Generalized Domain Adaptation for Visual Domains

by

Bhadrinath Nagabandi

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2020 by the
Graduate Supervisory Committee:

Sethuraman Panchanathan, Co-Chair
Hemanth Venkateswara, Co-Chair
Troy McDaniel

ARIZONA STATE UNIVERSITY

May 2020

ABSTRACT

Humans have a great ability to recognize objects in different environments irrespective of their variations. However, the same does not apply to machine learning models which are unable to generalize to images of objects from different domains. The generalization of these models to new data is constrained by the domain gap. Many factors such as image background, image resolution, color, camera perspective and variations in the objects are responsible for the domain gap between the training data (source domain) and testing data (target domain). Domain adaptation algorithms aim to overcome the domain gap between the source and target domains and learn robust models that can perform well across both the domains.

This thesis provides solutions for the standard problem of unsupervised domain adaptation (UDA) and the more generic problem of generalized domain adaptation (GDA). The contributions of this thesis are as follows. (1) Certain and Consistent Domain Adaptation model for closed-set unsupervised domain adaptation by aligning the features of the source and target domain using deep neural networks. (2) A multi-adversarial deep learning model for generalized domain adaptation. (3) A gating model that detects out-of-distribution samples for generalized domain adaptation. The models were tested across multiple computer vision datasets for domain adaptation. The dissertation concludes with a discussion on the proposed approaches and future directions for research in closed set and generalized domain adaptation.

ACKNOWLEDGEMENTS

The submission of my thesis concludes my incredible journey at Arizona State University. I would like to express my gratitude to many wonderful people who helped me to cross the finish line.

First and foremost, I would like to sincerely thank my advisor, Dr. Hemanth Venkateswara, for his continuous support. Without his guidance and motivation, the thesis would not have been possible. I enjoyed an enormous amount of freedom in working the problems of my interest. I was also consistently backed up financially and emotionally by him. Dr. Hemanth was very knowledgable, inspiring, approachable, and truly humble. I would also like to thank my committee member Dr. Sethuraman Panchanathan and Dr. Troy McDaniel for their support, guidance, and inspiring conversations.

I'm blessed to have had the opportunity to work with an amazing set of peers. I would like to thank Raghav, Maunil, Rishab, Piyush, and Andrew for their continuous support and having many productive conversations. I would also like to thank the Ph.D. students Bijan, Meredith for sharing their incredible experiences. My time at ASU would not have been more exciting without the love and support of my friends Sumanth, Mounika, Karna, Surendra, Vinay, Bryan, Abhik, josh, Michael.

Finally, I dedicate my thesis to my parents, Ramana and Rani, and my sister Mani for all the years of love and support.

TABLE OF CONTENTS

Page

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Goals and Motivation

Deep neural networks have shown impressive performance on a large number of computer vision tasks like Image Classification, Object Detection, Image Segmentation, etc. However, the critical factor behind these performances was the availability of labeled data. With large amounts of labeled data, complex models with enough capacities are designed to achieve human-level performances. However, in practice, it is not always feasible to collect and annotate large amounts of data to sufficiently train a model for the required task. Moreover, the conventional learning algorithms trained on one task do not generalize well to a relevant but new task owing to domain shift Ben-David *et al.* (2010). It is because all these learning algorithms are trained with an underlying assumption that the data used for both training and testing are sampled from the same distribution. For example, Consider you have developed a model that can classify plants into different categories with very high accuracies. Then you built an application and deployed the model into smartphones with the hope to classify plant species captured with those cameras. Does the model classify these test images with the same accuracy?

Our previous knowledge strongly hints a ***no***. We reason the drop in performance of the classifier with several factors such as the change in resolution of the camera, different backgrounds, change in illumination, the difference in intrinsic and extrinsic parameters of the camera, photographer's preferences on shooting angles, etc. Multiple solutions do exist to counter the performance issues. A simple solution is

to collect, annotate, and fine tune the model for all the devices of interest. However, this solution is infeasible. An alternative approach is to build models that can adapt to these mismatch between the data distributions and perform the task well. Domain Adaptation algorithms overcome the domain shift between the source and target domain to learn a robust mapping that learns only from the source domain and generalizes to the target domain.

1.2 Contributions

The contributions of the dissertation are as follows.

1. A new semi-supervised learning based approach is proposed for Unsupervised Domain Adaptation. The model predicts the target label with an exponential moving average version of the same model.

2. The problem of Generalized Domain Adaptation is introduced. A double discriminator approach to align source and target distributions and at the same time reject the out-of-distribution samples from the target domain is proposed.

3. A probabilistic approach for Generalized Domain Adaptation is proposed. The designed architecture introduces a gating module that spits out a probability for each sample being seen and unseen and aid the classification of seen samples and rejection of unseen from the target domain.

1.3 Dissertation Outline

The dissertation is structured in the following manner.

Chapter 2 briefly presents the problem of Domain Adaptation. Furthermore, it outlines a mathematical notation for Domain Adaptation. The mathematical notation clearly describes the source and target domains and is consistent throughout

the dissertation. This chapter also introduces the current state of Domain Adaptation in computer vision. Finally, the chapter concludes by describing the challenging datasets that are being solved in the field of computer vision.

Chapter 3 is a literature survey on Domain Adaptation. It also outlines the research in other areas that are relevant to Domain Adaptation. Firstly, the survey begins with some of the recent advances in the Unsupervised Domain Adaptation. The following section describes the similarity of Unsupervised Domain Adaptation with semi-supervised learning and outlines the relevant ideas in semi-supervised learning. The subsequent section would be an overview of Partial Domain Adaptation followed by Open-set Domain Adaptation. These two settings slightly relax the assumptions on label spaces of Closed-Set Domain Adaptation and can be considered as sub problems of Generalized Domain Adaptation. Finally, the chapter concludes with a review of the Universal Domain Adaptation from the existing literature.

Chapter 4 describes a deep domain adaptation model based on Rapid-Smooth networks that are similar to Mean-Teacher models from semi-supervised learning. The chapter introduces a Certain and Consistent Domain Adaptation model for Closed-Set Domain Adaptation. The algorithm trains a pair of models with a unique objective function to predict the labels for the samples from the target domain. The chapter also makes a detailed analysis of the results and perform ablation studies to understand the individual contributions of the model. Finally, the chapter concludes with a comment on the performance of the model and future directions.

Chapter 5 progresses from Closed-Set Unsupervised Domain Adaptation to Generalized Domain Adaptation. The chapter introduces a Multi Adversarial approach for Generalized Domain Adaptation. The model aligns the source and target distributions only in the shared label space to adapt to the domain mismatch. Furthermore, it minimizes the probability of being classified as a source private class and simulta-

neously rejects the samples from the target private class as out-of-distribution(OOD) samples. The chapter displays the results with a couple of experiments and finally concludes with few comments on the model and future directions of the approach.

Chapter 6 introduces a probabilistic approach for Generalized Domain Adaptation. The model aligns the source and target distributions only in the shared label space to adapt to the domain mismatch. Furthermore, the model incorporates a gating module to predict the probability of a sample being seen and unseen. The predicted probabilities are used in output smoothing and followed by thresholding to reject out-of-distribution samples. The following section investigates the approach with a set of experiments and concludes with observations and future work.

Chapter 2

DOMAIN ADAPTATION

2.1 Introduction to Domain Adaptation

The goal of Domain Adaptation is to learn a model that can adapt to the mismatches in data distribution and perform better to the tasks in hand. In Domain Adaptation, there is a source domain and a target domain. The goal in domain adaptation algorithms is to overcome the domain shift between the source and the target domain to learn a robust model that transfers the knowledge from the source domain to the target domain. Thus, a critical aspect of domain adaptation algorithms is to learn a mapping that is invariant to domain shift. If the source domain is fully labeled and the target domain is unlabeled yet assumed to have identical label space between the source and target domains, the problem is termed as Unsupervised Domain Adaptation (UDA). Recent approaches Ganin *et al.* (2016); Tzeng *et al.* (2017); Pei *et al.* (2018); Long *et al.* (2015); Venkateswara *et al.* (2017b); Bousmalis *et al.* (2017); Deng *et al.* (2018); Hoffman *et al.* (2018) in Unsupervised Domain Adaptation are capable of producing excellent performance on many domain adaptation tasks Saenko *et al.* (2010); Venkateswara *et al.* (2017b). These methods largely rely on minimizing the distance between the source and target domains by aligning the feature space with a distance metric Long *et al.* (2017b). Other approaches Ganin *et al.* (2016); Pei *et al.* (2018) try to learn domain invariant features using an adversarial domain discriminator. In the former approaches, the source and target features are projected into a high dimensional space and a statistic criterion is used to minimize the distance between them. Earlier works utilized Maximum Mean Discrepancy Long *et al.*

(2017b), KL-Divergence, Wasserstein distance Shen *et al.* (2018) as distance criteria. However, recent approaches use an adversarial domain discriminator to separate the source features from the target features. Furthermore, recent approaches also introduce a gradient reversal layer. The gradient reversal layer backpropagates the negative of gradient from the adversarial discriminator to the feature extractor to learn domain invariant features. Hence, a classifier learned on these domain invariant features generalizes well to the source and target domains.

Irrespective of these impressive performances, the existing approaches Ganin *et al.* (2016); Tzeng *et al.* (2017); Pei *et al.* (2018); Long *et al.* (2015); Venkateswara *et al.* (2017b) in the literature are not readily applicable for real-world scenarios because the approaches make a closed-set assumption. In a closed-set scenario, it is assumed that the source domain and target domain has identical label space but with a domain shift, i.e. the categories in the source and target domains are the same. However, in real-world scenarios, the assumption does not hold all the time. Very often any of the following scenarios can be encountered. 1) The target domain's label distribution is a subset of the source domain's label distribution. 2) The source domain's label distribution is a subset of the target domain's label distribution. 3) The source and target domain's label distribution overlap is unknown.

In the current literature, there is also a line of work that relaxed the closed-set assumption and aimed towards scenarios 1 and 2. The instances where the target domain is a subset of the source domain is referred to as Partial Domain Adaptation Cao *et al.* (2018b); Zhang *et al.* (2018); Cao *et al.* (2019). Similarly, Busto et al.Panareda Busto and Gall (2017) proposed a scenario where the source domain is a subset of the target domain and named the framework as Open-set Domain Adaptation. In this work, they also introduced the idea of an 'unknown' class i.e., classes that belong only to the target domain but not to the source domain are referred to as

unknown. But, in their work, they assumed the common classes between the source and target domains are known during training. Saito et. al Saito *et al.* (2018) further relaxed the assumption on the prior knowledge of unknown classes in the source domain and modified the problem of open-set domain adaptation. Ideas explored in both the Partial Domain Adaptation and Open-set Domain Adaptation paved the path for a typical realistic scenario called Generalized Domain Adaptation.

Generalized Domain Adaptation is a realistic and more complicated setting. There do exist many real-world problems with a fixed and pre-determined label space common to both the source and target domains. However, a closer observation reveals that these scenarios are merely special cases of Generalized Domain Adaptation and the real problems are more often unconstrained in their label spaces. In Generalized Domain Adaptation, we have a labeled source domain and a related but different target domain that is unlabeled. However, in this setting, the prior knowledge on their label spaces is unknown hence termed 'Generalized'. Generalized Domain Adaptation aims to learn a model from the source domain to generalize it to the target domain irrespective of the domain gap and the category gap. An additional challenge here is to classify a target sample if and only if it belongs to one of the source classes and mark it 'unknown' when it is an outlier.

Generalized Domain Adaptation has to deal not only with the domain gap but also the category gap - the difference in label spaces between the source and target domains. Hence, none of the existing domain adaptation solutions are suitable in this setting because of the domain gap and category gap in the unlabeled target domain. As the target label space is unknown, one has no clue about which categories of source domain have to be aligned with the target domain. Naively aligning the source and target domains doesn't help and top of it can cause a negative transfer. Finally, the literature also shows that out-of-distribution samples tend to be classified as

one of the source classes with very high confidence, hence, rejecting those sample as 'unknown' is an arduous task. To address Generalized Domain Adaptation, we propose a novel architecture that overcomes the domain shift and category shift to learn a robust model that generalizes to the target domain. We implement our model with novel loss criteria and reweighing mechanisms to learn from the source domain and generalize it to the target domain.

In the following section, I formalize the idea of Domain Adaptation and introduce a notation for UDA and GDA that is consistent over the dissertation.

2.2 Mathematical Notation

In Generalized Domain Adaptation setup I consider a source domain $\mathcal{D}_s = \{\boldsymbol{x}_i^s, y_i^s\}_{i=1}^{n_s}$ with n_s sample pairs where $\boldsymbol{x}_i^s \in X$ are images sampled from a space X and $y_i^s \in Y$ are their corresponding labels from a discrete label space $Y_s = \{1, \ldots, \mathcal{C}_s\}$. Thus, source domain has $\mathcal{C}_s = |Y_s|$ number of classes. Likewise, let Y_t be the label space of the target domain with $\mathcal{C}_t = |Y_t|$ number of classes. Let $Y = Y_s \cap Y_t$ be the set of labels common to the source and target domains. Also, I define $\bar{Y_s} := Y_s \setminus Y$ and $\bar{Y_t} := Y_t \setminus Y$ to be the set of labels sets private to the source and target domains respectively. I term Y_s as the known categories and $\bar{Y_s}$ as the unknown categories. At the time of training I know $\mathcal{D}_s$, $\mathcal{D}_t$ and Y_s, but I am not privy to Y_t. The goal of the proposed Generalized Domain Adaptation framework is to predict $\hat{y}_t$ for the samples $\hat{\boldsymbol{x}}_t \in \mathcal{D}_t$, whose label $\hat{y}_t \in Y$.

I borrow the notation **commonness**(ξ) from the work Universal Domain Adaptation You *et al.* (2019) for the ease of experiments. ξ between the source and the target domain is defined as the Jaccard distance of the two label sets: $\xi := \frac{|\mathcal{C}_s \cap \mathcal{C}_t|}{|\mathcal{C}_s \cup \mathcal{C}_t|}$. From the defined equation it is very clear that Closed set domain adaptation is a special case of Generalized domain adaptation when $\xi = 1$. Hence, in Generalized

Domain Adaptation, the goal is to design a model that is unaware of ξ but is capable of performing well on both the source and target domain. In overall, the constraints in Generalized Domain Adaptation is that the joint distribution of the source and target are different with $P_s(X, Y) \neq P_t(X, Y)$ and the commonness ξ is unknown.

2.3 Datasets for Domain Adaptation in Computer Vision

Office-31 Saenko *et al.* (2010) is the common benchmark dataset used to evaluate domain adaptation algorithms. The dataset consists of about 4650 images from 31 categories of everyday objects. It has 3 domains: *Amazon*(**A**), *DSLR*(**D**) and *Webcam*(**W**). The Amazon domain has 2817 images whereas Webcam and DSLR have only 795 and 498 images respectively. Models are evaluated on the 6 transfer tasks $\mathbf{A} \rightarrow \mathbf{W}$, $\mathbf{D} \rightarrow \mathbf{W}$, $\mathbf{W} \rightarrow \mathbf{D}$, $\mathbf{A} \rightarrow \mathbf{D}$, $\mathbf{D} \rightarrow \mathbf{A}$ and $\mathbf{W} \rightarrow \mathbf{A}$ across all the domains. Here $\mathbf{A} \rightarrow \mathbf{W}$ implies, **A** is the source and **W** is the target.

Figure 2.1: Samples of Amazon, Webcam and DSLR domains from Office-31 dataset

Office-Home Venkateswara *et al.* (2017b) is a more challenging dataset with more than 15,500 images from 65 categories belonging to the following four domains: *Art*

(**Ar**), *Clipart* (**Cl**), *Product* (**Pr**) and *Real-World* (**Rw**). The image categories are everyday objects from office and home settings. Similar to the *Office-31* experiments, in *Office-Home*, models are evaluated on all the 12 transfer tasks **Ar** $\rightarrow$ **Cl**, **Ar** $\rightarrow$ **Pr**, **Ar** $\rightarrow$ **Rw**, **Cl** $\rightarrow$ **Ar**, **Cl** $\rightarrow$ **Pr**, **Cl** $\rightarrow$ **Rw**, **Pr** $\rightarrow$ **Ar**, **Pr** $\rightarrow$ **Cl**, **Pr** $\rightarrow$ **Rw**, **Rw** $\rightarrow$ **Ar**, **Rw** $\rightarrow$ **Cl** and **Rw** $\rightarrow$ **Pr** across all the domains.

Figure 2.2: Samples of Art, Clipart, Product, and Real World domains from Office-Home dataset

Chapter 3

LITERATURE SURVEY

3.1 Closed Set Domain Adaptation

I provide a brief survey of statistical and adversarial approaches to domain adaptation that are relevant to my work. For a detailed survey of domain adaptation, I direct the reader to Csurka (2017); Venkateswara *et al.* (2017a).

A standard procedure to aligning the data distributions of the source and target is reducing the Maximum Mean Discrepancy (MMD) between source and target features after projecting them onto a high (infinite) dimensional space Long *et al.* (2015, 2016); Venkateswara *et al.* (2017b). The MMD is a non-parametric measure of the difference between two distributions. A variation of the MMD criterion is deployed by Long et al., where they develop a joint MMD using both input features and labels Long *et al.* (2017b). Distribution alignment is also achieved by reducing the Wasserstein's distance or Earth Mover's distance between distributions Bhushan Damodaran *et al.* (2018); Courty *et al.* (2017); Shen *et al.* (2018).

The most popular approach to reduce domain disparity is through adversarial training. Adversarial training was introduced through the Generative Adversarial Networks by Goodfellow et al. Goodfellow *et al.* (2014). Adversarial training in domain adaptation is a two-network min-max game in which one network tries to differentiate between the source and target data and the second network tries to align the two distributions of the datasets. The Domain Adversarial Neural Network (DANN) Ganin *et al.* (2016) is a seminal approach that applied adversarial training to domain adaptation. The DANN has a feature extractor network attempting to

extract domain-aligned features and an auxiliary network called the discriminator that is trained to discriminate between the features of the source and the target. The feature extractor is trained to make the discriminator perform poorly by negating the gradient of the discriminator using a gradient reversal layer (GRL). There have been multiple variations of adversarial training in domain adaptation literature; maximal domain confusion loss Tzeng *et al.* (2015), untied feature extractor and discriminator (ADDA) Tzeng *et al.* (2017) and multiple domain discriminators to enable fine-grained discrimination of data distributions (MADA) Pei *et al.* (2018), to name a few. These approaches apply the adversarial training principle to align feature spaces between domains.

Apart from aligning feature spaces, adversarial training has also been applied to align image spaces. Image translation based domain adaptation approaches convert images from the source domain to the target domain (or vice versa) using adversarial training Bousmalis *et al.* (2017); Deng *et al.* (2018); Hoffman *et al.* (2018). The popular DIRT-T model utilizes adversarial training along with enforcing a cluster assumption with conditional entropy minimization to estimate target labels Shu *et al.* (2018). More recent approaches that consider the challenge of differences in label space between the source and the target also take recourse to adversarial training to align the domains Cao *et al.* (2018c); Saito *et al.* (2018); You *et al.* (2019). In this work, I applied the standard version of the DANN Ganin *et al.* (2016) to reduce domain disparity.

3.2 Semi Supervised Learning

In the semi supervised learning paradigm there is a small set of labeled data and a large set of unlabeled data from the same distribution. The goal in SSL is to train a transductive model that can effectively predict the labels of the target. Recent

literature in SSL focuses on *Consistency Regularization* or *Entropy minimization*.

Under Consistency Regularization the network is expected to maintain consistent predictions for an image under different augmentations. This is also called the *smoothness assumption*. The augmentations can be either in the input space with different stochastic transformations of the input or in the parameter space as in Dropout Srivastava *et al.* (2014). An ensemble of perturbations is usually applied to implement consistency regularization and has shown promising performance Laine and Aila (2016); Tarvainen and Valpola (2017). Laine et al. Laine and Aila (2016), introduced two different models implementing consistency regularization. In the Π-model the unlabeled data is passed through the network twice with different perturbations and a mean square error between the two predictions is minimized to maintain consistency. In Temporal Ensembling, a moving average of predictions is maintained and these are considered as training targets for the unlabeled input.

The Mean-Teacher model Tarvainen and Valpola (2017), defines a pair of coupled identical networks, a Student and a Teacher, where the parameters of the Teacher are a moving average of the Student's parameters. The Student network is trained using a consistency measure (mean-squared loss) between the predictions of the Teacher and the Student. Other measures like consistent attention Zagoruyko and Komodakis (2016) and feature correlation using Gram matrix Gatys *et al.* (2015) have also been developed for the Mean Teacher setup.

Entropy minimization is the other popular technique which forces the decision boundary to cut through low density regions of the target and thus generate confident target predictions Grandvalet and Bengio (2005). The popular DIRT-T approach applies entropy minimization along with adversarial training for domain adaptation Shu *et al.* (2018). In the proposed Certain and Consistent Domain Adaptation (CCDA), I train a pair of networks, `Rapid-Smooth`, to output consistent predictions over the

target along the lines of Mean-Teacher Tarvainen and Valpola (2017). In addition the CCDA also ensures certainty in label predictions by minimizing the variance in predictions across multiple augmentations of a data point Li *et al.* (2019).

3.3 Partial Domain Adaptation

In Partial Domain Adaptation(PDA) Cao *et al.* (2018b,a, 2019); Zhang *et al.* (2018) setting, the label space of the target domain is a subset of the source domain i.e. source domain has all the categories of the target domain with some extra classes that are private to the source domain. The major problem in PDA is the classification of target samples into the private classes of the source domain. Unlike the Closed Set Unsupervised Domain Adaptation, aligning all the features of the source and target domains are not effective and seldom causes negative transferWang *et al.* (2019b) instead. Therefore, the effect of private classes in the source domain has to be minimized. Cao et. alCao *et al.* (2018b) proposed a weighing mechanism that re-weighs the source domain so that the samples from the common classes have a higher weight. Thus, both the source and target domains are aligned in the shared label space using the weighted samples. InCao *et al.* (2018a), multiple discriminators are used to estimate class level weights and instance-level weights. Thus, a selective transfer from the source domain to the target domain is achieved by aligning the features in the shared label space. Similar to Cao *et al.* (2018a), Zhang *et al.* (2018) used only 2 discriminators. One of the discriminators provides the probability of a sample belonging to a shared label space or an outlier class while the other discriminator aligns the source and target domains in the shared label space. Similarly, Cao *et al.* (2019) used an auxiliary label predictor and an auxiliary discriminator to estimate the weight for each sample depending on its transferability.

Since in Partial Domain Adaptation the source domain has additional categories

in its label space, a common approach followed in the existing literature is to down weigh the private classes of the source domain and align the features of the source and target domains in their shared label space.

3.4 Open Set Domain Adaptation

In Open set Domain AdaptationPanareda Busto and Gall (2017); Liu *et al.* (2019); Saito *et al.* (2018) the source domain label space is a subset of the target domain label space i.e. a model trained on the source domain is tested in the real world without any restriction on the label space. While testing, if an outlier(data points which are not from the shared label space) is input to the model, it has to be rejected. The target samples only from the shared label space are to be classified. Busto et. alPanareda Busto and Gall (2017) was the first to propose open set domain adaptation. They used an Assign-and-transfer algorithm to map the target samples to the source classes. Then they trained an SVM for the classification. However, the drawback of their algorithm is, it requires us to know the source and target private classes beforehand. Saito *et al.*Saito *et al.* (2018) used adversarial training and explicitly added an extra class "unknown" in the classification problem. Then, they extracted and rejected the features of the unknown target class. Lie *et al.*Liu *et al.* (2019) proposed a coarse-to-fine approach to progressively separate the unknown class features from known class features. In the coarse step, they trained a binary classifier for each class to obtain the probability of a given sample belonging to that particular class. Furthermore, a threshold is applied to the probabilities to separate the features as a known class or an unknown. Finally, a logistic classifier is trained on these separated samples to finely separate them.

Observe that, the similarity between several approaches from the existing liter-

ature is that all the methods selectively align the features of the source and target domains in the shared label space and reject the out-of-distribution samples. Additionally, these methods motive for a more general and realistic problem setting Generalized Domain Adaptation and provide a path for future research

3.5 Universal Domain Adaptation

The problem highlighted in this dissertation falls into this category. In Generalized Domain Adaptation, the relationship between the source label space and target label space is unknown i.e., the source and target domains may or may not have shared classes but have their private classes. This scenario is more practical as the adaptation is made possible with most of the relevant datasets. In UDAYou *et al.* (2019), a non-adversarial domain discriminator is introduced to estimate the domain similarity for a given sample x. A different sample-level weighing mechanism is introduced for both the source and target samples. The weighing mechanism uses the normalized entropy and the domain similarity to down weigh the samples in the source and target domains that do not belong to the shared label space. Since the class discriminative information is learned only from the source domain the model is dominated by the samples from the source domain, to balance the effect, Universal Domain Adaptation adopts entropy minimization to deliver reliable predictions in both the source and target domains.

Chapter 4

CERTAIN AND CONSISTENT DOMAIN ADAPTATION

4.1 Domain Adaptation with Semi Supervised Learning

In this chapter, I propose a novel deep domain adaptation algorithm that uses adversarial training Ganin *et al.* (2016) to align the features of the source and target domains. I hypothesize that when the source and the target domains are aligned, domain adaptation is very similar to the idea of semi supervised learning. In the paradigm of semi supervised learning, a small labeled dataset and a large unlabeled dataset sampled from the same distribution is available. In domain adaptation, when the domains are aligned, the labeled source domain and unlabeled target domain can be compared to labeled and unlabeled data as encountered in semi supervised learning.

Semi supervised learning (SSL) techniques leverage the unlabeled data in a transductive manner and use it for training while simultaneously learning from the labeled data Chapelle *et al.* (2009). Deep learning based SSL approaches fall under two categories, *Consistency Regularization* and *Entropy Minimization*. The basic assumption in consistency regularization is that a classifier is expected to output the same class probability distribution even after it is augmented or deformed by slightly modifying the pixel content in the input image. This is in line with the *smoothness* assumption which constraints decision boundaries to vary smoothly. Entropy minimization approaches force the decision boundaries to pass through low density regions in the input space so that the model can effectively discriminate between the categories. This is based on the *cluster* assumption which is complementary to the smoothness

assumption. Hence, in this chapter, I propose a novel approach that combines both these assumptions when predicting the target labels.

I propose using an adversarial training Ganin *et al.* (2016) approach to align the source and target domains. With the domains aligned, I take inspiration from the Mean Teacher model Tarvainen and Valpola (2017) and propose a new approach that uses a network pair - `Rapid-Smooth` - to perform consistency regularization. Enforcing consistency regularization when the network parameters are nascent can lead to negative transfer. To counter the negative transfer, I propose using a strategy to select the samples that the model is 'certain' about. The 'Certainty' is estimated by measuring the variance in predictions across stochastic perturbations of the input data Li *et al.* (2019). Additionally, I enforce a consistency loss on these 'certain' samples along with entropy minimization. Finally, considering all the objectives, the approach is termed as Certain and Consistent Domain Adaptation (CCDA).

The contributions in this work are two fold. (1) align the features of the source and target domains thereby reducing the domain adaptation problem to a semi supervised learning (SSL) problem allowing us to avail a rich set of solutions from SSL literature. (2) develop the CCDA model using principles of adversarial learning, entropy regularization, consistency regularization, and prediction certainty. Finally, I evaluate the CCDA model on popular benchmark datasets (*Office-31* Saenko *et al.* (2010) and *Office-Home* Venkateswara *et al.* (2017b)) and demonstrate that the CCDA outperforms competitive baselines from unsupervised domain adaptation literature.

4.1.1 Idea Motivation

To solve this problem I propose a Certain and Consistent Domain Adaptation (CCDA) model - a deep neural network that is trained to predict labels for the target by gradually improving their certainty and consistency over multiple iterations of

training. The CCDA takes source and target images as input and extracts image features by ameliorating the domain discrepancy. The CCDA is a coupled network system that is modeled after the Mean Teacher Tarvainen and Valpola (2017). I slightly tweaked the terminology and renamed them as `Rapid-Smooth` networks to denote their training strategies. The `Rapid-Smooth` networks are used to identify the data samples that have high certainty in their label predictions. These certain data points are used to drive a consistency loss for training the coupled networks. Besides, the source data is used to train the CCDA with a cross-entropy loss and the target data is used to determine the unsupervised entropy loss.

The `Rapid` network has parameters $\{\theta_f, \theta_y\}$ and the `Smooth` network has parameters $\{\bar{\theta}_f, \bar{\theta}_y\}$, where θ_f and $\bar{\theta}_f$ are the parameters of the feature extractor and θ_y and $\bar{\theta}_y$ are the classifier parameters. The CCDA model has a discriminator network with parameters θ_d. The CCDA model is trained using mini-batches of source and target data where each mini-batch consists of source samples $\mathcal{X}_s = \{\boldsymbol{x}_i^s\}_{i=1}^B$ and corresponding labels $\mathcal{Y}_s = \{y_i^s\}_{i=1}^B$ and target samples $\mathcal{X}_t = \{\boldsymbol{x}_i^t\}_{i=1}^B$. The mini-batch size is $2B$ and the number of mini-batches goes from $\tau = 0$ to $\mathcal{T}$. In the following sections, the chapter outlines the different components of the CCDA and discuss them in greater detail. The CCDA model is depicted in Figure 4.1.

4.1.2 Domain Alignment

In order to transfer the knowledge from the source domain to the target domain, one needs to overcome the domain discrepancy between the source and target. I introduce a domain discriminator network G_d with parameters θ_d that will ensure the feature extractors of the `Rapid-Smooth` networks output features that have very little to no domain discrepancy between the source and target features. I follow the adversarial approach proposed in DANN Ganin *et al.* (2016), to train the CCDA to

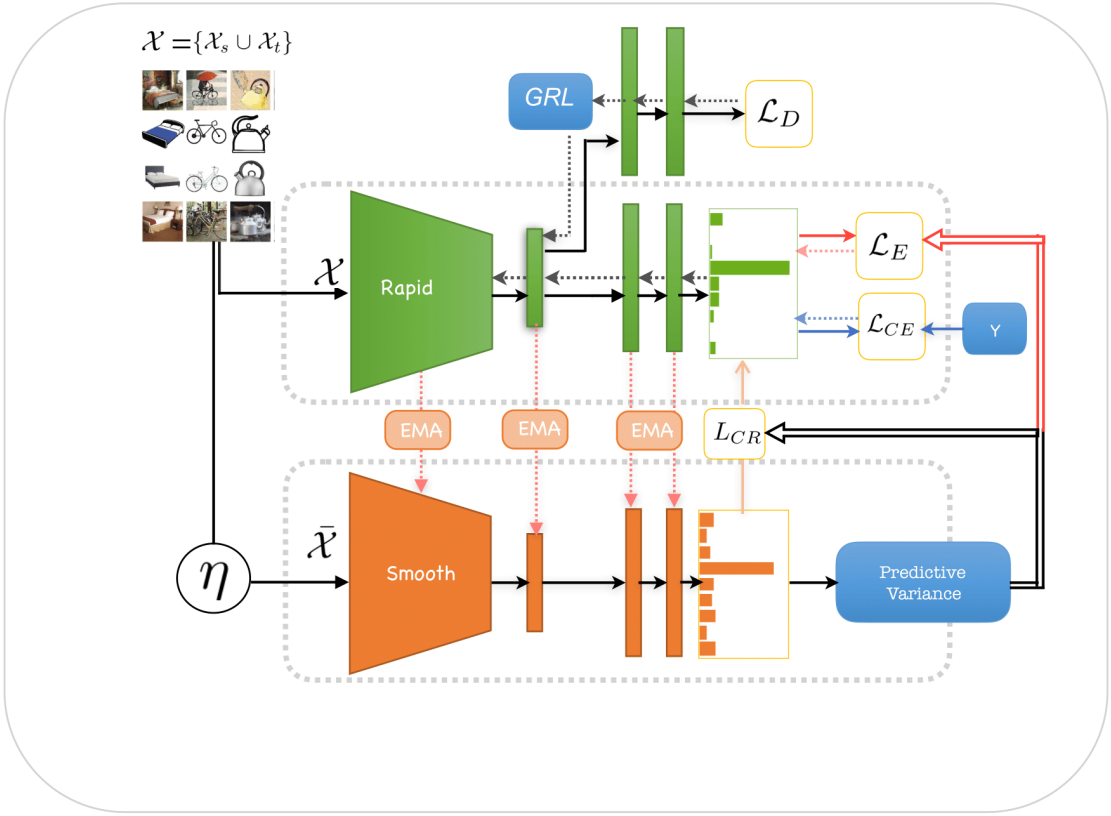


Figure 4.1: The Certain and Consistent Domain Adaptation (CCDA) model. The coupled `Rapid-Smooth` networks have identical architecture based on ResNet-50. The `Rapid` network is trained using loss terms $\mathcal{L}_D$ (discriminator), $\mathcal{L}_E$ (entropy), $\mathcal{L}_{CE}$ (cross-entropy) and $\mathcal{L}_{CR}$ (consistency). The parameters of `Smooth` are an exponential moving average (EMA) of `Rapid`. Random augmentations of the input $\eta(\mathcal{X})$ are used to identify input samples with 'Certain' predictions using predictive variance. The indices of these samples are used to estimate consistency loss (source+target) and entropy loss (target only).

align the feature domains. The data is assigned domain labels where $d = 1$ indicates the sample belongs to the source and $d = 0$ is for the target sample. The discriminator network G_d is trained to distinguish between the source and target features output from the feature extractor (G_f) of the `Rapid` network using the domain label for supervision. The discriminator's objective function is,

$$\mathcal{L}_D(\theta_d, \theta_f) = -\frac{1}{n_s + n_t} \sum_{x \in \{\mathcal{X}_s \cup \mathcal{X}_t\}} d\log[G_d(G_f(\boldsymbol{x}))] + (1-d)(1-\log[G_d(G_f(\boldsymbol{x}))]), \quad (4.1)$$

where $G_d(G_f(\boldsymbol{x}))$, is the output of a sigmoid activation denoting the probability that $\boldsymbol{x}$ belongs to the source. The parameters of G_d are trained using backpropagation by minimizing the objective $\mathcal{L}_D$ which enables the discriminator to distinguish between source and target features. The Gradient Reversal Layer (GRL) reverses the gradient $-\frac{\partial \mathcal{L}_D}{\partial \theta_f}$ when modifying the parameters of the feature extractor G_f (see Figure 4.1). This form of adversarial training ensures the source and target features output from the feature extractor are indistinguishable to the discriminator thereby assuring domain alignment.

4.1.3 Certainty and Consistency

When the distributions of the feature vectors from the source and target are aligned the data can be viewed as coming from a single distribution. The source data can be considered as the labeled set and the target data as the unlabeled dataset and semi supervised learning approaches are applicable in this setting. To further improve the perdictions on the target samples, I incorporate the unlabeled target data into the training process by estimating their pseudo-labels. In the absence of supervision, self ensembling approaches have led the way in skillfully utilizing unlabeled data to evaluate robust labels for the target with transductive training Laine and Aila (2016);

Tarvainen and Valpola (2017). The core idea in these approaches is smoother estimates and consistent predictions across an ensemble induced by multiple random perturbations. I model a coupled network pair along the lines of Tarvainen and Valpola (2017), and induce random perturbations to estimate certain and consistent predictions for the target data. In the following sections, I will discuss the `Rapid-Smooth` coupled network followed by the predictive variance procedure to identify data points with certain and consistent predictions. I then introduce a consistency loss over the coupled-network predictions of these data points.

4.1.4 Rapid-Smooth Coupled Network:

The `Rapid-Smooth` network pair is illustrated in Figure 4.1. The model has two parallel networks, `Rapid` and `Smooth`, similar to the Student and Teacher networks in Tarvainen and Valpola (2017). The networks are so named based on their training strategies. The `Rapid` network updates its weights (θ_f, θ_y) across every mini-batch of source and target data. This leads to a noisy weight update when the target labels are incorrect. On the other hand, the `Smooth` network only updates its weights $(\bar{\theta}_f, \bar{\theta}_y)$ using an exponential moving average (EMA) of the `Rapid` network weights. This results in a relatively smoother weight update which yields significantly better results Tarvainen and Valpola (2017). If τ denotes the mini-batch index, then the weights of the `Smooth` network are updated using,

$$\bar{\theta}_{f,\tau} = \alpha\bar{\theta}_{f,\tau-1} + (1-\alpha)\theta_{f,\tau}, \quad \text{and} \tag{4.2}$$

$$\bar{\theta}_{y,\tau} = \alpha\bar{\theta}_{y,\tau-1} + (1-\alpha)\theta_{y,\tau}, \tag{4.3}$$

where, α is the momentum hyper parameter for the EMA and the `Rapid` and `Smooth` networks are initialized with the same values, i.e., $\bar{\theta}_{f,\tau=0} := \theta_{f,\tau=0}$ and $\bar{\theta}_{y,\tau=0} := \theta_{y,\tau=0}$.

In our bid to deploy the target data for training, the `Rapid` network could be

misguided with incorrect target labels leading to a confirmation bias - a hazard caused by over-reliance on the incorrect target predictions. To mitigate this effect I propose a Certain and Consistent loss where I only penalize the inconsistency in prediction over only those samples on which the model is certain. In the following I will outline a procedure to identify the data samples that the `Smooth` network is certain about.

4.1.5 Measure of Certainty:

Usually, ensemble methods consider the outputs of the `Smooth` (Teacher) network to be sufficiently accurate and apply them to penalize the `Rapid` (Student) network for inconsistencies in their outputs when compared to the `Smooth` network French *et al.* (2017); Laine and Aila (2016); Tarvainen and Valpola (2017). At the beginning of the training procedure, when the network parameters are still close to their random initialization points, there is a chance that the `Smooth` network is not completely certain about its predictions. In view of that, I introduced an uncertainty measure to identify the samples the `Smooth` network is certain about. I propose using predictive variance as a metric to estimate the uncertainty and distinguish between the certain and uncertain samples in the mini-batch Li *et al.* (2019). *Certainty* is the ability of the network to output similar predictions for a data point $\boldsymbol{x}$ under stochastic data augmentations. I used T different stochastic augmentations of a data point $\boldsymbol{x}$ denoted by $\{\eta_t(\boldsymbol{x})\}_{t=1}^{T}$ during a forward pass through the `Smooth` network. The T augmentations for each image are chosen from random flips, crops, rotation, Gaussian noise addition, and occlusion by removing an image patch. The output of the network for each augmentation t is a softmax probability vector $\bar{G}_f(\bar{G}_y(\boldsymbol{x})) = [p(y = 1|\eta_t(\boldsymbol{x}), \bar{\theta_f}, \bar{\theta_y}), \ldots, p(y = \mathcal{C}|\eta_t(\boldsymbol{x}), \bar{\theta_f}, \bar{\theta_y})]^\top$. The goal is to identify samples with consistent predictions across the T random augmentations. I therefore gather the T predictions for the sample and estimate the uncertainty of

prediction using variance which is calculated as,

$$\mu_c = \frac{1}{T}\sum_{t=1}^{T} p(y = c|\eta_t(\boldsymbol{x}), \bar{\theta_f}, \bar{\theta_y}), \tag{4.4}$$

$$PV = \sum_{c=1}^{\mathcal{C}} \Big(\frac{1}{T}\sum_{t=1}^{T} \big(p(y = c|\eta_t(\boldsymbol{x}), \bar{\theta_f}, \bar{\theta_y}) - \mu_c\big)^2\Big). \tag{4.5}$$

Here, PV stands for predictive variance. With PV as the criteria, I sort all the data samples in the mini-batch (consisting of both the source and target data samples) in ascending order. The data sample the `Smooth` network is most 'certain' about comes first and the least certain data sample comes last in this sorted list. The output of the predictive variance procedure is a set of indices $\mathcal{I}(\{\mathcal{X}_s \cup \mathcal{X}_t\}, PV, \tau)$ identifying the source ($\mathcal{X}_s$) and target ($\mathcal{X}_t$) samples that the `Smooth` network is certain about. Here, $\mathcal{I}(\mathcal{X}, PV, \tau)$ is the set of indices chosen from dataset $\mathcal{X}$ based on the predictive variance PV and mini-batch index τ. Rather than filtering the list with a threshold certainty value, I used a sigmoid ramp-up strategy based on the mini-batch number τ to incrementally identify the samples as 'certain' from the sorted list. These indices are used to determine the Consistency regularization and the Entropy loss which is outlined in the following.

4.1.6 Consistency Regularization:

Consistency regularization is very crucial to leverage the unlabeled target data. It is applied to ensure the `Rapid` classifier outputs similar probability distribution compared to the `Smooth` classifier under different transformations. This is implemented by penalizing the `Rapid` network with a consistency regularization term for deviations in the predictions compared to the `Smooth` network. In order to avoid the rapid network trying to be consistent with smooth network's uncertain predictions at the beginning of training, I apply the consistency loss across the predictions of only those

samples the `Smooth` network is certain about. The certain samples indices are given by $\mathcal{I}_\tau = \mathcal{I}(\{\mathcal{X}_s \cup \mathcal{X}_t\}, PV, \tau)$. More formally, the consistency regularization term is expressed as,

$$\mathcal{L}_{CR}(\theta_f, \theta_y) = \frac{1}{|\mathcal{I}_\tau|} \sum_{i \in \mathcal{I}_\tau} \left|\left|G_y(G_f(\eta(\boldsymbol{x}_i))) - \bar{G}_y(\bar{G}_f(\eta(\boldsymbol{x}_i)))\right|\right|_2^2, \tag{4.6}$$

where $\eta(\boldsymbol{x})$ is a random perturbation of the input image.

4.1.7 Entropy Regularization

Existing literature demonstrates that a model trained using only source data tends to be highly confident on the source like samples and less confident on target like samples Vu *et al.* (2019). Besides, when the model has not explored the target data space there is a high probability that the decision boundary of the model passes through high density regions of the target space which implies target data classification is incorrect. In order to ensure a low-density separation between target classes and to utilize the target data for training, I include an additional loss term entropy minimization Grandvalet and Bengio (2005); Long *et al.* (2016); Shu *et al.* (2018). For a given image $\boldsymbol{x}$, the softmax output of the `Rapid` network is $G_f(G_y(\boldsymbol{x})) = [f_1(\boldsymbol{x}), \ldots, f_C(\boldsymbol{x})]^\top$, where $f_j(\boldsymbol{x}) = p(y = j|\boldsymbol{x}, \theta_f, \theta_y)$ - the probability that image $\boldsymbol{x}$ belongs to class j. The output $G_f(G_y(\boldsymbol{x}))$ is a probability vector whose components sum to 1. When the network has high confidence in prediction the output is similar to a one-hot vector where all the components of the probability vector are zeros except for one component. Such a prediction has zero (low) entropy. When the network predicts that the input image $\boldsymbol{x}$ belongs to all classes with equal probability, such a prediction has the highest entropy - the network is not confident about the label. By minimizing entropy the model i.e., `Rapid` network is forced to have confident predictions over the target data. In the early stages of training, the model usually has random predictions on a

target sample. Rather than forcing the model to be confident in random predictions, we identify the target samples the network is certain about using our certainty measures outlined earlier. Let $\mathcal{I}_{t,\tau} = I(\mathcal{X}_t, PV, \tau)$ be the indices of the target samples for which the `Smooth` network has high certainty predictions for mini-batch index τ. The entropy regularization loss is then given by,

$$\mathcal{L}_E(\theta_f, \theta_y) = -\frac{1}{|\mathcal{I}_{t,\tau}|} \sum_{i \in \mathcal{I}_{t,\tau}} \sum_{j=1}^{\mathcal{C}} f_j(\boldsymbol{x}_i) \log f_j(\boldsymbol{x}_i). \tag{4.7}$$

4.1.8 Cross Entropy Loss

Cross entropy is the standard supervised classification loss for multi-class classification. `Rapid` network uses the source data in the mini-batch to minimize the cross-entropy loss over the known labels. If the source images and their corresponding labels in a mini-batch are represented as $\{\mathcal{X}_s, \mathcal{Y}_s\}$, the cross entropy loss is given by,

$$\mathcal{L}_{CE}(\theta_f, \theta_y) = -\frac{1}{|\mathcal{X}_s|} \sum_{\boldsymbol{x} \in \mathcal{X}_s, y \in \mathcal{Y}_s} \sum_{j=1}^{\mathcal{C}} 1\{y = j\} \log f_j(\boldsymbol{x}), \tag{4.8}$$

where, $1\{\texttt{cond}\}$ is an indicator function which is true if the `cond` is true.

4.1.9 CCDA Objective Functions

The `Rapid` network is trained with an objective function that brings together multiple loss terms. The overall objective function brings together the discriminator loss Equation (6.2), the consistency regularization Equation (4.6), the entropy loss Equation (6.8) and the cross-entropy loss Equation (6.1). The parameters of the `Rapid` network are modified using,

$$\{\theta_f^*, \theta_y^*\} = \underset{\theta_f, \theta_y}{\operatorname{argmin}} \left[\mathcal{L}_{CE} + \gamma \mathcal{L}_{CR} + \beta \mathcal{L}_E - \lambda \mathcal{L}_D\right] \quad \text{and} \tag{4.9}$$

$$\{\theta_d^*\} = \underset{\theta_d}{\operatorname{argmin}} \left[\lambda \mathcal{L}_D\right], \tag{4.10}$$

where, γ, β and λ are hyper parameters that control the importance of individual loss terms. While Equations (4.9) and (4.10) update the parameters of the `Rapid` network and the adversarial discriminator, the parameters of the `Smooth` network are updated using the exponential moving average (EMA) as outlined in Equations (4.2), (4.3).

4.2 Experiments & Analysis

The performance of the model is evaluated on two benchmark datasets and compared the results with other competitive domain adaptation algorithms.

4.2.1 Experimented Datasets

Office-31 Saenko *et al.* (2010) is the common benchmark dataset used to evaluate domain adaptation algorithms. The dataset consists of about 4650 images from 31 categories of everyday objects. It has 3 domains: *Amazon*(**A**), *DSLR*(**D**) and *Webcam*(**W**). The Amazon domain has 2817 images whereas Webcam and DSLR have only 795 and 498 images respectively. I evaluated the model performance on the 6 transfer tasks $\mathbf{A} \rightarrow \mathbf{W}$, $\mathbf{D} \rightarrow \mathbf{W}$, $\mathbf{W} \rightarrow \mathbf{D}$, $\mathbf{A} \rightarrow \mathbf{D}$, $\mathbf{D} \rightarrow \mathbf{A}$ and $\mathbf{W} \rightarrow \mathbf{A}$ across all the domains. Here $\mathbf{A} \rightarrow \mathbf{W}$ implies, **A** is the source and **W** is the target.

Office-Home Venkateswara *et al.* (2017b) is a more challenging dataset with more than 15,500 images from 65 categories belonging to the following four domains: *Art* (**Ar**), *Clipart* (**Cl**), *Product* (**Pr**) and *Real-World* (**Rw**). The image categories are everyday objects from office and home settings. Similar to the *Office-31* experiments, the model performance is evaluated on all the 12 transfer tasks $\mathbf{Ar} \rightarrow \mathbf{Cl}$, $\mathbf{Ar} \rightarrow \mathbf{Pr}$, $\mathbf{Ar} \rightarrow \mathbf{Rw}$, $\mathbf{Cl} \rightarrow \mathbf{Ar}$, $\mathbf{Cl} \rightarrow \mathbf{Pr}$, $\mathbf{Cl} \rightarrow \mathbf{Rw}$, $\mathbf{Pr} \rightarrow \mathbf{Ar}$, $\mathbf{Pr} \rightarrow \mathbf{Cl}$, $\mathbf{Pr} \rightarrow \mathbf{Rw}$, $\mathbf{Rw} \rightarrow \mathbf{Ar}$, $\mathbf{Rw} \rightarrow \mathbf{Cl}$ and $\mathbf{Rw} \rightarrow \mathbf{Pr}$ across all the domains.

4.2.2 Implementation Details

The pre-trained Resnet-50 He *et al.* (2016) model from PyTorch Paszke *et al.* (2017) is the base neural network i.e., the feature extractor. I removed the original classifier and added a bottleneck layer of 256 dimensions after the global average pooling layer. Similar to Long *et al.* (2017a), a classifier and a domain discriminator (dimensions 1024-1024-1) are defined after the bottleneck layer. As the classifier and discriminator are trained from scratch we use 10 times the learning rate that is used to fine-tune the feature extractor. We use the same learning rate strategy implemented in Ganin *et al.* (2016): with $\eta_p = \frac{\eta_0}{(1+\alpha p)^\gamma}$, where p is the training progress varying between $[0, 1]$, while η_0, α and γ are optimized with importance-weighted cross-validation Sugiyama *et al.* (2007). The hyperparameters are set as the default values provided from Ganin *et al.* (2016) without further fine tuning. To update the weights, mini-batch stochastic gradient descent with Nesterov as the optimizer is used. Similar to Long *et al.* (2017a), a weight decay of $5e-4$ with a momentum $= 0.9$ in the optimizer is used. For the `Rapid-Smooth` network, momentum α for EMA $= 0.999$ Tarvainen and Valpola (2017). Also, I followed a sigmoid ramp-up strategy from Tarvainen and Valpola (2017) to filter the samples on which the model is confidently certain about.

4.2.3 Results

The results of Certain and Consistent Domain Adaptation model on *Office-31* and *Office-Home* are shown in Table 4.1 and Table 4.2 respectively. All the reported scores are the classification accuracies for different tasks. I only note the baselines that are relevant to my work and also report their results for comparison. For a fair comparison, the accuracies for baselines are directly reported from their original

papers. In addition to comparing with competitive baselines, I also compare the full model (CCDA)'s performance with CCDA-without-predictive-variance (CCDA w/o PV) and CCDA-without-entropy-and-predictive-variance (CCDA w/o (Ent+PV)). As reported in Table 4.1, the model outperforms other methods in most of the tasks. The average performance of the model across all the tasks is also better than the other approaches. Particularly, in tasks like **A** $\rightarrow$ **D** and **D** $\rightarrow$ **W**, my model has the highest accuracy. Experiments **D** $\rightarrow$ **A** and **W** $\rightarrow$ **A** have low accuracies across all methods. This can be attributed to the data imbalance between the source and target datasets.

Office-Home is a more difficult challenge with more number of domains and categories. From Table 4.2, it can be seen that the proposed method outperforms the baselines with a significant margin. Particularly in the transfer tasks **Cl** $\rightarrow$ **Pr** and **Pr** $\rightarrow$ **Ar**, the improvement is around 10%. In the following transfer learning tasks **Ar** $\rightarrow$ **Pr**, **Cl** $\rightarrow$ **Rw**, **Rw** $\rightarrow$ **Ar** and **Rw** $\rightarrow$ **Pr**, the CCDA improves over the best-reported method by a margin of 6%.

Table 4.1: Accuracy of CCDA on *Office-31* (ResNet-50)

Method	A $\rightarrow$ W	D $\rightarrow$ W	W $\rightarrow$ D	A $\rightarrow$ D	D $\rightarrow$ A	W $\rightarrow$ A	Avg
ResNet He *et al.* (2016)	68.4	96.7	99.3	68.9	62.5	60.7	76.1
DAN Long *et al.* (2015)	80.5	97.1	99.6	78.6	63.6	62.8	80.4
RTN Long *et al.* (2016)	84.5	96.8	99.4	77.5	66.2	64.8	81.6
DANN Ganin *et al.* (2016)	82	96.9	99.1	79.7	68.2	67.4	82.2
ADDA Tzeng *et al.* (2017)	86.2	96.2	98.4	77.8	69.5	68.9	82.9
JAN Long *et al.* (2017b)	85.4	97.4	**99.8**	84.7	68.6	**70.0**	84.3
MADA Pei *et al.* (2018)	**90**	97.4	99.6	87.8	**70.3**	66.4	85.2
CCDA(w/o (Ent+PV))	83.6	97.4	99.6	80.8	68.1	67.6	82.8
CCDA(w/o PV)	88.1	**98.9**	99.3	88.1	66.8	66.5	84.6
CCDA	89.5	**98.9**	99.7	**91.4**	66.7	66.4	**85.4**

Table 4.2: Accuracy of CCDA on *Office-Home* (ResNet-50)

Method	**Ar→Cl**	**Ar→Pr**	**Ar→Rw**	**Cl→Ar**	**Cl→Pr**	**Cl→Rw**	**Pr→Ar**	**Pr→Cl**	**Pr→Rw**	**Rw→Ar**	**Rw→Cl**	**Rw→Pr**	Avg
ResNet He *et al.* (2016)	34.9	50	58	37.4	41.9	46.2	38.5	31.2	60.4	53.9	41.2	59.9	46.1
DAN Long *et al.* (2015)	43.6	57	67.9	45.8	56.5	60.4	44.0	43.6	67.7	63.1	51.5	74.3	56.3
DANN Ganin *et al.* (2016)	45.6	59.3	70.1	47.0	58.5	60.9	46.1	43.7	68.5	63.2	51.5	76.8	57.60
JAN Long *et al.* (2017b)	45.9	61.2	68.9	50.4	59.7	61.0	45.8	43.4	70.3	63.9	52.4	76.8	58.31
CCDA(w/o (Ent+PV))	46.3	61.2	70.8	48.3	59.6	60.4	47.2	43.1	68.8	64.7	53.6	77.6	58.5
CCDA(w/o PV)	47.1	**67.2**	74.1	54.6	68.9	66.9	53.1	**49.1**	73.6	68.1	57.8	80.6	63.4
CCDA	**48.2**	67.1	**74.6**	**55.6**	**71.4**	**69.3**	**54.5**	48.3	**76.5**	**68.7**	**58.8**	**82.5**	**64.6**

4.2.4 Ablation Studies

In the CCDA, the parameters of the `Smooth` network are an exponential moving average (EMA) of the weights of the `Rapid` network. I further conducted a study to evaluate the advantage of an EMA weight update. I trained the `Rapid-Smooth` network without the consistency loss $\mathcal{L}_{CR}$ and without the entropy regularization $\mathcal{L}_E$, i.e., the `Smooth` network is merely an EMA version of the `Rapid` while the `Rapid` is trained with cross-entropy $\mathcal{L}_{CE}$ and domain discrimination $\mathcal{L}_D$. These results are depicted in the row CCDA(w/o (Ent+PV)). Note that without the consistency and entropy regularization, the `Rapid` network is similar to the DANN. From both the tables one can observe that the `Smooth` network performs consistently better than the DANN leading us to conclude that an ensemble update of the parameters of a network using EMA is a better update than a regular update. I also want to highlight the fact that the `Rapid-Smooth` model does not incur any significant computation cost as the `Smooth` network is only an exponential moving average of the `Rapid` network.

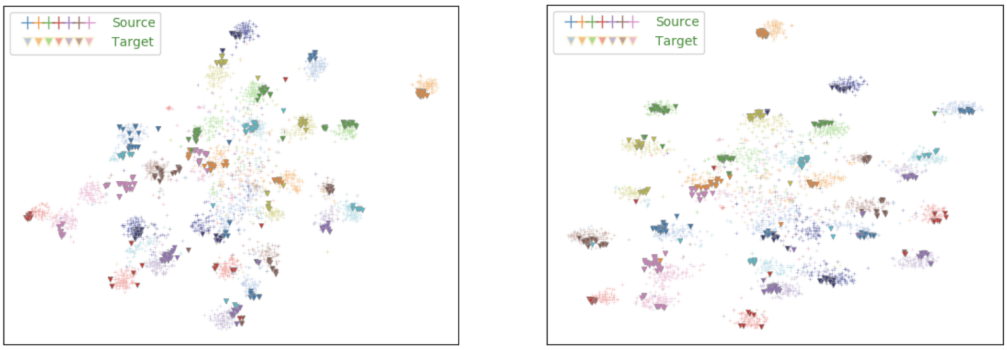
In Section 4.1.7, I discussed that a classifier trained only on source data is likely to misclassify target data as there is a high probability that the decision boundaries cut

through high density regions of the target distribution. To validate this hypothesis I conducted another set of experiments by introducing a penalty over the target classification using entropy regularization. We expect a low entropy penalty on the target samples to improve the confidence of the model on target predictions and force the decision boundaries to pass through low density regions in the target space. The results obtained by including entropy are in row CCDA(w/o PV). The effect of entropy regularization significantly boosts the accuracies as seen in both Table4.1 and Table4.2.

The results of introducing consistency regularization $\mathcal{L}_{CR}$, entropy regularization $\mathcal{L}_E$ along with predictive variance (PV) gives us the entire CCDA. The `Smooth` network can be uncertain on the target predictions at the early stages of training. The $\mathcal{L}_{CR}$ with Certainty ensures that the `Rapid` network is updated using samples the `Smooth` network is 'certain' about. The effect of selectively penalizing consistency between the `Rapid` and the `Smooth` network pays rich dividends as seen in the CCDA row of the tables. I further evaluated the effect of penalizing only certain samples by comparing the CCDA model to CCDA-without-Certainty (CCDA w/o Certainty), where I penalize all the samples for consistency. I also compared the test accuracies of $\mathbf{A} \rightarrow \mathbf{D}$ transfer learning task from *Office-31* dataset in Figure 4.3. Finally, I observed that the performance of the model with Certainty measure is always smooth and largely monotonic, whereas the performance of (CCDA w/o Certainty) model is noisy and converges rapidly with relatively lower accuracy.

4.2.5 Feature Visualization

I visualized the feature space of $\mathbf{A} \rightarrow \mathbf{D}$ transfer learning task from the *Office-31* dataset. The t-SNE Embeddings are used to visualize the features taken from the output of the feature extractor. In Figure 4.2, I show the cluster formations of both



(a) CCDA w/o (Ent+PV) (b) CCDA

Figure 4.2: The t-SNE visualizations of CCDA w/o (Ent+PV) and CCDA features for **A** $\rightarrow$ **D** from *Office-31* with different classes labeled in different colors.

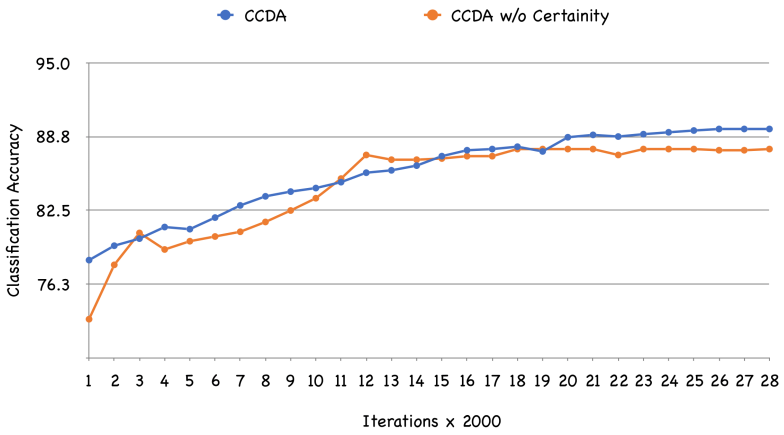


Figure 4.3: Classification accuracies on **A** $\rightarrow$ **D** comparing CCDA and CCDA w/o Certainty vs training iterations.

the source and target domain with (CCDA w/o (Ent+PV)) and the final CCDA model. As expected, the CCDA model forms compact well-defined clusters compared to the baseline. Also, observe that the clusters in CCDA are wide spread with samples of the same class held together and samples of other categories spread farther apart.

4.3 Conclusions

In this chapter, I discussed the Certain and Consistent Domain Adaptation model. The core of the model is based on reducing the domain adaptation problem to a semi supervised learning problem through adversarial domain alignment. I introduce consistency regularization and entropy regularization with a Certainty measure to transductively estimate the target labels. Furthermore, I empirically showed that my model produced competitive results and outperforms state-of-the-art results in a number of transfer learning tasks across benchmark datasets. I also note that in some transfer learning tasks, due to domain imbalance (number of source samples $\ll$ number of target samples), the model performs poorly. Finally, I believe that by re-weighting instances can improve the performance in the imbalanced tasks and I leave it for future work.

Chapter 5

MULTI ADVERSARIAL GENERALIZED DOMAIN ADAPTATION

5.0.1 Introduction

In the earlier chapter, a semi-supervised learning based approach for Unsupervised Domain Adaptation was discussed in detail. However, the problem by itself is biased and impractical because of the underlying assumption. The assumption states that both the source and target domains are similar in their same label space. To present the problem realistically, the assumption on the label spaces behind Unsupervised Domain Adaptation is relaxed and variants such as Partial Domain Adaptation(PDA) and Open-set Domain Adaptation(open-set DA) are proposed. The occurrences where the target label space is a subset of the source label space is referred to as Partial Domain Adaptation Cao *et al.* (2018b); Zhang *et al.* (2018); Cao *et al.* (2019). Similarly, Busto et al.Panareda Busto and Gall (2017) introduced Open-set Domain Adaptation, a scenario where the source label space is a subset of the target label space. Though PDA and Open-DA settings are more realistic than UDA, one still needs to have prior knowledge of the label spaces of the source and target domains. Hence, in this chapter, I outline a new problem setting, namely Generalized Domain Adaptation(GDA). Generalized Domain Adaptation does not require us to have prior knowledge on the label spaces of source and target domains. GDA also enables one to perform adaptation between any datasets without any constraint on the label space.

In this chapter, I propose a Multi-Adversarial Generalized Domain Adaptation(MAGDA). The proposed model simultaneously aligns the features of the source and target do-

mains and also separates the features of the known classes from unknown classes. Finally, a classifier is trained to learn categorical discriminative features that generalize to both the source and target domains. In the following sections, I describe the architecture design and the objective of the model in more detail. Subsequently, I show some experimental results of the model on the office-31 dataset and finally conclude the chapter with comments and future directions.

5.0.2 Overview

As shown in the figure 5.1, the model consists of a feature extractor F with parameters θ_f, an adversarial domain discriminator D with parameters θ_d, an adversarial known-unknown discriminator U with parameters θ_u and a classifier G with parameters θ_g. The model takes in samples from both the source and target domains and extracts features from the feature extractor F. Let $\boldsymbol{x}_i$ be an input sample that is fed into the feature extractor F and $F(\boldsymbol{x})$ be its corresponding feature vector. The feature vector is then fed to the classifier that classifies the input into one of the source categories. A thresholding strategy is applied to the model predictions confidences to reject the out-of-distribution samples. Also, the same feature vector $F(\boldsymbol{x})$ is fed to the adversarial domain discriminator and the known-unknown discriminator. The domain discriminator aligns the features of the source and target domains in the shared label space while the known-unknown discriminator separates the features of the known class from the unknown class. In the following subsections, I will introduce the different components in the proposed model and discuss them in great detail.

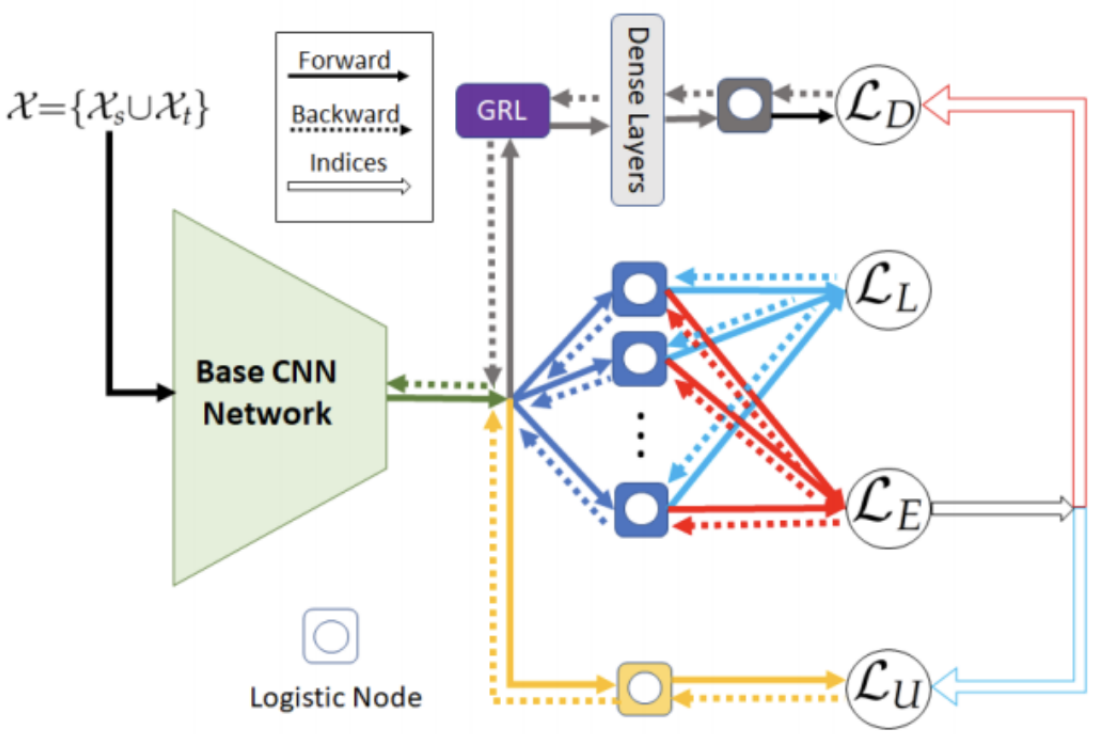


Figure 5.1: Illustration of the Multi Adversarial Generalized Domain Adaptation model. The Base CNN Network is the shared feature extractor that takes in both the source and target images. $\mathcal{L}_L$ is the cross-entropy loss of the source samples. $\mathcal{L}_E$ is the entropy loss minimized over the target logits. $\mathcal{L}_D$ is the Adversarial Domain Discriminator that aligns the source and target features in the shared label space. Finally, $\mathcal{L}_U$ is the discriminator to separate the features of the known and unknown classes. The model is trained end-to-end with all the losses. The indices of the samples are used to distinguish between the samples of the seen and unseen classes which are necessary for both the discriminators

5.0.3 Label Classifier

In the proposed architecture, the label classifier G with parameters θ_g is no more a fully connected layer. Instead, the classifier is a network with multiple binary logistic classifiers equal to the number of categories in the source domain i.e. a binary logistic classifier for each class in the source domain. Hence, the number of such classifiers would be $|\mathcal{C}_s|$. Given an input, the output of the feature extractor $F(\boldsymbol{x})$ is fed to all

the binary logistic classifiers. The loss $\mathcal{L}_L$ is the sum over C_s logistic outputs that are all the known categories from the source. These are named as the source logits. Each of these logistic units feeding into $\mathcal{L}_L$ is a 'One vs Rest' binary classifier for the source data points.

$$\mathcal{L}_L(\theta_f, \theta_g) = -\frac{1}{|\mathcal{X}_s|} \sum_{\boldsymbol{x}\in\mathcal{X}_s, \boldsymbol{y}\in\mathcal{Y}_s} \sum_{c=1}^{\mathcal{C}_s} y_c log(\tilde{y_c}) + (1 - y_c) log(1 - \tilde{y_c}) \tag{5.1}$$

where $y_c \in \{0, 1\}$ is the value in one-hot label encoding for that class. $\tilde{y_c} = G(F(\boldsymbol{x}))$ is the prediction of the c^{th} logistic classifier. $\mathcal{C}_s$ is the total number of source categories and $|\mathcal{X}_s|$ is the total number of samples in the source domain. On the estimated probabilities of all the binary logistic classifiers, a thresholding strategy is performed on all the output predictions. If a sample surpasses the threshold in at least one of the classes, then it is considered to be a sample of the known class. However, if the values of the prediction are lower than the threshold τ in all the binary logistic classifiers then it is considered to be a sample from an 'unknown' class and hence rejected. The following equation [5.2] formulates the label classifier more formally:

$$I_k(x_i^t) = \begin{cases} 1, & \text{if } \sum_{c=1}^{\mathcal{C}_s} I\{p_c(\hat{y}|x_i^t) > \tau\} \geqslant 1 \qquad \forall i \in \{1, \ldots, n_t\}, \\ 0, & otherwise \end{cases} \tag{5.2}$$

where $\hat{y} = G(F(x)$ i.e. $\hat{y}$ is the output prediction from a binary logistic classifier. τ is the threshold set to choose if a given sample has to be retained or rejected. $I_k(x_i^t)$ is the index with values $\{0, 1\}$ and denotes if the sample is retained or rejected. 1 denotes retained and 0 denotes otherwise.

5.0.4 Domain Alignment

To transfer the knowledge from the source domain to the target domain the domain gap between the source and target domain has to be nullified. I follow the same approach as described in DANN Ganin *et al.* (2016) by introducing a domain discriminator. The domain discriminator D with parameters θ_d ensures the features of the source and the target domain are properly aligned. The features $F(\boldsymbol{x})$ from the feature extractor is fed into the domain discriminator and trained using the domain labels. Thus, a domain label d = 0 indicates the samples from the target domain and d = 1 for the source domain. Using the features and the domain labels the domain discriminator D is trained to differentiate the features of the source and the target domain. Thus, the discriminator objective is:

$$\mathcal{L}_D(\theta_d, \theta_f) = -\frac{1}{n_s + n_t} \sum_{x \in \{\mathcal{X}_s \cup \mathcal{X}_t\}} d\log[D(F(\boldsymbol{x}))] + (1-d)(1 - \log[D(F(\boldsymbol{x}))]), \quad (5.3)$$

where d is the domain label and $D(F(\boldsymbol{x}))$ is the sigmoid output of the domain discriminator. The discriminator is trained to minimize the loss $\mathcal{L}_D$ enabling it to differentiate between the source and the target features. Following the approach of DANN, we incorporate a parameterless Gradient Reversal Layer(GRL) between the feature extractor and the domain discriminator. The GRL does not affect the features in the forward pass but reverses the gradient $-\frac{\partial \mathcal{L}_D}{\partial \theta_f}$ from the discriminator into the feature extractor. The intuition is that, while the gradients from the discriminator are useful to distinguish the source and the target domain, a parameter update with the negative gradient forces the features to be indistinguishable by the discriminator implying domain invariant.

5.0.5 Known-Unknown Feature Separator

Similar to the domain discriminator, I propose using another discriminator to distinguish between the features of the known and unknown class. Given the extracted features $F(\boldsymbol{x})$ from the feature extractor F, the Known-Unknown discriminator U ensures the features of Known classes are well separated from the features of the Unknown class. The discriminator acts as a binary classifier and trained to distinguish between the known and unknown class features using their domain labels. Therefore, the extracted features $\mathrm{F}(\boldsymbol{x})$ from F are passed to the label classifier and samples are indexed as a known class or an unknown class. Let $\mathcal{I}_{t,b} = I(f(\mathcal{X}_t), \tau, b)$ be the indices of the target samples for which the network has confident predictions greater than α for mini-batch index b. The Known-Unknown discriminator takes the features $\mathrm{F}(\boldsymbol{x})$ as input and the Known-Unknown indexes from equation 5.2 as labels and learns to differentiate between the features of known and unknown classes.

$$\mathcal{L}_U(\theta_u, \theta_f) = -\frac{1}{n_s + n_t} \sum_{x \in \{\mathcal{X}_s \cup \mathcal{X}_t\}} d\log[D(F(\boldsymbol{x}))] + (1-d)(1 - \log[D(F(\boldsymbol{x}))]), \quad (5.4)$$

where $d \in \{0, 1\}$ corresponds to a sample belonging to a seen class or unseen class. The labels d is generated from the equation 5.2. The discriminator is trained to minimize the loss $\mathcal{L}_U$ enabling it to differentiate between the source and the target features. However, unlike the domain discriminator, the known-unknown discriminator has no Gradient Reversal Layer. Hence, the feature extractor learns to produces the feature that are easily separable.

5.0.6 Entropy Minimization

Existing literature demonstrates that a model trained using only source data tends to be highly confident on the source like samples and less confident on target like samples Vu *et al.* (2019). Besides, when the model is unaware of the target data space

there is a high probability that the decision boundary of the model passes through a high-density region of the target space. Hence, it implies the target data samples are either misclassified or classified correctly with very low confidence. In order to ensure a low-density separation between target classes and to utilize the target data for training, I deploy entropy minimization Grandvalet and Bengio (2005); Long *et al.* (2016); Shu *et al.* (2018). For a given image $\boldsymbol{x}$, the softmax output of the network is $G(F(\boldsymbol{x})) = [f_1(\boldsymbol{x}), \ldots, f_{\mathcal{C}}(\boldsymbol{x})]^\top$, where $f_j(\boldsymbol{x}) = p(y = j|\boldsymbol{x}, \theta_f, \theta_g)$ - the probability that image $\boldsymbol{x}$ belongs to class j. The softmax output $G(F(\boldsymbol{x}))$ is a probability vector whose components sum to 1. When the network has high confidence in its prediction, the output is similar to a one-hot vector where all the components of the probability vector are zeros except for one component. Such a prediction has zero entropy. When the network predicts the input image $\boldsymbol{x}$ with equal probability for all the classes it is trained with, such a prediction has the highest entropy - the network is not confident about the label. By minimizing entropy the model is forced to be confident over its predictions on a given target data sample $\boldsymbol{x}$. In the early stages of training, the pseudo labels generated by the model are not as expected. And forcing the model to be confident on these uncertain pseudo labels can hurt the performance of the model. Hence, to alleviate this effect I use a ramp function to adjust the weight assigned to the entropy loss in the overall objective. The ramp function initially assigns a lower weight but gradually increases the weight assigned to the entropy loss. Also, this is in line with the intuition, as the model training progresses the pseudo labels are more accurate and forcing the model to be confident on these labels would only improve the performance.

The entropy regularization loss is then given by,

$$\mathcal{L}_E(\theta_f, \theta_g) = -\frac{1}{|\mathcal{X}_t|} \sum_{\boldsymbol{x} \in \mathcal{X}_t} \sum_{j=1}^{\mathcal{C}} f_j(\boldsymbol{x}_i) \log f_j(\boldsymbol{x}_i). \tag{5.5}$$

5.0.7 *Final Objective*

The network is trained with a unique objective function that brings together all loss terms discussed in the above sections. The overall objective function brings together the discriminator loss Equation (6.2) which aligns the source and target domains, the known-unknown discriminator loss Equation (5.4), the Entropy minimization loss Equation (6.8) and the logistic loss Equation from the multiple binary classifiers (5.1). The parameters of the network are modified using,

$$\begin{aligned} \{\theta_f^*, \theta_y^*\} &= \underset{\theta_f, \theta_y}{\operatorname{argmin}} \ \big[\mathcal{L}_L + \gamma \mathcal{L}_U + \beta \mathcal{L}_E - \lambda \mathcal{L}_D\big] \quad \text{and} \\ \{\theta_d^*\} &= \underset{\theta_d}{\operatorname{argmin}} \ \big[\lambda \mathcal{L}_D\big], \\ \{\theta_u^*\} &= \underset{\theta_u}{\operatorname{argmin}} \ \big[\lambda \mathcal{L}_U\big], \end{aligned} \tag{5.6}$$

where, γ, β and λ are hyper parameters that controls the weight assigned to each loss term.

During inference, the outputs of the multiple binary logistic classifiers are compared with the threshold τ as described in 5.2. If a sample exceeds the threshold in at least one of the binary classifiers then the sample is retained else it is rejected. In the end, only the samples retained are classified into one of the source categories and the rejected samples are labeled as unknowns.

5.1 Implementation Details

I use the pre-trained Resnet-50 He *et al.* (2016) model from PyTorch Paszke *et al.* (2017) package as the base neural network. I further removed the classifier from the network and added a bottleneck layer of 256 dimensions after the global average pooling layer. Similar to the approaches in Long *et al.* (2017a); Wang *et al.* (2019a); Cao *et al.* (2019, 2018b), I defined a domain discriminator after the bottle-

neck layer. Unlike other approaches, I incorporated multiple binary logistic classifiers instead of a fully connected layer for the classification. The number of binary classifiers defined would depend on the number of categories in the source domain and the domain discriminator is of dimensions 1024-1024-1. In addition, I also add a Known-Unknown discriminator with dimensions the same as domain discriminator. The Known-Unknown discriminator is also defined after the bottle-neck layer and in parallel to classifier and Domain discriminator. As the classifier and the discriminators are trained from scratch I used a learning rate that is 10 times of what is used to fine-tune the feature extractor. I use the same learning rate strategy implemented in Ganin *et al.* (2016): with $\eta_p = \frac{\eta_0}{(1+\alpha p)^\gamma}$, where p is the training progress varying between $[0, 1]$, while η_0, α and γ are optimized with importance-weighted cross-validation Sugiyama *et al.* (2007). I also set the default values provided from Ganin *et al.* (2016) without further fine tuning. To update the weights of the full network, I used mini-batch stochastic gradient descent with Nesterov as the optimizer. Similar to Long *et al.* (2017a); Wang *et al.* (2019a), I added a weight decay of $5e-4$ with momentum = 0.9 in the optimizer. A threshold of $\tau = 0.9$ is used in the binary classifier to index a sample as known or unknown.

5.2 Experiments

The results of the proposed approach are shown in Table 5.1. The initial set of experiments is mostly performed on the office-31 dataset particularly on the task **A** $\rightarrow$ **W**. As it is preliminary work and the experiments are still in the early stage, The results are not compared with the Universal Domain Adaptation work. However, the baseline performances are reported to analyze the gain in the performance using the proposed approach.

We use the Multi Adversarial Generalized Domain Adaptation approach to esti-

Table 5.1: Accuracy of MAGDA on *Office-31* using (ResNet-50). The experiments include using both the discriminators. The experiments listed are the performance on $\mathbf{A} \rightarrow \mathbf{W}$ with a commonness=0.5 between the source and target domain as discussed in equation 5.6.

Method	Accuracy	closed-label acc.	rejection acc.
w/o k-unk discriminator (iter 1000)	82.5	81.3	83.7
w/o k-unk discriminator (iter 5000)	71.3	73.1	69.6
GDA with pseudo labels (iter 1000)	83.5	-	-
GDA with pseudo labels (iter 5000)	81.23	-	-
GDA with correct labels (iter 1000)	86.17	-	-
GDA with correct labels (iter 5000)	83.3	-	-
GDA w/o MSE (iter 1000)	85.1	84.9	85.3
GDA w/o MSE (iter 5000)	80.1	86.9	73.3
GDA & MSE (iter 1000)	87.6	86.3	88.9
GDA & MSE (iter 5000)	81.1	89.3	72.9

mate the performance on the target data.

From Table 5.1, one can interpret the performance of Multi Adversarial Generalized Domain Adaptation. For each method, the performances are compared at two different stages during training. I observed a common trend in all these approaches. As the training progresses, the number of samples that are correctly classified from the shared label space increases. On the other hand, the number of out-of-distribution samples or outliers that are classified as one of the source categories is increasing. Furthermore, the experiments show that the out-of-distribution samples are classified into the source of private classes instead of rejecting them. Thus, with the experiments, I conclude that with a strong weighting mechanism that down weighs the source private classes can certainly help in improving the performance. Regarding rejecting out-of-distribution samples there was no particular trend observed in the

predictions.

5.3 Conclusions

In this chapter, the aim was to propose a practical and challenging domain adaptation scenario called Generalized Domain Adaptation. I further outlined the fundamental challenges in the proposed framework and presented a novel approach Multi-Adversarial Generalized Domain Adaptation. The proposed method was not highly successful in rejecting the outliers, however, it provides an early direction of research towards the Generalized Domain Adaptation. I will further discuss the directions for future research. The features of images from known classes and unknown classes are not well separated due to the absence of priors in the target domain. Hence, using the pseudo labels effectively can act as a proxy for the absent priors. Finally, the private classes of the source domain are hurting the performance of the model in rejecting outliers. Hence, the knowledge of the pseudo labels with an ideal weighing function must be included to assign a higher weight to the classes in the shared label space and also down weigh the classes that are private to the source domain. An architectural design choice or effective use of pseudo labels of the target samples can play a key role in Generalized Domain Adaptation.

Chapter 6

GENERALIZED DOMAIN ADAPTATION WITH GATED SMOOTHING

6.0.1 Introduction

The previous chapters discuss the major constraint behind Unsupervised Domain Adaptation and also outlines the ideas of Partial Domain Adaptation and Open-set Domain Adaptation, variants that are more realistic than UDA. I also discussed the idea of Generalized Domain Adaptation and proposed a model Multi-Adversarial Generalized Domain Adaptation in the previous chapter. However, rejecting outliers a.k.a out-of-distribution sample is a fundamental aspect in Generalized Domain Adaptation. Aligning the features in the shared label space to avoid negative transfer requires one to know the priors on the source and target domains. But it is forbidden in the case of GDA. In other words, aligning the features requires one to reject the outliers so that the model is forced to align with only the in-distribution samples. Similarly, weighing the source classes with the pseudo labels of the target samples and using only the samples of higher weight source classes will enable us to align the domains in the shared space.

Hence, in this chapter, I proposed a novel approach that smooths out the predicted output probability of the classifier on outliers in a probabilistic manner. Hence, a simple thresholding based rejection strategy is sufficient to reject the outliers. The proposed approach is named as Generalized Domain Adaptation with Gated Smoothing(GAMDA) to be consistent with the idea. The proposed model aligns the features of the source and target domains only in their shared label space to avoid any negative transfer from the source domain to the target domain. The proposed work introduces

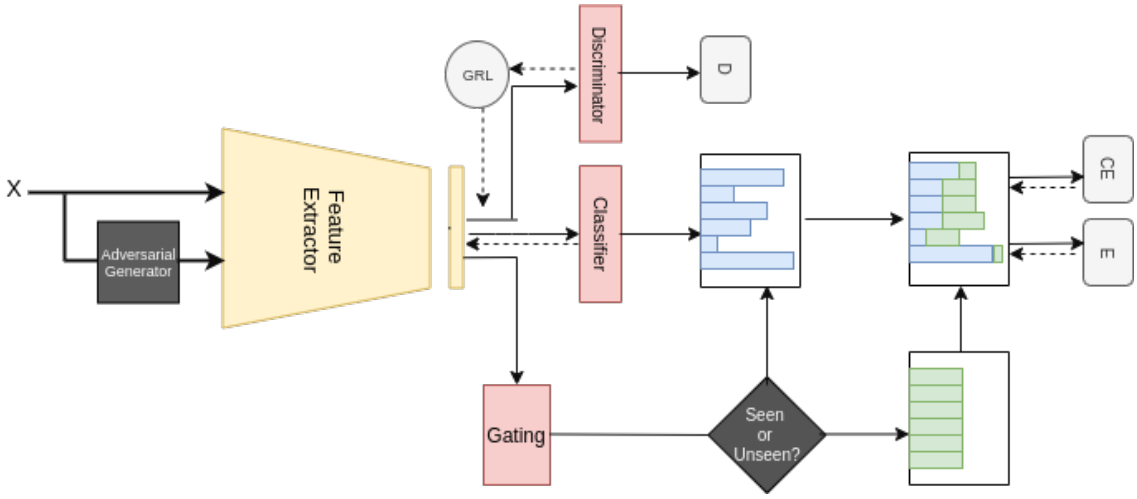


Figure 6.1: Illustration of the Generalized Domain Adaptation model with Gated Smoothing. The Base CNN Network is the feature extractor that takes in both the source and the target images. CE is the cross-entropy loss of the source samples. E is the entropy loss minimized over the target logits. D is the Adversarial Domain Discriminator loss that aligns the source and target features in the common label space. The black box before the feature extractor takes in the clean images and generates adversarial images. Both the clean and adversarial images are used to train gating module. Depending upon the $p(seen)$ or $p(unseen)$ the predictions are smoothed. The model is trained end-to-end with the final objective function.

a gated module to smoothen the predictions on outliers. Finally, a classifier is trained to learn categorical discriminative features that generalize to both the source and target domains and simultaneously reject the outliers. In the following subsections, I describe the architecture design and the objective of the approach in more detail. Subsequently, I show experimental results of the proposed model on the office-31 dataset and finally conclude the chapter with comments and future directions.

6.0.2 Overview

As shown in figure 6.1, the model consists of a feature extractor, domain discriminator, gating module and a label classifier. Let F be the feature extractor with parameters θ_f., an adversarial domain discriminator D with parameters θ_d, a gating

module U with parameters θ_u and a classifier G with parameters θ_g. The model initially takes in the samples from both the source and target domains as input and extracts features from the shared feature extractor F. Let $\boldsymbol{x}_i$ be an input sample that is fed into the feature extractor F and $F(\boldsymbol{x})$ be its corresponding feature vector. The feature vector $F(\boldsymbol{x})$ is fed to the domain discriminator to align the source and target domains in their shared label space. Simultaneously, the label classifier is also fed with the feature vector to output a probability distribution over all the source categories for the given sample. Likewise, the gating module also takes in the feature vector as input and estimates the probability of the sample from a seen class. Using the gating module, the estimated probabilities are applied as a filter on the output predictions of the label classifier. If the probability of seen is high, then the probabilities of the classifier are more sharpened. Similarly, if the probability of the unseen is high, then the outputs are smoothed. Additionally, a thresholding strategy is applied to the normalized outputs to reject the out-of-distribution samples. Finally, the pseudo labels of the target samples are utilized to reweigh all the source classes to minimize the effect of private classes of source domain on the target samples.

6.0.3 Label Classifier

The label classifier G in Generalized Domain Adaptation with Gated Smoothing with parameters θ_g is a linear fully connected layer with the number of outputs equal to the number of categories in the source domain. The feature vector $\boldsymbol{x}$ from the feature extractor is fed into the label classifier to produce a score corresponding to each class for the given sample. I then apply a softmax activation to normalize the output scores to a probability distribution that sums to 1.

The loss $\mathcal{L}_L$ for the normalized outputs of the label classifier is the standard Cross entropy loss used for multi-class classification. The label classifier uses the features of

source data in mini-batches to minimize the cross-entropy loss over the ground truth labels from the source domain. If the source images and their corresponding labels in a mini-batch are represented as $\{\mathcal{X}_s, \mathcal{Y}_s\}$, the cross entropy loss is given by,

$$\mathcal{L}_{CE}(\theta_f, \theta_y) = -\frac{1}{|\mathcal{X}_s|} \sum_{\boldsymbol{x} \in \mathcal{X}_s, y \in \mathcal{Y}_s} \sum_{j=1}^{\mathcal{C}} 1\{y = j\} \log f_j(\boldsymbol{x}), \tag{6.1}$$

where, $1\{\texttt{cond}\}$ is an indicator function which is true if the `cond` is true.

6.0.4 Domain Alignment

To transfer the knowledge from the source domain to the target domain the domain gap between the source and target domains has to be nullified. I take the same approach as described in DANN Ganin *et al.* (2016) by introducing a domain discriminator. The domain discriminator D with parameters θ_d ensures the features of the source and the target domain are properly aligned. The features $F(\boldsymbol{x})$ from the feature extractor is fed into the domain discriminator and trained using the domain labels. Thus, a domain label d = 0 indicates the samples from the target domain and d = 1 for the source domain. Using the features and the domain labels the domain discriminator D is trained to differentiate the features of the source and the target domain. Thus, the discriminator objective is:

$$\mathcal{L}_D(\theta_d, \theta_f) = -\frac{1}{n_s + n_t} \sum_{x \in \{\mathcal{X}_s \cup \mathcal{X}_t\}} d \log[D(F(\boldsymbol{x}))] + (1 - d)(1 - \log[D(F(\boldsymbol{x}))]), \tag{6.2}$$

where d is the domain label and $D(F(\boldsymbol{x}))$ is the sigmoid output of the domain discriminator. The discriminator is trained to minimize the loss $\mathcal{L}_D$ enabling it to differentiate between the source and the target features. Following the approach of DANN, we incorporate a parameterless Gradient Reversal Layer(GRL) between the feature extractor and the domain discriminator. The GRL does not affect the features in the forward pass but reverses the gradient $-\frac{\partial \mathcal{L}_D}{\partial \theta_f}$ from the discriminator into the

feature extractor. The intuition is that, while the gradients from the discriminator are useful to distinguish the source and the target domain, a parameter update with the negative gradient forces the features to be indistinguishable by the discriminator implying domain invariant.

6.0.5 Gating Module

The gating module is one of the key components of the proposed approach that is discussed in the current chapter. The module takes in the feature vector $F(\boldsymbol{x})$ as input and outputs a scalar between 0 and 1 i.e. the probability of the input sample belonging to a seen class. The goal is to train the gating module with examples from both the seen and unseen classes. Hence, when the feature vector of a new test sample is fed to the gating module, it estimates the probability of the sample belonging to the seen classes(shared label space) more reliably. However, training the gating module is a very challenging task because we are unaware of the samples from the unseen classes.

To overcome this problem, I propose to train a gating module with adversarial samples of given mini-batch samples. I hypothesize that, in the absence of data from the unseen classes, one can use the adversarial samples to distinguish between the seen and unseen classes. I consider the original samples in a mini-batch as the samples from the known class and the generated adversarial samples as the samples from the unknown class. For a given mini-batch, I generated an equal number of adversarial samples and then trained the gating module. In the initial set of experiments, I applied the Fast Gradient Signed Method(FGSM) attack on every image in the mini-batch to generate adversarial samples. Finally, in every iteration with a mini-batch of original samples and adversarial samples, I trained the gating module to differentiate

between the samples of seen and unseen classes.

$$x_{adv} = x + \epsilon \; sign(\nabla_x J(x, y)) \tag{6.3}$$

where x is the clean input image, y is the true label for the input image, x_{adv} is the adversarial image corresponding to the input image, $\nabla_x J$ is the gradient of the model's loss function with respect to the original input pixel vector x. ϵ is a tunable hyperparameter that controls the amount of noise to be added to the input image.

With the adversarial samples, I trained the gating module with a logistic loss to distinguish between the in-distribution and the out-of-distribution samples. The assumption being the clean original samples as the in-distribution samples and the adversarial samples as the out-of-distribution samples. Hence, we expect the gating module to estimate the probability of seen or unseen reliably for a given target sample.

$$\mathcal{C}_U(\theta_f, \theta_u) = -\frac{1}{|\mathcal{X}_s|} \sum_{\boldsymbol{x} \in \mathcal{X}_s} \; ylog(\tilde{y}) + (1 - y)log(1 - \tilde{y}) \tag{6.4}$$

where $y \in \{0, 1\}$ is the ground truth label and $\tilde{y} = U(F(x))$ or $U(F(\tilde{x}))$ is the prediction of classifier on original or adversarial images. $y = 1$ if the classifier is trained with a clean original batch of image whereas $y = 0$ if the classifier is trained with adversaries.

6.0.6 Output Smoothing

Output smoothing is one of the key concepts of the proposed approach. With output smoothing, the out-of-distribution samples that are predicted as one of the source categories with very high confidence are flattened out. Thus, with the thresholding strategy, the out-of-distribution samples are rejected more reliably. The smoothing approach is formally defined as follows:

$$p(y|x;\theta) = \sum_{q\in\{s,u\}} p(y,q|x;\theta) \tag{6.5}$$

$$p(y) = p(s)\ p(y|s) + p(u)\ p(y|u) \tag{6.6}$$

where $p(s)$ is the probability of the sample from a seen class. $p(u) = 1 - p(s)$ is the probability of the sample from an unseen class. And, $p(y|s)$ is the estimated probability of the sample assuming it is from a seen class and similarly $p(y|u)$ is to unseen. The probabilities $p(s)$ and $p(u)$ is obtained using the gating module as discussed in the previous section. $p(y|s)$ implied that it belongs to a seen class, which means $p(y|s)$ is a probability distribution over all the source categories because the source categories are the seen classes. The key challenge here is to estimate $p(y|u)$ because the model has no information regarding the unseen classes and estimating a probability distribution over unseen classes is not possible. I propose an alternative solution to estimating a probability distribution over unseen classes. Ideally, if a sample is from an unseen class but passed through the model to estimate the probability over the seen classes then it should be equiprobable over all the source categories because it does not belong to any of the source categories. With this intuition, I define $p(y|u)$ as a uniform distribution with the probability

$$p(y|c_i) = \frac{1}{|\mathcal{C}_s|} \tag{6.7}$$

for each class over all the source categories. where c_i is the i^{th} category in the source domain and $|\mathcal{C}_s|$ is the total number of categories in the source domain.

Hence, If a given sample belongs to any of the source classes, then $p(y|s) > p(y|u)$ implies the softmax predictions are dominant in $p(y)$. However, if the given sample belongs to an unseen class i.e. an out-of-distribution sample then $p(y|u) > p(y|s)$ which means the over-confident softmax predictions are smoothed with the uniform

distribution values. Thus, smoothing avoids the out-of-distribution samples to be classified as a source class with high confidence and assists the model by rejecting them. Therefore, the set threshold can easily reject the samples from the unseen classes.

6.0.7 Entropy Minimization

Existing literature demonstrates that a model trained using only source data tends to be highly confident on the source like samples and less confident on target like samples Vu *et al.* (2019). Besides, when the model is unaware of the target data space there is a high probability that the decision boundary of the model passes through a high-density region of the target space. Hence, it implies the target data samples are either misclassified or classified correctly with very low confidence. In order to ensure a low-density separation between target classes and to utilize the target data for training, I deploy entropy minimization Grandvalet and Bengio (2005); Long *et al.* (2016); Shu *et al.* (2018). For a given image $\boldsymbol{x}$, the softmax output of the network is $G(F(\boldsymbol{x})) = [f_1(\boldsymbol{x}), \ldots, f_{\mathcal{C}}(\boldsymbol{x})]^\top$, where $f_j(\boldsymbol{x}) = p(y = j|\boldsymbol{x}, \theta_f, \theta_g)$ - the probability that image $\boldsymbol{x}$ belongs to class j. The softmax output $G(F(\boldsymbol{x}))$ is a probability vector whose components sum to 1. When the network has high confidence in its prediction, the output is similar to a one-hot vector where all the components of the probability vector are zeros except for one component. Such a prediction has zero entropy. When the network predicts the input image $\boldsymbol{x}$ with equal probability for all the classes it is trained with, such a prediction has the highest entropy - the network is not confident about the label. By minimizing entropy the model is forced to be confident over its predictions on a given target data sample $\boldsymbol{x}$. In the early stages of training, the pseudo labels generated by the model are not as expected. And forcing the model to be confident on these uncertain pseudo labels can hurt the performance of the model.

Hence, to alleviate this effect I use a ramp function to adjust the weight assigned to the entropy loss in the overall objective. The ramp function initially assigns a lower weight but gradually increases the weight assigned to the entropy loss. Also, this is in line with the intuition, as the model training progresses the pseudo labels are more accurate and forcing the model to be confident on these labels would only improve the performance.

The entropy regularization loss is then given by,

$$\mathcal{L}_E(\theta_f, \theta_g) = -\frac{1}{|\mathcal{X}_t|} \sum_{\boldsymbol{x} \in \mathcal{X}_t} \sum_{j=1}^{\mathcal{C}} f_j(\boldsymbol{x}_i) \log f_j(\boldsymbol{x}_i). \tag{6.8}$$

6.0.8 Final Objective

The network is trained with an objective function that brings together multiple loss terms. The overall objective function brings together the discriminator loss Equation (6.2) which aligns the domains, the gating loss Equation (6.6), the Entropy loss Equation (6.8) and the cross-entropy loss Equation (6.1). The parameters of the network are modified using,

$$\{\theta_f^*, \theta_y^*\} = \underset{\theta_f, \theta_y}{\operatorname{argmin}} \ \left[\mathcal{L}_{CE} + \gamma \mathcal{L}_U + \beta \mathcal{L}_E - \lambda \mathcal{L}_D\right] \quad \text{and} \tag{6.9}$$

$$\{\theta_d^*\} = \underset{\theta_d}{\operatorname{argmin}} \ \left[\lambda \mathcal{L}_D\right], \tag{6.10}$$

where, γ, β and λ are hyper parameters that control the importance of individual loss terms.

However, notice that though there is no loss term for the output smoothing, it plays a major role in rejecting the outliers. During, inference the outputs of the label classifier are smoothed using the probabilities estimated from the gating classifier. Finally, the samples which surpass the threshold are selected and classified into one of the source classes. However, the samples which failed to pass the threshold are rejected and labeled 'unknown.'

$$I_k(x_i^t) = \begin{cases} 1, & \text{if } \max\{p_c(\hat{y}|x_i^t)\} > \tau \qquad \forall i \in \{1, \ldots, n_t\}, \\ 0, & otherwise \end{cases} \tag{6.11}$$

where $\hat{y} = G(F(x)$ i.e. $\hat{y}$ is the smoothed output prediction. τ is a threshold hypoerparameter set to choose if a sample has to be retained or rejected. Hence, all the samples that have the value of $I_k(x_i^t) = 0$ are rejected and the remaining samples are classified into one of the source categories.

6.1 Implementation Details

I use the pre-trained Resnet-50 He *et al.* (2016) model from PyTorch Paszke *et al.* (2017) package as the base neural network. I removed the existing classifier from the network and add a bottleneck layer of 256 dimensions after the global average pooling layer. Similar to the approaches in Long *et al.* (2017a); Wang *et al.* (2019a); Cao *et al.* (2019, 2018b), I added a label classifier and domain discriminator after the bottleneck. Both the label classifier and the domain discriminator are fully connected layers with dimensions (1024-1024-1). Additionally, the gating module is also implemented as a 2 layer fully connected neural network with dimensions (256-256-1). We set a threshold $\tau = 0.9$ to reject the out-of-distribution samples. As the classifier and the discriminator are trained from scratch I used 10 times the learning rate that is used to fine-tune the feature extractor. I followed the same learning rate strategy as implemented in Ganin *et al.* (2016): with $\eta_p = \frac{\eta_0}{(1+\alpha p)^\gamma}$, where p is the training progress varying between $[0, 1]$, while η_0, α and γ are optimized with importance-weighted cross-validation Sugiyama *et al.* (2007). I used the default values provided from Ganin *et al.* (2016) without further fine tuning. To update the weights of the full network, I used mini-batch stochastic gradient descent with Nesterov as the optimizer. Similar to Long *et al.* (2017a); Wang *et al.* (2019a), we use a weight decay of $5e-4$

with momentum = 0.9 in the optimizer.

6.2 Experiments

The results of the proposed approaches are shown in Table 6.1. The initial set of experiments is conducted mostly on the office-31 dataset particularly on the task $\mathbf{A} \rightarrow \mathbf{W}$. As it is preliminary work and the results are unsuccessful, I do not compare the results with Universal Domain Adaptation. However, I report the model performance to analyze the results achieved with the proposed technique. I conduct a few experiments on the GDA setting using Generalized Domain Adaptation with Gated Smoothing. Notice that, the core idea of approach relies on the working of the gated module. If the gating module can estimate the $p(seen)$ and $p(unseen)$ reliably the out-of-distribution sample rejection will be reliable. Hence, we conduct our preliminary experiments to analyze the performance of the gating module.

Table 6.1: Accuracy of GDAGS on *Office-31* using (ResNet-50). The experiments include using both the FGSM attack and the one pixel attack for the gating module(GM). The experiments listed are the performance on $\mathbf{A} \rightarrow \mathbf{W}$ with a commonness=0.5 between the source and target domain. The experiment values are the $p(seen)$ on average across all the samples given the inputs to the gating module are out-of-distribution samples

Method	$p(seen)$
GM with FGSM (iter 500)	96.7
GM with FGSM (iter 2500)	92.9
GM with FGSM (iter 5000)	90.2
GM with FGSM (iter 10000)	97.3
GM with One-pixel attack (iter 2000)	95.4

From Table 6.1, it can be observed that the gating module doesn't predict the outlier with higher unseen probabilities. It is because there are no out-of-distribution samples available to train the binary classifier. However, I proposed that using adversarial samples as out-of-distribution samples may be a workaround. My preliminary

experiments show that only a handful of outliers are predicted as out-of-distribution samples but the rest are predicted as the samples from the shared label space.

Hence, in this section, I have outlined the results from my preliminary experiments on the proposed approach. I would like to state that the experiments were neither exceptional nor produced state-of-the-art results, however, these experiments have raised many fundamental questions and meaningful insights that require us to address for solving the problem of Generalized Domain Adaptation.

6.3 Conclusions

I described a probabilistic approach for Generalized Domain Adaptation in this chapter. The proposed method was not very successful in predicting the probability of being seen or unseen for a new target sample. In a real-world scenario, it is often the case of Generalized Domain Adaptation where one does not have any knowledge on the priors. Hence, I firmly believe that the proposed approach can open up many new ideas for solving a very important problem. The current approach discussed in the chapter does not reject the out of distribution samples adequately. Hence, more effort is required on understanding the concept of out-of-distribution and building the classifiers that can classify the in-distribution and the out-of-distribution samples. Also, classifying the out-of-distribution samples as an additional 'other' class or rejecting the samples with lower or equal confidences over the in-distribution classes are ideas worth exploring. Finally, the effect of the private classes from the source distribution on the samples of the target distribution has to be minimized. A clever way of reweighing the source classes such that the shared classes are given higher weights are worth exploring.

Chapter 7

CONCLUSIONS

The dissertation aims to solve the problem of Domain Adaptation by providing effective ways to overcome the domain shift between the training and testing distributions. In the first section, the dissertation outlines the goals and motivation describing it's importance. Then I formally describe the problem statement with mathematical notation and point out the fundamental problems. The dissertation then introduces 3 different approaches, one approach to solve the sub problem of UDA and 2 approaches for GDA. We first introduced the Certain and Consistent Domain Adaptation to provide a solution through a different perspective for the Closed-set Domain Adaption. The dissertation then progresses to solve Generalized Domain Adaptation and outlines two approaches Multi-Adversarial Generalized Domain Adaptation and Generalized Domain Adaptation with Gated Smoothing. The highlight of the dissertation is that it aims to solve the practical problem of Generalized Domain Adaptation. Generalized Domain Adaptation doesn't rely on the assumption of shared label space hence there is no hard constraint that limits it to deploy in the real world.

Similar to many other academic works, the research ideas presented in this thesis ask more questions than it answers. Some of the questions are aligning the marginal distributions alone, the source and target distributions are aligned? How can one distinguish between an in-distribution sample and an out-of-distribution sample? How do you counter the negative transfer when aligning the marginal distributions in the presence of out of distribution samples? Is aligning the marginals instead of joint distributions the right thing? Why do deep neural networks classify anomalies as one of the classes with a very high probability? The datasets used for evaluation do they

contain sufficient variations and are large enough?

Finally, the dissertation concludes by describing some future directions of research. The first and foremost thing in all modern machine learning models is the data. More challenging datasets are required, datasets that contain more images to learn and many domains for transfer tasks are required. Aligning only marginal distributions does not align the source and target domains well, hence ways to align Joint distributions are needed to be explored. However, the target domain is entirely unlabeled which makes it more challenging to align. A good understanding of out-of-distribution(OOD) has to be studied. Approaches to distinguish out-of-distribution samples when there are no available examples of OOD.

BIBLIOGRAPHY

Ben-David, S., J. Blitzer, K. Crammer, A. Kulesza, F. Pereira and J. W. Vaughan, "A theory of learning from different domains", Machine learning **79**, 1-2, 151–175 (2010).

Bhushan Damodaran, B., B. Kellenberger, R. Flamary, D. Tuia and N. Courty, "Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation", in "Proceedings of the ECCV", pp. 447–463 (2018).

Bousmalis, K., N. Silberman, D. Dohan, D. Erhan and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks", in "CVPR", pp. 3722–3731 (2017).

Cao, Z., M. Long, J. Wang and M. I. Jordan, "Partial transfer learning with selective adversarial networks", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 2724–2732 (2018a).

Cao, Z., L. Ma, M. Long and J. Wang, "Partial adversarial domain adaptation", in "Proceedings of the European Conference on Computer Vision (ECCV)", pp. 135–150 (2018b).

Cao, Z., L. Ma, M. Long and J. Wang, "Partial adversarial domain adaptation", in "Proceedings of the ECCV", pp. 135–150 (2018c).

Cao, Z., K. You, M. Long, J. Wang and Q. Yang, "Learning to transfer examples for partial domain adaptation", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 2985–2994 (2019).

Chapelle, O., B. Scholkopf and A. Zien, "Semi-supervised learning (chapelle, o. et al., eds.; 2006)", IEEE Transactions on Neural Networks **20**, 3, 542–542 (2009).

Courty, N., R. Flamary, A. Habrard and A. Rakotomamonjy, "Joint distribution optimal transportation for domain adaptation", in "NeurIPS", pp. 3730–3739 (2017).

Csurka, G., "A comprehensive survey on domain adaptation for visual applications", in "Domain Adaptation in Computer Vision Applications", pp. 1–35 (Springer, 2017).

Deng, W., L. Zheng, Q. Ye, G. Kang, Y. Yang and J. Jiao, "Image-image domain adaptation with preserved self-similarity and domain-dissimilarity for person reidentification", in "The IEEE Conference on CVPR", (2018).

French, G., M. Mackiewicz and M. Fisher, "Self-ensembling for visual domain adaptation", arXiv preprint arXiv:1706.05208 (2017).

Ganin, Y., E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand and V. Lempitsky, "Domain-adversarial training of neural networks", The Journal of Machine Learning Research **17**, 1, 2096–2030 (2016).

Gatys, L. A., A. S. Ecker and M. Bethge, "A neural algorithm of artistic style", arXiv preprint arXiv:1508.06576 (2015).

Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio, "Generative adversarial nets", in "Advances in NeurIPS", pp. 2672–2680 (2014).

Grandvalet, Y. and Y. Bengio, "Semi-supervised learning by entropy minimization", in "Advances in NeurIPS", pp. 529–536 (2005).

He, K., X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition", in "Proceedings of the IEEE conference on CVPR", pp. 770–778 (2016).

Hoffman, J., E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. Efros and T. Darrell, "CyCADA: Cycle-consistent adversarial domain adaptation", in "Proceedings of the 35th ICML", vol. 80, pp. 1989–1998 (2018).

Laine, S. and T. Aila, "Temporal ensembling for semi-supervised learning", arXiv preprint arXiv:1610.02242 (2016).

Li, Y., L. Liu and R. T. Tan, "Certainty-driven consistency loss for semi-supervised learning", arXiv preprint arXiv:1901.05657 (2019).

Liu, H., Z. Cao, M. Long, J. Wang and Q. Yang, "Separate to adapt: Open set domain adaptation via progressive separation", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 2927–2936 (2019).

Long, M., Y. Cao, J. Wang and M. I. Jordan, "Learning transferable features with deep adaptation networks", arXiv preprint arXiv:1502.02791 (2015).

Long, M., Z. Cao, J. Wang and M. I. Jordan, "Domain adaptation with randomized multilinear adversarial networks", CoRR **abs/1705.10667** (2017a).

Long, M., H. Zhu, J. Wang and M. I. Jordan, "Unsupervised domain adaptation with residual transfer networks", in "Advances in NeurIPS", pp. 136–144 (2016).

Long, M., H. Zhu, J. Wang and M. I. Jordan, "Deep transfer learning with joint adaptation networks", in "ICML-Volume 70", pp. 2208–2217 (2017b).

Panareda Busto, P. and J. Gall, "Open set domain adaptation", in "Proceedings of the IEEE International Conference on Computer Vision", pp. 754–763 (2017).

Paszke, A., S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga and A. Lerer, "Automatic differentiation in PyTorch", in "NIPS Autodiff Workshop", (2017).

Pei, Z., Z. Cao, M. Long and J. Wang, "Multi-adversarial domain adaptation", CoRR **abs/1809.02176**, URL `http://arxiv.org/abs/1809.02176` (2018).

Saenko, K., B. Kulis, M. Fritz and T. Darrell, "Adapting visual category models to new domains", in "ECCV", pp. 213–226 (Springer, 2010).

Saito, K., S. Yamamoto, Y. Ushiku and T. Harada, "Open set domain adaptation by backpropagation", in "Proceedings of the ECCV", pp. 153–168 (2018).

Shen, J., Y. Qu, W. Zhang and Y. Yu, "Wasserstein distance guided representation learning for domain adaptation", in "Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, 2018", pp. 4058–4065 (2018).

Shu, R., H. H. Bui, H. Narui and S. Ermon, "A dirt-t approach to unsupervised domain adaptation", arXiv preprint arXiv:1802.08735 (2018).

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting", Journal of Machine Learning Research **15**, 1929–1958 (2014).

Sugiyama, M., M. Krauledat and K.-R. MÃžller, "Covariate shift adaptation by importance weighted cross validation", JMLR **8**, May, 985–1005 (2007).

Tarvainen, A. and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results", in "Advances in NeurIPS", pp. 1195–1204 (2017).

Tzeng, E., J. Hoffman, T. Darrell and K. Saenko, "Simultaneous deep transfer across domains and tasks", in "Proceedings of the IEEE ICCV", pp. 4068–4076 (2015).

Tzeng, E., J. Hoffman, K. Saenko and T. Darrell, "Adversarial discriminative domain adaptation", in "IEEE CVPR", pp. 7167–7176 (2017).

Venkateswara, H., S. Chakraborty and S. Panchanathan, "Deep-learning systems for domain adaptation in computer vision: Learning transferable feature representations", IEEE Signal Processing Magazine **34**, 6, 117–129 (2017a).

Venkateswara, H., J. Eusebio, S. Chakraborty and S. Panchanathan, "Deep hashing network for unsupervised domain adaptation", in "CVPR", pp. 5018–5027 (2017b).

Vu, T.-H., H. Jain, M. Bucher, M. Cord and P. Pérez, "Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation", in "CVPR", pp. 2517–2526 (2019).

Wang, X., L. Li, W. Ye, M. Long and J. Wang, "Transferable attention for domain adaptation", in "AAAI Conference on Artificial Intelligence (AAAI)", (2019a).

Wang, Z., Z. Dai, B. Póczos and J. Carbonell, "Characterizing and avoiding negative transfer", in "Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition", pp. 11293–11302 (2019b).

You, K., M. Long, Z. Cao, J. Wang, and M. Jordan, "Universal domain adaptation", in "The IEEE Conference on CVPR", (2019).

Zagoruyko, S. and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer", arXiv preprint arXiv:1612.03928 (2016).

Zhang, J., Z. Ding, W. Li and P. Ogunbona, "Importance weighted adversarial nets for partial domain adaptation", in "CVPR", pp. 8156–8164 (2018).

APPENDIX A

PERMISSION STATEMENTS FROM CO-AUTHORS

Permission for including co-authored material in this dissertation was obtained from co-authors, Prof. Sethuraman Panchanathan, Dr. Hemanth Venkateswara and Andrew Dudley.