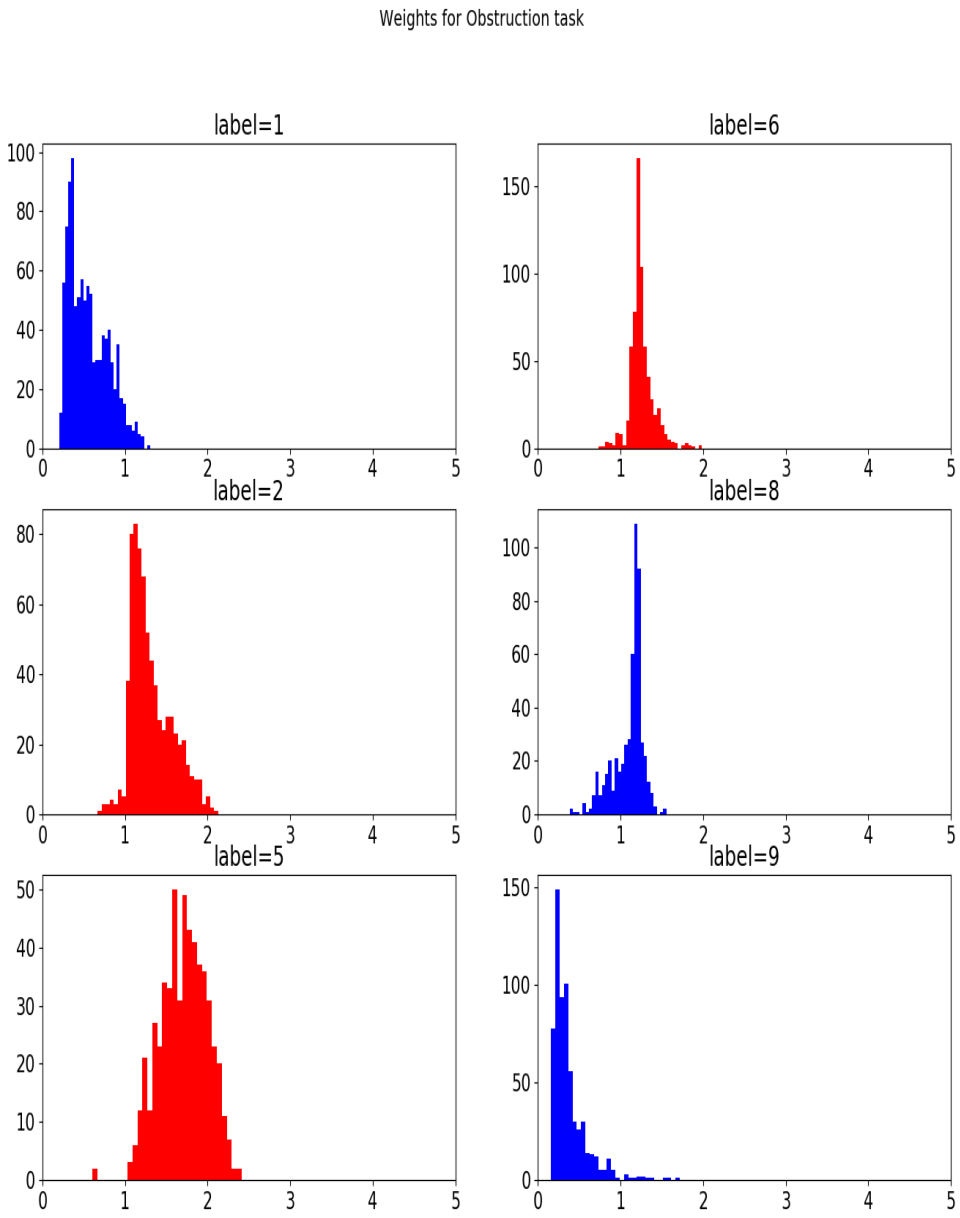

Figure 5.5: Weight histogram for obstruction task ($n = 14$). In red 5 transfer relevant labels and in blue 5 outlier labels.

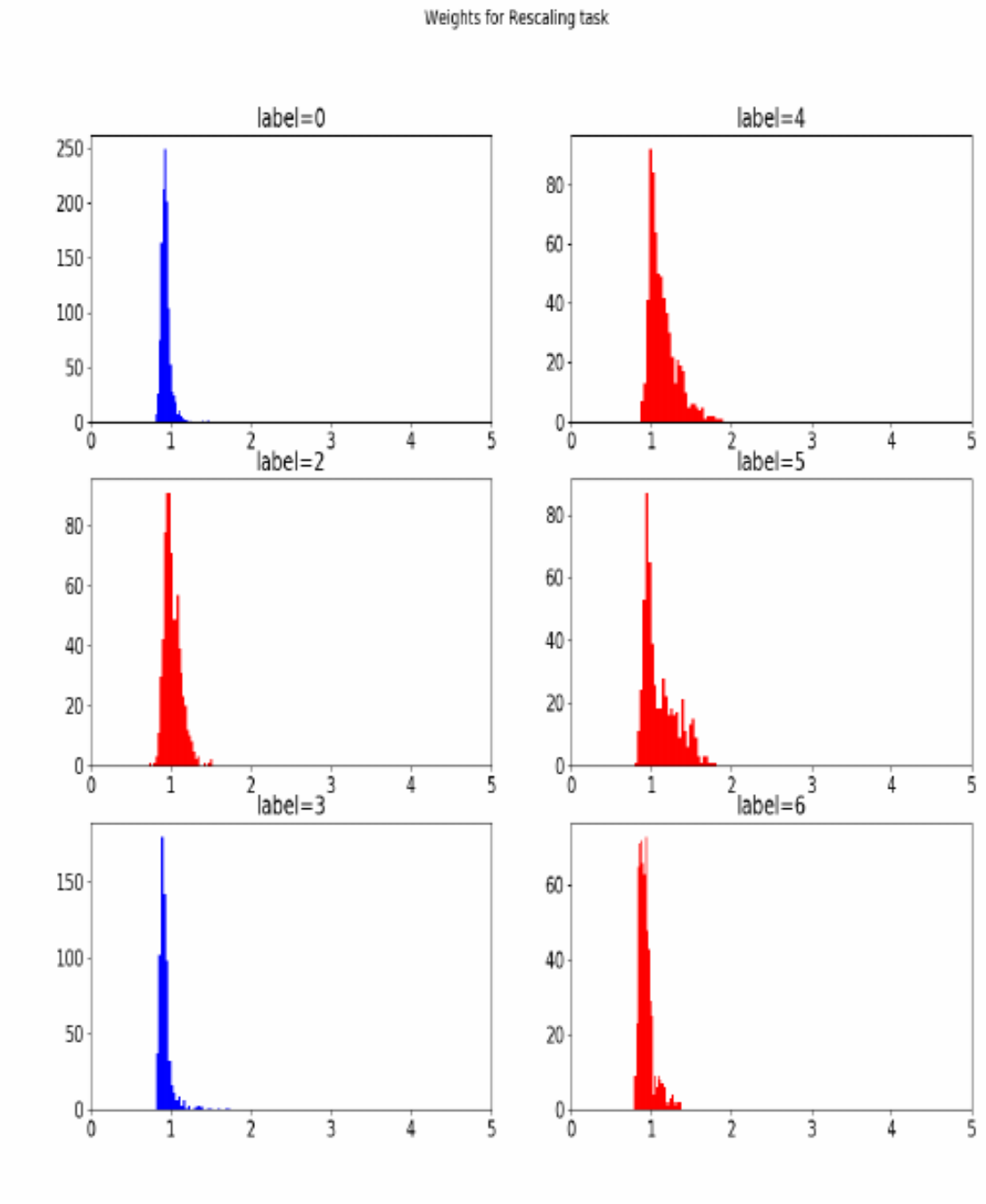

Figure 5.6: Weight histogram for rescaling task $(0.6, 0.6)$. In red 2 transfer relevant labels and in blue 4 outlier labels.

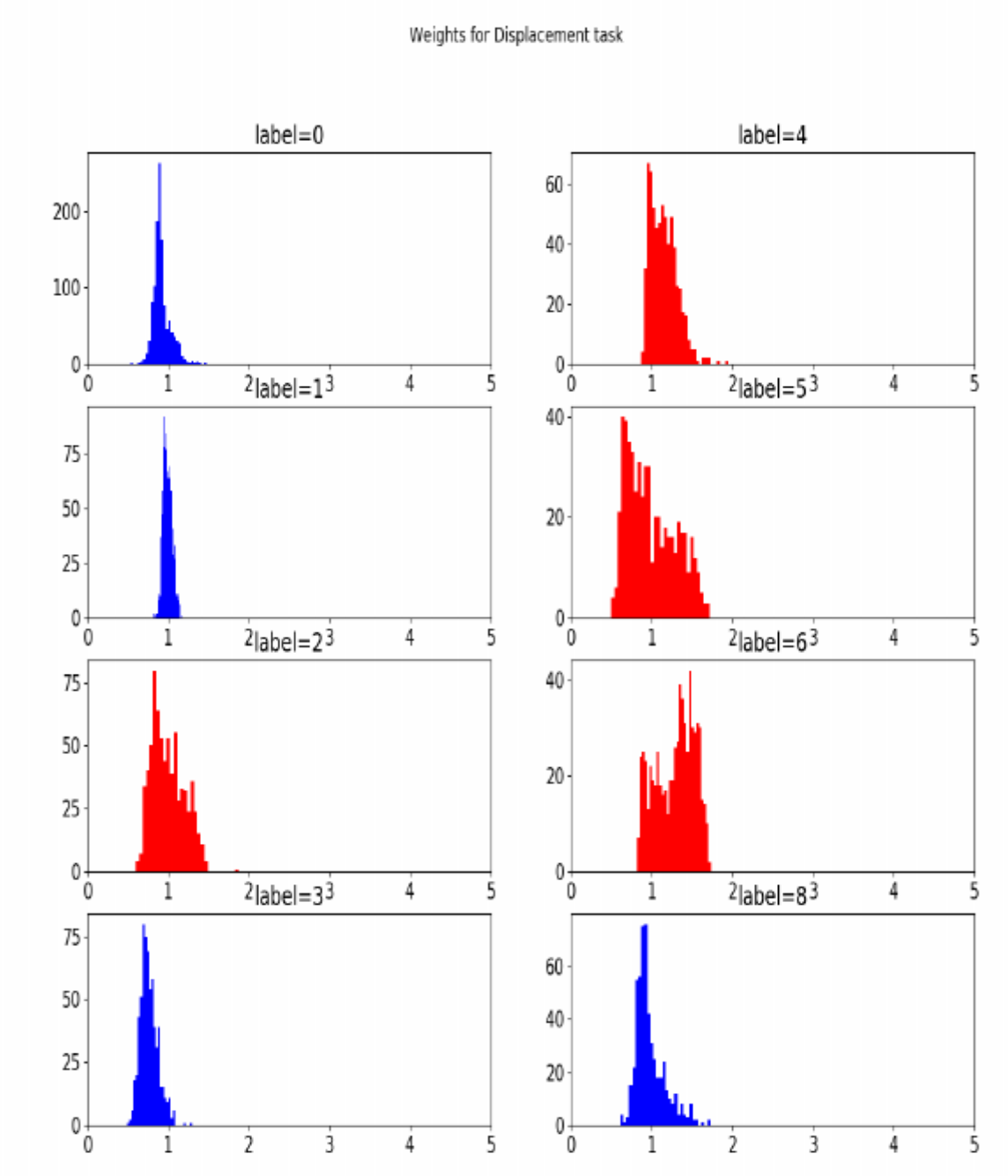

Figure 5.7: Weight histogram for displacement task $(5, 3)$. In red 4 transfer relevant labels and in blue 4 outlier labels.

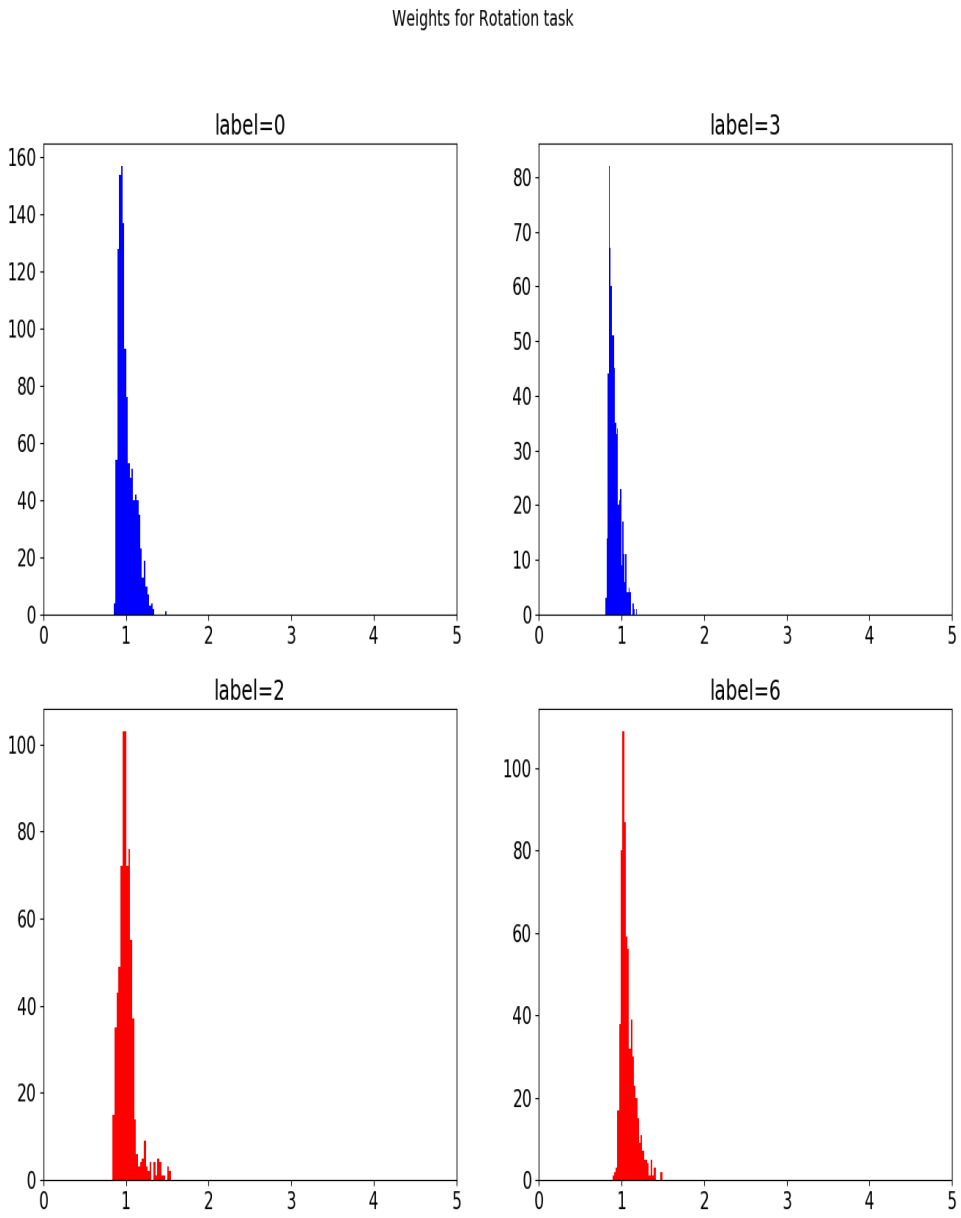

Figure 5.8: Weight histogram for displacement task ($\theta = -30$). In red 2 transfer relevant labels and in blue 4 outlier labels.

Table 5.6: Result for Rescaling experiment $(0.6, 0.6)$ (Accuracy on transfer relevant instances, 62.49% $\rightarrow$ 73.63%)

Cut-off threshold	% of total transfer relevant instances considered	% transfer relevant instances in considered data	**# of target instances considered**	**Accuracy**
0	100.00	58.42	4455	37.15%
1	53.51	78.39	1777	67.36%
1.25	32.85	98.16	871	92.66%
1.5	20.86	100.00	543	100.00%

Table 5.7: Result for Displacement experiment $(5, 3)$ (Accuracy on transfer relevant instances, 65.66% $\rightarrow$ 76.92%)

Cut-off threshold	% of total transfer relevant instances considered	% transfer relevant instances in considered data	**# of target instances considered**	**Accuracy**
0	100.00	43.37	6002	49.92%
1	61.93	65.90	2446	60.95%
1.25	30.43	92.63	855	84.21%
1.5	7.99	96.74	216	91.16%

Table 5.8: Result for Rotation experiment $\theta = -30$ (Accuracy on transfer relevant instances, 64.95% $\rightarrow$ 68.43%)

Cut-off threshold	% of total transfer relevant instances considered	% transfer relevant instances in considered data	**# of target instances considered**	**Accuracy**
0	100.00	42.96	3247	42.22%
1	55.84	59.19	1316	38.98%
1.25	21.57	86.24	349	50.43%
1.5	5.96	98.96	97	35.16%

CONCLUSIONS & FUTURE WORK

In this project we have motivated the use of domain adaptation schemes for solving the label transfer problem and we investigated possible improvements on classical adversarial domain adaptation methods. In particular, we proposed an adversarial neural network algorithm (*WADDA*) that relies on the Wasserstein distance as a distribution alignment metric, which achieves much better results on most transfer tasks we considered. The Wasserstein distance was also motivated theoretically, extending previous work on *GANs*. It was observed that *WADDA* with weight clipping or weight projection to a sphere may become unstable due to exploding gradients and in practice we found that using gradient regularization is a good practice to avoid this. We empirically found that the effectiveness of *WADDA* relative to other *JSD* based methods is task dependent.

Furthermore, we introduced two-way partial domain adaptation and proposed an algorithm to tackle it. We illustrated experimentally that outlier target labels are a source of negative transfer in domain adaptation and we provided a way to isolate probably transfer relevant instances. We showed that in the two-way setting our algorithm mitigates negative transfer substantially for our considered tasks and allows us to train our models faster, using larger learning rates. In addition, we found that in the *PDA* setting our *DIWAN* algorithm is competitive with *IWPDA*.

In the future, our methods will be tested on larger more complex datasets, possibly consisting of natural images. In addition, we plan to investigate domain adaptation on sequential data and recurrent neural networks. Furthermore, we plan to leverage the Wasserstein cost function for developing solution for the partial and two-way partial setting. This was not done in the present work because of time constraints. Finally, alternative ways of identifying transfer relevant instances in the two-way partial domain adaptation setting will be investigated.

ABBREVIATIONS - ACRONYMS

ADDA	Adversarial Discriminative Domain Adaptation
ANN	Artificial Neural Network
CADA	Conditional Adversarial Domain Adaptation
DA	Domain Adaptation
DIWAN	Doubly Importance Weighted Adversarial Network
EM	Earth-Mover
GAN	Generative Adversarial Networks
ITL	Inductive Transfer Learning
IWAN	Importance Weighted Adversarial Network
IWPDA	Importance Weighted Partial Domain Adaptation
JSD	Jenshen Shannon Divergence
KL	Kullback-Liebler
ML	Machine Learning
MMD	Maximum Mean Discrepancy
MMDE	Maximum Mean Discrepancy Embedding
MSE	Mean Squared Error
MTL	Multi-Task Learning
PAC	Probably Approximately Correct
PCA	Principal Component Analysis
PDA	Partial Domain Adaptation
RKHS	Reproducing Kernel Hilbert Space
SAN	Selective Adversarial Network
SDP	Semi-Definite Programming
TL	Transfer Learning
TTL	Transductive Transfer Learning
TV	Total Variation
VC	Vapnik-Chervonenkis
WADDA	Wasserstein Adversarial Discriminative Domain Adaptation
WDGRL	Wasserstein Distance Guided Representation Learning
WGAN	Wasserstein Generative Adversarial Network

BIBLIOGRAPHY

[1] Sinno Jialin Pan and Qiang Yang. *A Survey on Transfer Learning*. Fellow, IEEE.

[2] Gabriela Csurka. *Domain Adaptation for Visual Applications: A Comprehensive Survey*. arXiv:1702.05374v2 [cs.CV] 30 May 2017.

[3] H. Shimodaira. *Improving predictive inference under covariate shift by weighting the log-likelihood function*. Journal of Statistical Planning and Inference, vol. 90, no. 2, pp. 227–244, 2000.

[4] Arthur Gretton, Karsten M. Borgwardt, Malte Rasch, Bernhard Scholkopf and Alexander J. Smola. *A Kernel Approach to Comparing Distributions*.

[5] Sinno Jialin Pan, James T. Kwok and Qiang Yang. *Transfer Learning via Dimensionality Reduction*. Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (2008).

[6] K. Q. Weinberger, F. Sha and L. K. Saul. *Learning a kernel matrix for nonlinear dimensionality reduction*. In Proceedings of the 21st International Conference on Machine Learning, 2004.

[7] Charles A. Micchelli, Yuesheng Xu and Haizhang Zhang. *Universal Kernels*. Journal of Machine Learning Research 7 (2006) 2651-2667.

[8] Kartik Krishnan Sivaramakrishnan, John E. Mitchell. *Properties of a Cutting Plane Method for Semidefinite Programming*. 2007.

[9] Sinno Jialin Pan, Ivor W. Tsang, James T. Kwok and Qiang Yang. *Domain Adaptation via Transfer Component Analysis*. IEEE Transactions on Neural Networks and Learning Systems, 2011.

[10] Jason Yosinski, Jeff Clune, Yoshua Bengio and Hod Lipson. *How transferable are features in deep neural networks?*. arXiv:1411.1792v1 [cs.LG], 6 Nov 2014.

[11] Pengfei Liu, Xipeng Qiu, Xuanjing Huang. *Adversarial Multi-task Learning for Text Classification*. arXiv:1704.05742v1 [cs.CL], 19 Apr 2017.

[12] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. *Generative Adversarial Nets*. arXiv:1406.2661 [stat.ML]

[13] Mingsheng Long, Zhangjie Cao, Jianmin Wang, Michael I. Jordan. *Conditional Adversarial Domain Adaptation*. arXiv:1705.10667 [cs.LG], 10 Feb 2018.

[14] Eric Tzeng, Judy Hoffman, Kate Saenko, Trevor Darrell. *Adversarial Discriminative Domain Adaptation*. arXiv:1702.05464 [cs.CV], 17 Feb 2017.

[15] Martin Arjovsky, Soumith Chintala and Léon Bottou. *Wasserstein GAN*. arXiv:1701.07875v3 [stat.ML], 6 Dec 2017.

[16] Nicolas Courty, Rémi Flamary and Mélanie Ducoffe. *Learning Wasserstein Embeddings*. arXiv:1710.07457v1 [stat.ML], 20 Oct 2017.

[17] Victor M. Panaretos and Yoav Zemel. *Statistical Aspects of Wasserstein Distances*. arXiv:1806.05500v2 [stat.ME], 11 Aug 2018.

[18] Martin Arjovsky, Léon Bottou. *Towards Principled Methods for Training Generative Adversarial Networks*. arXiv:1701.04862v1 [stat.ML], 17 Jan 2017.

[19] Jian Shen, Yanru Qu, Weinan Zhang, Yong Yu. *Wasserstein Distance Guided Representation Learning for Domain Adaptation*. arXiv:1707.01217 [stat.ML], 9 Mar 2018.

[20] Cédric Villani. *Optimal transport, old and new*. Springer, June 13, 2008.

[21] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, Aaron Courville. *Improved Training of Wasserstein GANs*. arXiv:1704.00028 [cs.LG], 25 Dec 2017.

[22] Yves Grandvalet, Yoshua Bengio. *Semi-supervised Learning by Entropy Minimization*. 2005.

[23] Zhangjie Cao, Lijia Ma, Mingsheng Long, and Jianmin Wang. *Partial Adversarial Domain Adaptation*. arXiv:1808.04205 [cs.CV], 10 Aug 2018.

[24] Zhangjie Cao, Mingsheng Long, Jianmin Wang, Michael I. Jordan. *Partial Transfer Learning with Selective Adversarial Networks*. arXiv:1707.07901 [cs.LG], 25 July 2017.

[25] Jing Zhang, Zewei Ding, Wanqing Li, Philip Ogunbona. *Importance Weighted Adversarial Nets for Partial Domain Adaptation*. arXiv:1803.09210v2 [cs.CV], 28 Mar 2018.

[26] Lisa Torrey and Jude Shavlik. *Transfer Learning*. University of Wisconsin, Madison WI, USA.

[27] Thomas M. Cover, Joy A. Thomas. *Elements of Information Theory*. Second Edition. Wiley-Interscience, John Wiley & Sons, Inc., 2006, New Jersey. ISBN-13 978-0-471-24195-9

[28] Sergios Theodoridis. *Machine Learning: A Bayesian and Optimization Perspective*. Elsevier, 2015.

[29] Yu Zhang and Qiang Yang. *A Survey on Multi-Task Learning*. arXiv:1707.08114v2 [cs.LG], 27 Jul 2018.

[30] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, Fran cois Laviolette, Mario Marchand, Victor Lempitsky. *Domain-Adversarial Training of Neural Networks*. arXiv:1505.07818v4 [stat.ML] 26 May 2016.

[31] Yishay Mansour, Mehryar Mohri, Afshin Rostamizadeh. *Domain Adaptation: Learning Bounds and Algorithms*. arXiv:0902.3430v2 [cs.LG] 23 Feb 2009.

[32] John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman. *Learning Bounds for Domain Adaptation*. NIPS Proceedings, 2008.