Domain Adaptation for Image Classification and Crowd Counting

by

Shekhor Chanda

A thesis submitted to
The Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements
of the degree of

Master of Science

Department of Computer Science
The University of Manitoba
Winnipeg, Manitoba, Canada
October 2022

Thesis advisor

Yang Wang

Author

Shekhor Chanda

Domain Adaptation for Image Classification and Crowd Counting

Abstract

We consider a problem of domain adaptation in image classification and crowd counting. Given a pre-trained model learned from a source domain, our goal is to adapt this model to a target domain using unlabeled data. The solution of this problem has a lot of potential applications in computer vision research that require a neural network model adapted to a target dataset. In this thesis, we propose two different approaches for domain adaptation. First, inspired by a source free domain adaptation, we propose a black-box model adaptation and distillation for image classification. The key challenge of this problem setting is that we do not have access to any internal information of the source model, including model architecture, model parameters, or even intermediate feature maps. We can only access the output of the source model, hence the source model is a "black-box" to us. Once the model is adapted to the target domain, we perform knowledge distillation to obtain a compact model for deployment. Second, we apply dynamic transfer for solving domain adaptation problems in crowd counting. The key insight is that adapting the model for the target domain is achieved by adapting the model across the data samples. The experimental results on several benchmark datasets demonstrate the effectiveness of our approaches.

Contents

Abstract ii
Table of Contents iv
List of Figures v
List of Tables vii
Acknowledgments ix

1 Introduction **1**
1.1 Contributions 5
1.2 Thesis Organization 6

2 Related Work **7**
2.1 Domain Adaptation 7
2.2 Source-free domain adaptation 8
2.3 Knowledge Distillation 8
2.4 Dynamic Networks 9

3 Black-Box Model Adaptation and Distillation **10**
3.1 Problem Setup 10
3.2 Our Approach 11
3.2.1 Black-Box Adaptation with Side-Tuning 13
3.2.2 Distilling Compact White-box Model 15
3.3 Experiments 19
3.3.1 Datasets 19
3.3.2 Implementation Details 23
3.3.3 Baseline Methods 24
3.3.4 Non-Black-Box Methods 25
3.3.5 Results 26

4 Dynamic Transfer for Domain Adaptation in Crowd Counting **29**
4.1 Problem Setup 29
4.2 Our Approach 31

4.2.1 Network Architecture 32
4.2.2 Supervised Loss: 32
4.2.3 Domain Adaptation Loss 32
4.2.4 Dynamic Transfer 33
4.3 Experiment 34
4.3.1 Datasets 35
4.3.2 Implementation Details 36
4.3.3 Evaluation Metrics 37
4.3.4 Experimental Results 37

5 Conclusion and Future Work **39**

Bibliography **49**

List of Figures

1.1 **Problem illustration.** We consider a new problem called *black-box model adaptation and distillation* in image classification. Consider a vendor (as source domain) providing image classification services to its clients (as target domain). The vendor trains a model using its own data. A client has some unlabeled data. There is a domain shift between the vendor and the client. Due to privacy concerns, the vendor and the client do not want to share data with each other. In addition, due to risk and liability concerns, the vendor only provides a black-box model. The client cannot access the internal information (e.g. model architecture, parameters, intermediate feature maps) of the black-box model. Our goal is to adapt the black-box model to the target domain and distill the adapted model to a compact white-box model for deployment on the client-side. 2

1.2 **Illustration of the dynamic transfer for domain adaptation problem.** The key idea is to minimize domain shift between source and target domain using dynamic neural network. Dynamic transfer($\delta_{\theta(x)}$) learns the model parameter $\theta_{(x)}$ based on per sample of data. 3

3.1 **Overview of our approach.** Given a black-box model θ_{base} trained from the source domain, we use the idea of side-tuning [72] to add a side network θ_{side} to θ_{base} and obtain a target network Θ. We then learn Θ on the target domain while keep θ_{base} frozen, i.e. we only update θ_{side}. We then use θ_{side} to generate pseudo-labels on the target domain using a clustering technique similar to SHOT [37]. Finally, we use the pseudo-labels to distill a compact white-box model ψ from Θ. During the whole process, our approach does not need to access any internal information (e.g. model architecture, parameters, intermediate feature maps) of the black-box model θ_{base}. 12

4.1 **Overview of our approach.** We generate density map of source domain image and calculate supervised loss. Then we extract features from (1x1) convolution layer for both source, and target domain and we compute the mmd loss. Throughout the process, we update the model for combined loss of supervised and mmd loss. 31
4.2 **Architecture of the dynamic crowd counting network.** Static part of the network is replaced by a combination of static kernel matrix and a dynamic residual matrix. 31

List of Tables

3.1 **Summarization of various methods used in the comparison**. The 1st column corresponds to the name of each method (see Sec. 3.3.3 and Sec. 3.3.4 for detailed description of each method). The 2nd and 3rd columns indicate whether a method requires accessing the model parameters or any intermediate feature maps of the base model (✗is desirable). The last column indicates whether the final model of the method is a compact model (✓is desirable). The last four methods can be seen as "oracle" since they require accessing internal information of the base model. "*" indicates the method has some black-box component and not be deployable as a standalone application (e.g. when the black-box component is provided as web API). 18

3.2 **Results of the closed-set setting on the Digits dataset [22]**. We use LeNet architecture for the base network, the side network, and the target network on this dataset. Our approach outperforms other black-box methods. We also show results of non-black-box oracle methods. 19

3.3 **Results of the close-set setting on the Office dataset [54]**. The base model is ResNet101. The side model is ResNet152. The final model is ResNet18. 20

3.4 **Results of the close-set setting on the Office-Home dataset [66]**. The base model is ResNet101. The side model is ResNet152. The final model is ResNet18. 20

3.5 **Results of the close-set setting on the VisDA-C dataset [50]**. The base model is ResNet101. The side model is ResNet152. The target model is ResNet18. 21

3.6 **Results of the partial-set setting on the Office-Home dataset [66]**. The base model is ResNet101. The side model is ResNet152. The target model is ResNet18. 21

3.7 **Results of the open-set setting on the Office-Home dataset [66]**. The base model is ResNet101. The side model is ResNet152. The target model is ResNet18. 22

3.8 **Ablation study on the effect of side model architecture on Office-Home dataset [66].** Results show our approach is robust to the choice of the side network architecture. 22

4.1 **The performance of no adaptation (NoAdapt), domain adaptation(DA) and domain adaptation with dynamic neural network(DRT)** We use WorldExpo10 as the source domain dataset and City, PETS, Mall as target domain datasets 34

4.2 **The performance of no adaptation (NoAdapt), domain adaptation(DA) and domain adaptation with dynamic neural network(DRT) for different source and target dataset.** First we consider Mall as the source domain and ShanghaiTechPartA, ShanghaiTechPartB as the target domain datasets. Then we use UCSD for the source domain and ShanghaiTechPartA as the target domain data. 35

Acknowledgments

First and foremost, I wish to convey my appreciation to my advisor Dr. Yang Wang for his all encouragement and mentoring throughout my master's research. Also, it is an honor to have Dr. Carson Leung and Dr. Pingzhao Hu on my thesis committee and for their valuable suggestions to decorate my thesis perfectly.

I am truly grateful to Dr. Yang Wang for providing me with enough financial assistance and motivation which was the most essential part for completing my thesis. I would like to thank all support staff in the Department of Computer Science for their help.

Last but not the least, I am truly in debt to my family for providing me with enough support and strength throughout my academic endeavor.

Chapter 1

Introduction

In this thesis, we explore domain adaptation problem in image classification and crowd counting. We assume that we have a source domain with labeled images and a target domain with unlabeled images. There might exist a domain shift between these two domains due to the different scene, camera position, environmental condition, etc. Current research in computer vision is drawing a surge of interest to solve this domain shift problem to make computer vision problem more realistic. To be precise, recent domain adaptation approaches are making notable attention to solve domain shift problem without using source dataset which makes the domain adaptive application more flexible. Although, these methods are promising, they have some limitations. One main limitation is that these methods use internal information of the source model [33], [37] and it is very possible to recover source dataset [44] using the internal features or parameters of the source model. On the other hand, in crowd counting, existing domain adaptive approaches use static parameters of the model and it also requires to know the definition of the domain or the domain labels. We have analyzed

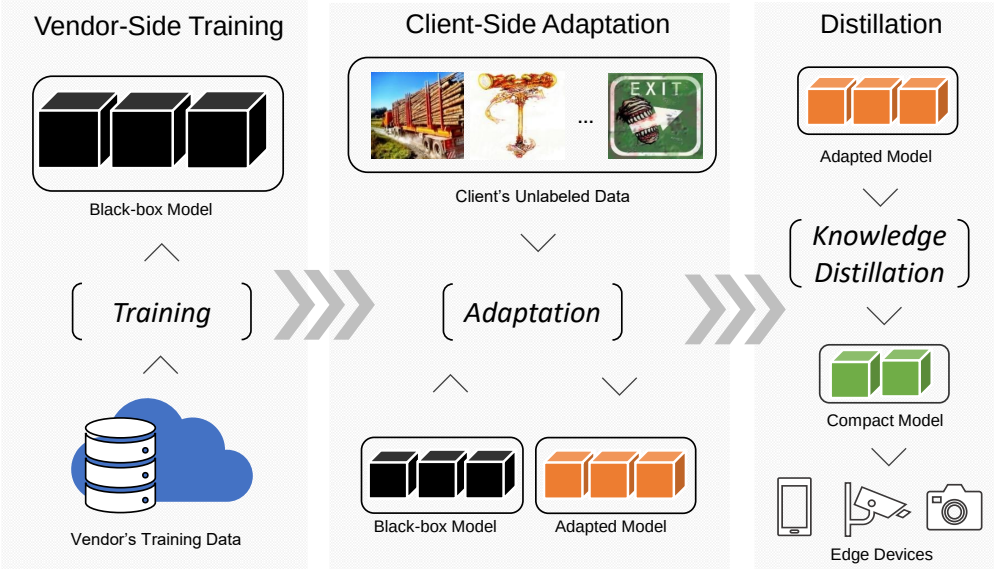

Figure 1.1: **Problem illustration.** We consider a new problem called *black-box model adaptation and distillation* in image classification. Consider a vendor (as source domain) providing image classification services to its clients (as target domain). The vendor trains a model using its own data. A client has some unlabeled data. There is a domain shift between the vendor and the client. Due to privacy concerns, the vendor and the client do not want to share data with each other. In addition, due to risk and liability concerns, the vendor only provides a black-box model. The client cannot access the internal information (e.g. model architecture, parameters, intermediate feature maps) of the black-box model. Our goal is to adapt the black-box model to the target domain and distill the adapted model to a compact white-box model for deployment on the client-side.

these limitations and in this thesis, we consider two different domain application problems, including domain adaptation for black-box model and dynamic transfer for domain adaptation. First, we consider adapting a black-box model to the target domain without accessing any internal information of the model, then we present how dynamic transfer improves the efficiency of adapting a new target domain without having any labeled target data.

For our first problem setup, we assume that we have a black-box model and we

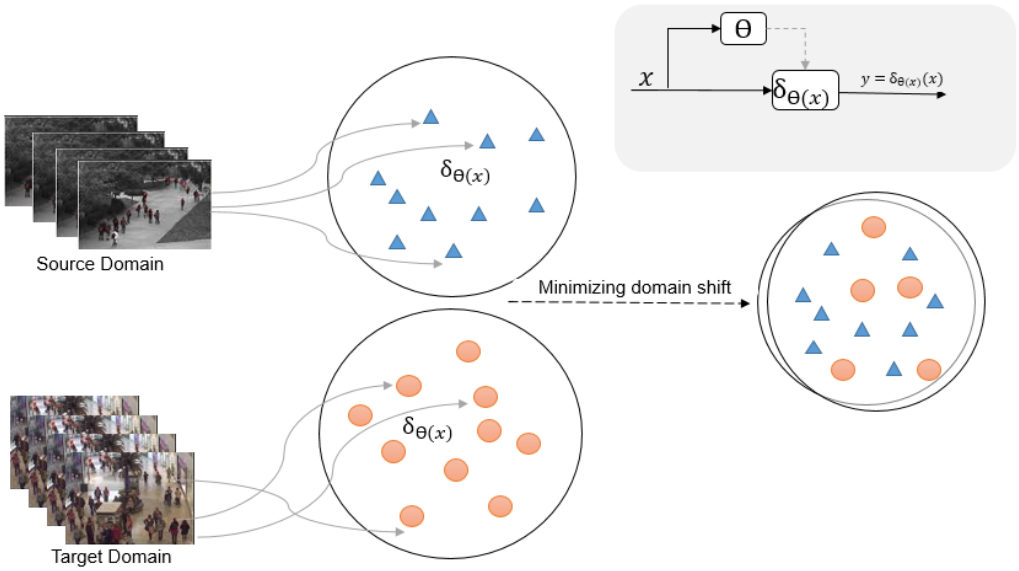

Figure 1.2: **Illustration of the dynamic transfer for domain adaptation problem.** The key idea is to minimize domain shift between source and target domain using dynamic neural network. Dynamic transfer($\delta_{\theta(x)}$) learns the model parameter $\theta_{(x)}$ based on per sample of data.

need to adapt this model to the target domain and possibly distill it to a compact target model for deployment. The problem setting is illustrated in Fig. 1.1. This problem setting is motivated by some challenges arising in real-world applications. Let us consider a hypothetical scenario of a vendor that provides image classification services to its clients. The vendor can train the image recognition model on a large-scale dataset it owns. The trained model can then be used by different clients, e.g. by either sending the model to the client or providing the image recognition functionality as a web API. In this scenario, we can consider the vendor as the source domain and the client as the target domain. Due to privacy issues, the vendor and the client do not want to share data with each other, and data on the client side are likely to be unlabeled. Considering this situation, the vendor might want to provide the trained

model as a black-box without revealing any internal information (e.g. architecture, parameters, even intermediate feature maps) of the model. For example, the black-box model might be provided as a web API that outputs class scores given an image. On the other hand, the client may not have the resources to acquire labeled data or train the model, so it is desirable for the client to take advantage of the pre-trained model from the vendor even if the model is only available as a black-box. But instead of directly using the black-box model, the client might want to do some model adaptation and compression to obtain a compact white-box model for deployment

In this thesis, we propose a two-stage approach to address this black-box adaptation and distillation problem. In the first stage, we use the idea of side-tuning [72] and attach a side network to the black-box base model. We then train this side network using the idea of source hypothesis transfer (SHOT) [37] using unlabeled data in the target domain. Since the base model is a black-box to us, we cannot perform any update on the base model. At the end of the training, the combination of the base model and the side network has been adapted to the target domain. In the second stage, we perform knowledge distillation (KD) to obtain a compact white-box model in the end. Our KD method is designed so that it does not require accessing any internal information of the base model.

In the final part of our thesis, we also present a highly effective domain adaptation approach for crowd counting. The problem setup is illustrated in Fig. 1.2. In this problem, we have crowd images from two different domains namely source and target. Similar to our first problem, images from the source domain are labeled and on the other hand, target domain images are unlabeled. It is very obvious that images from

these two different domains will have domain shift problem due to different camera positions, environmental conditions, and different viewpoints. Our goal is to generate a crowd counting system that will work effectively in the target environment.

Crowd counting problem has a large number of real world applications such as traffic monitoring, crowd-analysis, urban planning, etc that require scene adaptation to work in a target environment. Most of the research is based on a supervised setting, where training and testing data are from the same domain. But this supervised setting is not always applicable for most real-world applications where training and test images are from different domains. Domain adaptive approach [24] solve this problem by taking advantage of the labeled images from the source domain and unlabeled images from the target domain. This method aims to learn a domain agnostic model by optimizing it's static parameters. On the other hand, dynamic neural network adapts a domain by learning the model per sample.

In this thesis, we apply a dynamic neural network for domain adaptation in crowd counting. In this setting, the target domain dataset is also unlabeled. In general domain adaptation adapts the model between source and target domain data. On the other hand, dynamic transfer adapts the model based on per sample of data. It does not need the collection of domain labels or any knowledge over the definition of domains.

1.1 Contributions

We have made several contributions in this thesis. First, we introduce the new problem of black-box model adaptation and distillation, especially in the application

of image classification. We propose an approach for this problem based on source hypothesis transfer (SHOT) [37] and side-tuning [72]. Our proposed method is flexible and does not make any specific assumptions about the model architecture. Second, we apply dynamic transfer with domain adaptation for the crowd counting problem. In dynamic transfer, a neural network model learns to adapt the target domain based on sample of data without having any knowledge over the domain. We also illustrate the performance of our approaches for different applications.

1.2 Thesis Organization

We organize this remaining of the thesis as follows. In Chapter 2, we present the related work of our different approaches. More specifically, we discuss various domain adaptation methods, source-free domain adaptation, knowledge distillation, and different approaches of dynamic network. In Chapter 3, we outline our approach for source-free domain adaptation. We design this approach so that it can work for compact models. We show that without using source data and parameters of the source domain, we can adapt the model for the target domain. In Chapter 4, we present dynamic neural network with domain adaptation for crowd counting. Using this approach, a model can adapt by per sample of images instead of the whole domain. Finally, in Chapter 5, we conclude this thesis.

Chapter 2

Related Work

2.1 Domain Adaptation

There has been lots of prior work on domain adaptation in computer vision. Most of them focus on unsupervised domain adaptation (UDA) [55; 57; 39; 6; 62; 59; 5; 22; 25]. In UDA, the source domain contains labeled data, while the target domain only contains unlabeled data. There is a domain shift between the two domains. The goal is to learn a model that performs well in the target domain. Existing DA approaches often use feature matching or adversarial training to learn domain-agnostic features or models. There has also been work on other settings of UDA. In partial-set DA [9], the label set of the source domain subsumes that of the target domain. In the open-set DA [48], the target domain may contain some unknown classes that never appear in the source domain. The model needs to reject samples of those unknown classes in the target domain.

2.2 Source-free domain adaptation

In standard UDA, the learning algorithm in the target domain can directly access the source domain data. This can be impractical since the source domain data are often not available (e.g. due to privacy). In recent years, there has been work on the more challenging problem of source-free domain adaptation. In source-free DA, the adaptation in the target domain can only access a pretained model from the source domain, with no access to the source domain data. In [37], a source hypothesis transfer (SHOT) framework is proposed for source-free DA. SHOT freezes the classifier of the source model. It then uses information maximization and pseudo-labeling to learn target-specific feature representation. In [32], an instance level weighting mechanism is developed for source-free DA. The method in [31] first performs source-only domain generalization, then source-free target adaptation. Domain impression [33] uses a generative model to generate samples from the source domain for domain adapation. In addition to image classification, there has also been work [4; 40; 61] on source-free DA in image segmentation.

2.3 Knowledge Distillation

The goal of knowledge distillation (KD) is to use a trained large model (called *teacher*) to help the learning of a small model (called *student*). The original KD [21] uses a divergence loss defined on the soft outputs from the teacher and the student. In addition to soft outputs, there have also been KD methods based on various other information, such as intermediate layer features [52], similarity between examples [65],

semantic relations [36], relation knowledge of data sample [49], etc.

Recently, there has been work [42; 47; 19; 70; 12; 45] on data-free knowledge distillation. In data-free KD, we do not have access to the original data used in training the teacher model. Most data-free KD methods try to synthesize training data using the teacher model.

2.4 Dynamic Networks

Some neural network has architectures that depends on blocks [38; 68; 69; 13] or channels [26; 8] that update network parameters based on input samples. [69] increases the size and capacity of the network architecture dynamically based on the input sample by replacing normal convolution with a conditional convolution. [68] presents a technique to execute layers of a deep network based on the input. It reduces the computation time by providing significant prediction accuracy. Changing network architecture dynamically is also applied in [38]. [67; 26] present a feature-based attention module that can update features based on the input samples. In this thesis, we apply dynamic convolution for domain adaptation, that uses a dynamic residual kernel instead of a static kernel.

Chapter 3

Black-Box Model Adaptation and Distillation

In this chapter, we first describe the problem setup for Black-box model adaptation and distillation (Sec. 3.1). We then introduce our proposed approach for K-way image classification (Sec. 3.2), and finally, we provide an overview of our experimental illustration (Sec. 3.3).

3.1 Problem Setup

In this problem, vendor provide a black-box model to client without revealing any extra information. There are many reasons why the vendor wants to only provide a black-box model. First of all, the vendor might consider the network architecture of the model to be a proprietary intellectual property and does not want to reveal the details. Second, it is well known that deep learning models have various security

and privacy vulnerabilities. For example, they can suffer from various attacks such as white-box attack [7], [64], [63], membership inference attack [58], etc. It is also possible to recover images from feature maps [43]. Due to these possible risks and liabilities, the vendor is usually cautious and does not want to reveal extra information about the model. In this thesis, we propose a two-stage approach to address this challenging problem. In the first stage, we use the idea of side-tuning [72] and attach a side network to the black-box base model. We then train this side network using the idea of source hypothesis transfer (SHOT) [37] using unlabeled data in the target domain. Since the base model is a black-box to us, we cannot perform any update on the base model. At the end of the training, the combination of the base model and the side network has been adapted to the target domain. In the second stage, we perform knowledge distillation (KD) to obtain a compact white-box model in the end. Our KD method is designed so that it does not require accessing any internal information of the base model.

3.2 Our Approach

We focus on K-way image classification in this thesis, although our proposed method can also be extended to other problems. We are given a pre-trained black-box model (called *base model*) parameterized by θ_{base} for this classification problem. To simplify the notation, we use the same notation to denote both the function represented by a model and the parameters of the model throughout the thesis. For example, we use θ_{base} to denote parameters of the base model. But we also use $\theta_{base}(\cdot)$ to denote the function represented by the base model. Given an image x,

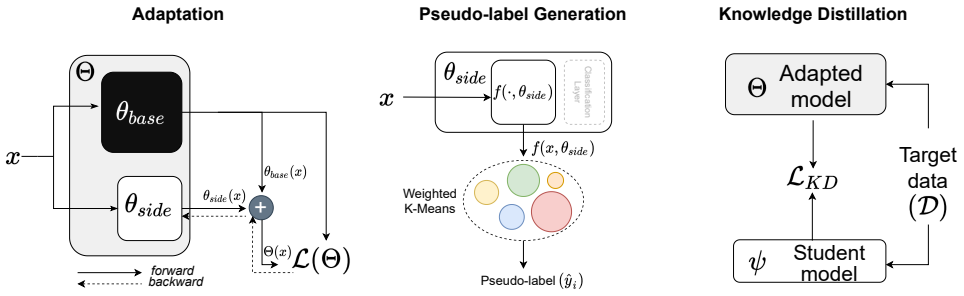

Figure 3.1: **Overview of our approach.** Given a black-box model θ_{base} trained from the source domain, we use the idea of side-tuning [72] to add a side network θ_{side} to θ_{base} and obtain a target network Θ. We then learn Θ on the target domain while keep θ_{base} frozen, i.e. we only update θ_{side}. We then use θ_{side} to generate pseudo-labels on the target domain using a clustering technique similar to SHOT [37]. Finally, we use the pseudo-labels to distill a compact white-box model ψ from Θ. During the whole process, our approach does not need to access any internal information (e.g. model architecture, parameters, intermediate feature maps) of the black-box model θ_{base}.

we use $\theta_{base}(x) \in \mathbb{R}^K$ to denote the predicted class scores before softmax. This base model θ_{base} is trained on some source training data. However, we have no access to the source training data during adaptation. In addition, we assume that θ_{base} is provided as a black-box model, i.e. we cannot examine any internal information of the model, including model architecture, parameters, intermediate feature maps. Given an image x, we only have access to the final prediction $\theta_{base}(x)$. We also have a target domain $\mathcal{D}$ with unlabeled images from a target domain, i.e. $\mathcal{D} = \{x_i\}_{i=1}^N$ where x_i is an unlabeled image in the target domain. There might be a domain shift between the source data used for learning θ_{base} and the target domain. Our goal is to obtain a compact model for the target domain $\mathcal{D}$ by exploiting the black-box base model θ_{base}.

Our approach consists of an adaptation stage (Sec. 3.2.1) and a distillation stage (Sec. 3.2.2). In the adaptation stage, our goal is to create a new model adapted to the target domain. This adapted model can be quite large. In addition, this model has some

black-box components which make it difficult to deploy in certain applications. In the distillation stage, we distill this adapted model to a more compact model. The final model is a compact white-box model that the end-user can easily deploy. An overview of our approach is shown in Fig. 3.1.

3.2.1 Black-Box Adaptation with Side-Tuning

Our adaptation approach is based on the source hypothesis transfer (SHOT) [37] originally proposed for source-free domain adaptation. SHOT has two stages. During the first stage, a target model is initialized from the source model. SHOT then updates the encoder of the target model (while keeping the decoder frozen) by minimizing the information maximization (IM) loss [30; 57; 27] between the source model and the target model using the unlabeled data in the target domain. In the second stage, pseudo-labels are generated on the target data and are used to further fine-tune the target model. Unfortunately, SHOT requires access to the internals of the source model. In SHOT, the target model is initialized by the source model parameters. For the pseudo label generation, SHOT extracts features from the target model which is initialized from the source model. So SHOT cannot be directly used in our setting.

In this section, our goal is to create a target model without accessing the internals of the base model θ_{base}. An important consideration is that we like the target model to have enough capacity so that it can effectively transfer the knowledge from the base model. Unfortunately, since we do not have access to the model architecture of θ_{base}, we cannot easily design the architecture of the target model.

Our solution is to use side-tuning [72] for model adaptation. Side-tuning adapts

a pre-trained base network by fusing it with another "side" network (see Fig. 3.1). During re-training, the base network is frozen and only the parameters of the side network are updated. Let θ_{side} be the side network. We use $\Theta = \theta_{base} \oplus \theta_{side}$ to denote the target network where $\oplus$ denote the fusion of the two networks as shown in Fig. 3.1. Note that since θ_{base} is frozen, Θ is parameterized by only θ_{side}. Given an image x, the output of Θ is the sum of the base network θ_{base} and the side network θ_{side}, i.e. $\Theta(x) = \theta_{base}(x) + \theta_{side}(x)$, where $\Theta(x) \in \mathbb{R}^K$.

We then learn Θ on the target domain data by optimizing the same IM loss used in SHOT [37]. The IM loss consists of an entropy loss L_{env} and a diversity loss L_{div}:

$$L_{IM}(\Theta) = L_{ent}(\Theta) + L_{div}(\Theta), \text{where} \tag{3.1a}$$

$$L_{ent}(\Theta) = \sum_{i=1}^{N} \sum_{k=1}^{K} \delta_k(\Theta(x_i)) \log \delta_k(\Theta(x_i)) \tag{3.1b}$$

$$L_{div}(\Theta) = \sum_{k=1}^{K} \hat{p}_k \log \hat{p}_k \tag{3.1c}$$

where $\delta_k(\cdot)$ denotes the k-th element of softmax output of a K-dimension vector input. Here $\hat{p} = \frac{1}{N} \sum_{i=1}^{N} \delta(\Theta(x_i))$ is the mean output vector of the entire target data and $\hat{p}_k$ is the k-th element of $\hat{p}$. The entropy loss L_{ent} enforces the target outputs to be individually certain. The diversity loss L_{div} makes the target outputs to be globally diverse. Overall, L_{IM} makes the target outputs to be similar to one-hot encodings but differ from each other. Readers are referred to [37] for details on why these kinds of target outputs are desirable.

Similar to [37], we also generate pseudo-labels for the target data. In [37], the pseudo-labeling strategy involves k-means clustering that requires access to the intermediate feature representations of the source model. Since we do not have access

to intermediate features of the source model, we directly use the output of the source model for pseudo-labeling. We call them pseudo-labels generated this way as *base pseudo-labels*. For each sample $x_i \in \mathcal{D}$, its base pseudo-label is obtained from the base model θ_{base} as $\hat{z}_i = \arg\max_k \theta^k_{base}(x_i)$, where $\theta^k_{base}(x_i) \in \mathbb{R}$ is the k-th element of $\theta_{base}(x_i)$ and $\hat{z}_i \in \{1, 2, ..., K\}$.

Our final overall objective for the model adaptation is the combination of the IM loss L_{IM} and the cross-entropy loss using the base pseudo-labels on the target domain:

$$\mathcal{L}(\Theta) = L_{IM}(\Theta) - \lambda \sum_{i=1}^{N} \sum_{k=1}^{K} \mathbb{1}_{[k=\hat{z}_i]} \log \delta_k(\Theta(x_i)) \tag{3.2}$$

where $\mathbb{1}_{[\cdot]}$ is the indicator function and λ is a hyperparameter controlling the relative weight of the two terms in Eq. 3.2

We update Θ by optimizing $\mathcal{L}(\Theta)$ in Eq. 3.2 while fixing the base model θ_{base}. In other words, we only update θ_{side} during this learning process. In the end, we obtain $\Theta = \theta_{base} \oplus \theta_{side}$ as the adapted model for the target domain.

During the whole process of black-box adaptation, we only need the output of the base model. We do not need to access any internal information of the base model.

3.2.2 Distilling Compact White-box Model

The target model Θ can be difficult to deploy in real world applications for two reasons. First of all, the model size of Θ can be quite large since it is a fusion of two separate models (base network θ_{base} and side network θ_{side}). For many real-world applications (e.g. models running on edge devices), we need the deployed model to be small. Second, the base network θ_{base} is a black-box model. In some cases, the

base network θ_{base} is provided as a callable API, and accessing this API might require Internet access. it might be cumbersome for end-users applications to directly call this API. It is desirable to have a compact white-box model for the final deployment. In this section, we use knowledge distillation to obtain a compact white-box model by distilling from Θ.

For knowledge distillation, we need some type of pseudo-labels for the target data. Although we can directly use the base pseudo-labels generated in Sec. 3.2.1, these base pseudo-labels are directly generated from the base model and may not be optimal for the target domain due to domain shift. Instead, we use the technique in [37] to generate refined pseudo-labels (called *target pseudo-labels*) tailored for the target domain using the adapted model Θ. The pseudo-labeling strategy in [37] performs weighted k-mean clustering on the feature vectors on the target data and uses the cluster centroids as the prototypes of each class. In our case, since we do not have access to the feature vectors of the adapted model Θ, we use the feature vectors of the side network θ_{side} as an approximation.

Let $f(\cdot;\theta_{side})$ be the feature extractor of the side network θ_{side}. Given an image x, $f(x;\theta_{side})$ is a feature vector extracted by θ_{side} corresponding to the last layer before the final classification. We perform weighted k-means clustering of the feature vectors in the target domain, where the centroid of each cluster corresponds to a class.

$$c_k = \frac{\sum_{i=1}^{N} \delta_k(\Theta(x_i)) f(x_i;\theta_{side})}{\sum_{i=1}^{N} \delta_k(\Theta(x_i))} \tag{3.3}$$

For each sample x_i, its target pseudo-label of a sample $\hat{y}_i$ can be obtained by the nearest centroid classifier:

$$\hat{y}_i = \arg\min_k D(f(x_i;\theta_{side}), c_k) \tag{3.4}$$

where $D(\cdot)$ is a measure of distance between two vectors. We can then refine the cluster centroids:

$$c'_k = \frac{\sum_{i=1}^{N} \mathbb{1}_{[\hat{y}_i=k]} f(x_i; \theta_{side})}{\sum_{i=1}^{N} \mathbb{1}_{[\hat{y}_i=k]}} \tag{3.5}$$

We can then use the new centroids $\{c'_k\}$ to refine the target pseudo-labels $\{\hat{y}_i\}$. This process can be repeated for several rounds.

In the end, we obtain sufficiently good target pseudo-labels $\{\hat{y}_i\}$ in the target domain. We obtain the compact white-box model using the target pseudo-labels and the knowledge provided by the target model Θ. Let ψ be the final small model. We learn ψ from unlabeled target data $\mathcal{D}$ by optimizing the following objective:

$$\mathcal{L}(\psi) = L_{kd}(\psi) + L_{ent}(\psi), \text{where} \tag{3.6a}$$

$$L_{kd}(\psi) = \sum_{i=1}^{N} KL(\delta(\psi(x_i)/T), \delta(\Theta(x_i)/T)) \tag{3.6b}$$

$$L_{ent}(\psi) = -\sum_{i=1}^{N} \sum_{k=1}^{K} \mathbb{1}_{[\hat{y}_i=k]} \log \delta_k(\psi(x_i)) \tag{3.6c}$$

where $L_{kd}(\cdot)$ is the knowledge distillation (KD) loss and L_{ent} is the standard cross-entropy loss using the target pseudo-labels. The KD loss $L_{kd}(\cdot)$ transfers knowledge from the adapted model Θ to the compact student model ψ. We use the original KD loss in [21] which minimizes the KL divergence between the softmax output of Θ and ψ. The parameter T in L_{kd} is the temperature for producing softer probability distribution for different classes. The loss $L_{ent}(\cdot)$ is a standard cross-entropy loss between the output generated by final model ψ and target pseudo-labels $\{\hat{y}_i\}$ of the target data $\mathcal{D}$.

In the end, we obtain a student model ψ. Compared with Θ, the student model ψ has two important properties. First of all, it has a smaller size. Second, while Θ

Method	Req. Param?	Req. Feat?	Compact?
black-box methods			
BM∗	✗	✗	✗
NL	✗	✗	✓
NL+KD	✗	✗	✓
Adaptation∗	✗	✗	✗
Ours	✗	✗	✓
non-black-box methods			
SHOT	✓	✓	✗
SHOT+KD	✓	✓	✓
SHOT†	✗	✓	✗
SHOT†+KD	✗	✓	✓

Table 3.1: **Summarization of various methods used in the comparison**. The 1st column corresponds to the name of each method (see Sec. 3.3.3 and Sec. 3.3.4 for detailed description of each method). The 2nd and 3rd columns indicate whether a method requires accessing the model parameters or any intermediate feature maps of the base model (✗is desirable). The last column indicates whether the final model of the method is a compact model (✓is desirable). The last four methods can be seen as "oracle" since they require accessing internal information of the base model. "*" indicates the method has some black-box component and not be deployable as a standalone application (e.g. when the black-box component is provided as web API).

has a black-box component (i.e. the base model θ_{base}), ψ is a white-box model. So ψ can be deployed in applications that do not have access to the black-box θ_{base}.

Method	S → M	U → M	M → U	Avg
black-box methods				
BM	71.4	87.3	72.4	77.0
NL	77.9	90.7	74.3	81.0
NL+KD	76.9	90.6	74.4	80.6
Adaptation	91.1	91.2	80.9	87.7
Ours	**93.5**	**92.4**	**82.7**	**89.5**
non-black-box methods				
SHOT	99.1	97.5	98.0	98.2
SHOT+KD	99.0	97.6	97.8	98.1
SHOT†	92.0	94.7	91.9	92.9
SHOT†+KD	90.8	94.8	92.0	92.5

Table 3.2: **Results of the closed-set setting on the Digits dataset [22]**. We use LeNet architecture for the base network, the side network, and the target network on this dataset. Our approach outperforms other black-box methods. We also show results of non-black-box oracle methods.

3.3 Experiments

We evaluate our approach on several benchmark datasets and compare with other alternative methods.

3.3.1 Datasets

We use the following datasets in our experiments. All datasets can be downloaded from the Github repo in [1]. Each dataset consists of multiple domains. On

Method	A → D	A → W	D → A	D → W	W → A	W → D	Avg
black-box methods							
BM	84.7	78.4	62.4	97.2	65.9	99.0	81.3
NL	91.0	84.4	66.9	98.0	69.8	99.4	84.9
NL+KD	90.6	83.7	65.7	97.9	68.8	**99.8**	84.4
Adaptation	91.0	87.3	70.9	**98.4**	72.0	**99.8**	86.6
Ours	**93.0**	**92.1**	**72.2**	97.9	**72.4**	99.4	**87.8**
non-black-box methods							
SHOT	95.0	92.2	76.8	98.9	76.7	99.8	89.9
SHOT+KD	95.2	92.1	76.9	98.9	76.5	99.8	89.9
SHOT†	94.8	92.3	75.4	97.4	75.0	99.6	89.1
SHOT†+KD	92.4	92.2	74.4	97.1	74.9	99.6	88.4

Table 3.3: **Results of the close-set setting on the Office dataset [54]**. The base model is ResNet101. The side model is ResNet152. The final model is ResNet18.

Method	Ar → Cl	Ar → Pr	Ar → Re	Cl → Ar	Cl → Pr	Cl → Re	Pr → Ar	Pr → Cl	Pr → Re	Re → Ar	Re → Cl	Re → Pr	Avg
black-box methods													
BM	52.3	70.9	76.8	58.4	66.7	69.1	56.9	48.6	76.3	70.5	51.9	80.4	64.9
NL	55.2	74.8	79.1	60.3	71.1	73.3	58.4	51.6	78.6	68.7	53.3	82.7	67.3
NL+KD	55.1	73.6	78.9	61.4	70.3	72.0	59.4	51.9	78.0	71.9	54.6	82.2	67.4
Adaptation	57.8	75.6	**81.0**	65.9	73.7	76.3	64.1	54.7	**80.7**	**74.0**	58.4	83.7	70.5
Ours	**59.3**	**77.6**	**81.0**	**68.0**	**75.1**	**77.9**	**65.5**	**56.3**	80.3	**74.0**	**60.0**	**84.0**	**71.6**
non-black-box methods													
SHOT	59.8	78.7	81.8	68.5	77.9	77.1	68.2	59.7	81.9	75.2	62.1	85.0	73.0
SHOT+KD	59.4	78.7	81.9	68.6	78.4	77.6	68.1	58.7	81.9	75.1	62.0	84.9	72.9
SHOT†	58.1	79.9	81.3	65.9	77.6	78.0	65.2	57.4	81.4	73.4	59.2	84.6	71.8
SHOT†+KD	57.8	78.4	81.1	64.4	76.1	76.4	63.6	56.9	80.6	73.3	58.3	84.6	71.0

Table 3.4: **Results of the close-set setting on the Office-Home dataset [66]**. The base model is ResNet101. The side model is ResNet152. The final model is ResNet18.

Method	plane	bicycle	bus	car	horse	knife	mcycl	person	plant	sktbrd	train	truck	Avg
black-box methods													
BM	47.8	21.8	46.6	80.0	65.0	6.4	78.5	31.0	56.8	32.6	82.4	7.6	46.4
NL	58.1	20.6	55.3	81.9	74.5	0.1	**84.6**	39.2	67.3	41.7	**86.4**	4.1	51.2
NL+KD	52.8	23.9	51.3	**82.9**	71.3	5.0	84.0	37.5	64.8	42.3	85.4	5.8	50.6
Adaptation	65.1	64.9	72.3	71.6	84.1	**41.5**	84.4	**66.1**	**84.8**	67.3	85.6	36.1	68.6
Ours	**71.5**	**76.2**	**77.4**	59.1	**88.7**	29.2	83.4	65.3	76.7	**71.3**	81.4	**48.3**	**69.1**
non-black-box methods													
SHOT	94.6	87.3	80.3	56.9	93.9	94.2	82.3	81.2	90.2	90.1	84.7	58.4	82.8
SHOT+KD	94.1	84.5	76.7	55.2	92.9	94.7	79.9	80.7	89.2	87.6	86.4	56.6	81.5
SHOT†	63.4	72.6	75.1	58.8	86.5	7.8	85.3	27.4	76.4	59.6	77.1	45.6	61.3
SHOT†+KD	69.3	76.0	76.1	58.1	87.2	6.3	79.0	30.9	70.6	54.3	77.4	43.5	60.7

Table 3.5: **Results of the close-set setting on the VisDA-C dataset [50]**. The base model is ResNet101. The side model is ResNet152. The target model is ResNet18.

Method	Ar → Cl	Ar → Pr	Ar → Re	Cl → Ar	Cl → Pr	Cl → Re	Pr → Ar	Pr → Cl	Pr → Re	Re → Ar	Re → Cl	Re → Pr	Avg
black-box methods													
BM	54.2	71.0	81.8	61.5	64.0	71.7	64.3	47.9	77.6	73.9	51.6	80.0	66.6
NL	63.6	81.6	89.1	72.4	77.0	83.1	71.8	53.6	85.6	79.4	58.8	85.9	75.2
NL+KD	62.6	78.8	87.5	67.8	74.6	82.2	68.3	52.2	84.1	77.9	58.1	83.3	73.1
Adaptation	**65.1**	83.6	**91.8**	72.5	80.5	**86.3**	**75.0**	55.3	86.8	81.3	**62.0**	85.4	77.1
Ours	64.7	**84.9**	91.3	**73.6**	**81.2**	86.1	74.8	**56.8**	**87.1**	**81.7**	**62.0**	**86.9**	**77.6**
non-black-box methods													
SHOT	70.8	84.8	91.9	78.0	74.4	90.2	81.8	64.8	89.7	84.7	66.4	88.7	80.5
SHOT+KD	70.8	84.8	92.4	77.2	74.3	90.6	81.5	63.9	89.8	84.1	66.2	88.8	80.4
SHOT†	64.5	82.2	90.7	73.7	73.4	87.7	74.8	61.4	86.5	80.9	60.1	89.2	77.1
SHOT†+KD	62.2	79.7	89.2	71.3	72.0	85.6	73.9	59.1	85.4	80.2	59.8	85.9	75.4

Table 3.6: **Results of the partial-set setting on the Office-Home dataset [66]**. The base model is ResNet101. The side model is ResNet152. The target model is ResNet18.

each dataset, we pick one source domain to train the base model θ_{base}, then perform adaptation and distillation on another target domain without using any internal information of the base model.

Method	Ar → Cl	Ar → Pr	Ar → Re	Cl → Ar	Cl → Pr	Cl → Re	Pr → Ar	Pr → Cl	Pr → Re	Re → Ar	Re → Cl	Re → Pr	Avg
black-box methods													
BM	40.0	57.7	67.4	39.0	45.8	54.4	39.5	31.5	61.1	52.2	39.0	64.3	49.3
NL	57.3	75.1	82.6	55.5	67.9	73.8	59.3	49.8	77.5	69.6	54.4	78.4	66.8
NL+KD	58.6	76.3	83.3	58.7	68.0	74.8	62.1	49.9	78.7	72.0	55.6	79.4	68.1
Adaptation	63.7	78.8	85.9	65.1	72.0	79.4	68.6	57.1	82.5	**77.6**	62.2	83.1	73.0
Ours	**65.5**	**80.3**	**86.5**	**66.7**	**74.1**	**80.7**	**69.9**	**57.9**	**82.9**	77.0	**63.5**	**83.6**	**74.1**
non-black-box methods													
SHOT	66.1	81.3	86.0	67.4	78.4	83.5	67.7	61.1	84.0	74.1	67.4	83.6	75.1
SHOT+KD	65.1	81.2	85.1	67.5	78.1	83.3	67.6	60.7	83.3	74.5	66.5	83.6	74.7
SHOT†	57.7	78.2	85.2	58.9	72.8	80.6	59.7	53.6	81.2	69.2	56.5	80.3	69.5
SHOT†+KD	58.9	78.2	84.0	61.9	72.7	79.3	60.8	53.1	80.8	69.9	58.5	81.6	70.0

Table 3.7: **Results of the open-set setting on the Office-Home dataset [66]**. The base model is ResNet101. The side model is ResNet152. The target model is ResNet18.

Model	VGG16	MobileNetV2	ResNet152
NL+KD	67.4	67.4	67.4
Adaptation	**69.6**	69.3	70.5
Ours	69.3	**69.7**	**71.6**

Table 3.8: **Ablation study on the effect of side model architecture on Office-Home dataset [66]**. Results show our approach is robust to the choice of the side network architecture.

- *Digits* [22] is a standard digit recognition DA benchmark. For our method, we utilize three domains: SVHN (S), MNIST (M), and USPS (U). Following [41; 37] we consider one domain for training our model and use another domain for testing and evaluation.

- *Office* [54] is a standard DA benchmark. It consists of three different domains, Amazon (A), DSLR(D), and Webcam(W). Each domain contains 31 classes of objects.

- *Office-Home* [66] is another challenging DA benchmark. It consists of several domains: Artistic Images(Ar), Clip Art(CI), Product Images(Pr), Real-world Images(Rw). Each domain contains 65 object classes.

- *VisDA-C* [50] is the most challenging large scale benchmark for DA task which contains 12 classes of synthetic and real objects. This dataset consists of two domains. The source domain has ∼152,000 synthetic images formed by rendering 3d models. The target domain contains ∼55,000 real images.

3.3.2 Implementation Details

Network architecture. For the Digits dataset, we use LeNet [34] for base, side, and final networks. For other datasets, we use ResNet101 [20] as the base model θ_{base} and ResNet152 as the side network θ_{side}. However, we also report the results with other network architectures in the ablation study. We intentionally choose different architectures for the θ_{base} and θ_{side} because in practice, we do not know architecture of θ_{base}. Therefore, θ_{side} could have different architecture from θ_{base}. We use ResNet18 as the student model ψ. Following [17] and [37] we remove the last FC layer and add a bottleneck layer with 256 units. We also append a task specific FC layer after the bottleneck layer [37]. In addition, a BN layer is added inside the bottleneck layer and also we utilize a weight normalization layer in the last FC layer [37]. On each dataset, we train a base model by minimizing cross-entropy loss with label smoothing technique [46] by following [37]. The parameters of the side network θ_{side} and the target small model ψ are initialized by pre-trained ImageNet models [53].

Network hyper-parameters. We follow the setting in [37] to train our network.

We use mini-batch SGD with momentum of 0.9, weight decay of 1e-3. Following [37], learning rate scheduler is used similar to [17; 41]. For KD, we set temperature hyperparameter $T = 1$. We implement our approach in PyTorch based on several open-source projects [1; 2; 3]. We will release our code upon publication.

3.3.3 Baseline Methods

Since this thesis addresses a new problem, there is no previous work that we can directly compare with. Nevertheless, we define several baseline methods for comparison.

Base model (BM). In this baseline, we directly use the base model θ_{base} as the final model without any adaptation. This baseline technically does not require accessing any internal information of θ_{base}. The disadvantage is that the model itself is not compact (e.g. θ_{base} can be ResNet101 in our case). In addition, if the base model is provided via a web API, this baseline requires Internet access and cannot be deployed as a standalone application.

Naive labeling (NL). In this baseline, we directly use the base model θ_{base} (black-box ResNet101) to make predictions on the target dataset and treat those predictions as the ground-truth labels. We then directly learn the small model ψ from the target dataset using these generated "ground-truth" labels as the final model. The standard cross-entropy loss is used for learning ψ.

Naive labeling with knowledge distillation (NL+KD). This baseline is similar to the previous baseline (NL). The difference is that in addition to the cross-entropy loss, we also add a knowledge distillation loss between the base model (e.g. black-box

ResNet101) and the target small model (e.g. ResNet18).

Adaptation. This is our proposed approach without the KD stage, i.e. it only performs the black-box domain adaptation with side-tuning.

Similar to our method, these fours baselines only need the predicted class scores of θ_{base} and do not require access to an internal information of θ_{base}. For "NL" and "NL+KD", the final target model is compact (e.g. ResNet18). But for "Adaptation", the final model is large (e.g. ResNet101 for θ_{base} and ResNet152 for θ_{side}).

3.3.4 Non-Black-Box Methods

We also consider the following non-black-box methods. These methods have access to various internal information (e.g. model architecture and parameters, feature maps, etc) of the base model. Since these approaches use extra information, they can be seen as "oracle" and provide upper bounds of our method.

SHOT. This method is the source hypothesis transfer (SHOT) in [37] for unlabeled domain adaptation. This approach requires access to all the internal information of the base model θ_{base} including the model parameters. From the privacy viewpoint, this is the least desirable approach. In addition, the final model of this approach is not compact since the same architecture as the base model (e.g. ResNet101).

SHOT with knowledge distillation (SHOT+KD). This approach is similar to SHOT. The difference is that after adapting the base model (ResNet101) to the target domain, we perform an additional step of knowledge distillation to get a compact model (ResNet18). Similar to SHOT, this approach requires access to the parameters of the base model, so it is not desirable from the privacy point of view.

SHOT with ImageNet pre-initialization (SHOT†). The reason that "SHOT" requires access to the model parameters of θ_{base} is that it uses the base model to initialize the parameters of the target model. In this baseline, we initialize the target model using a public pre-trained model (e.g. ImageNet pre-trained). This baseline removes the requirement on the model parameters on the base model θ_{base} in SHOT. Nevertheless, "SHOT+IM" still requires access to the feature vectors of the base model for generating pseudo-labels in the k-means clustering. The final model of this approach has the same architecture of θ_{base} (ResNet101) and is not compact.

SHOT† with knowledge distillation (SHOT†+KD). This is similar to "SHOT†". But it adds a knowledge distillation in the end to get a compact model (ResNet18).

Table 3.1 summarizes the various approaches in terms of the information of θ_{base} they need to access and whether the final model is compact.

In Table 3.2 3.3 3.4 3.5 3.6 3.7 in the first column, we present the methods that we used in our experiments. In all other column except the last column, we record the accuracy for all methods in different dataset setup. For example, in the office dataset, we have images from three different domains, Amazon (A), DSLR(D), and Webcam(W). So for Table 3.3, in the 2nd, and 3rd columns, we record the accuracy of different methods by considering Amazon(A) as the source domain and DSLR(D), and Webcam(W) as the target domain. Again, in the 4th and 5th columns, we record the accuracy by taking DSLR(D) as the source domain and Amazon (A), and Webcam(W) as the target domain. Then, in the 6th and 7th columns, we present the results for all methods by Webcam(W) as the source domain and DSLR(D), and Amazon (A) as the target domain dataset. Finally, in the last column, we provide the

average accuracy of all methods. Following the same setup, we present the accuracy for digit and office-home dataset in Table 3.2 3.3 3.4 3.6 3.7. For the Visda-C dataset, we consider the synthetic dataset as the source domain and the real object as the target domain and we present the accuracy for different methods in Table 3.5.

3.3.5 Results

Following [37], we consider experimental results on several different settings.

Closed-set setting. First, we consider the closed-set setting. This is the most popular setting in domain adaptation. In this setting, the source domain and the target domain have the same label space. We present our results on four benchmark datasets, including Digits (Table 3.2), Office (Table 3.3), Office-Home (Table 3.4) and VisDA-C (Table 3.5). On each dataset, we compare our method with other black-box method baselines. We also compare with some non-black-box methods.

We can make several observations from the results. First of all, directly using the base model (i.e. "BM") gives the worse performance. Second, non-black-box methods generally perform better than black-box ones. This is natural since non-black-box methods have access to more information (e.g. model parameters, intermediate features, etc) about the base model. Finally, among all black-box methods, our proposed method ("Ours") consistently performs the best on all datasets. In particular, it even performs better than "Adaptation" which is a non-compact model. We believe this is because the pseudo-labeling in our KD stage is obtained from the model adapted to the target domain, so the pseudo-labels have higher qualities than the ones used in the black-box adaptation stage.

Partial-set setting. Following [37], we also consider a partial-set setting. In this setting, the label space of the source domain subsumes that of the target domain. Similar to [37], we drop L_{div} from Eq. 3.1 for this setting.

For the pseudo labeling technique, we usually take K number of centroids. For the partial-set setting, there are some centroids which make k-means clustering empty. Similar to [37], we remove a centroid if the corresponding cluster size is less than 10.

Table 3.6 shows the results of this partial-set setting on the Office-Home dataset. We observe a similar trend as the closed-set setting. Our proposed method achieves the best performance among black-box methods.

Open-set setting. In the open-set setting, the target domain consists of some known classes seen in the source domain and some unknown classes. We utilize confidence thresholding technique to remove unknown classes [37] in the learning process for our adaptation and KD stages. Using K-means, we cluster unknown classes with higher mean uncertainty and reject them in the learning process. For measuring the performance of our model, we calculate accuracy for the known and unknown classes.

Table 3.7 shows the results of the open-set setting on the Office-Home dataset. Again, our proposed method outperforms other black-box baselines.

Ablation study. So far, we have used ResNet152 as the side network. An interesting question is whether our proposed approach can work with other network architectures. In Table 3.8, we show the results of using different architectures for the side network, such as VGG16 [60], MobileNetV2 [56]. We observe that different side network architectures give slightly different final results. But overall, our approach outperforms the black-box baseline ("NL+KD") regardless of the side net-

work architecture. This demonstrates that our method is fairly robust to the choice of the side network architecture.

Chapter 4

Dynamic Transfer for Domain Adaptation in Crowd Counting

In this chapter, we provide the problem setup (Sec. 4.1) for the crowd counting problem. Then we provide our domain adaptive approaches (Sec. 4.2) and lastly we present our experimental setup for crowd counting (Sec. 4.3).

4.1 Problem Setup

We consider to apply dynamic transfer for crowd counting problem. Crowd counting is a technique to count or estimate the number of people in an image. The popularity of this technique is growing significantly because of its usage in real-life applications such as urban planning, traffic monitoring, public safety, etc. There are a lot of approaches are available to solve the crowd counting problem. We apply a CNN-based density estimation method to solve this problem. This method works

by learning a linear mapping between features from images and their corresponding density maps. In this approach, the input is an image containing crowd heads and the output is a density map. This density map is also an image label that can estimate the number of people by processing the coordinate value of heads through a Gaussian Convolution. Particularly we extract the image features with the dimension of $H/8 \times W/8 \times 1024$ using Resnet-101 [20] as a backbone architecture. After that, we generate density map from the image features using a second sub-network which is comprised of dilated convolutional layers.

We assume that we have a source domain D_s and a target domain D_t for the crowd counting problem. Images from these two domains might have many differences such as viewing angles, illumination conditions, scene objects, etc. Due to this domain shift problem, a crowd-counting system trained in the source domain may not work in the target domain. There is a lot of existing work has been explored to solve this problem by utilizing different domain adaptation algorithms. In this thesis, we apply dynamic transfer [35] with domain adaptation to solve domain shift problems in crowd-counting. Dynamic transfer adapts the model per sample and each domain consists of a distribution of multiple image samples. In general, domain adaptation works for static network and it defines losses that pull all domains into shared latent space. In dynamic transfer, it is not necessary to pull all domains into a shared space. Model adaptation with dynamic transfer can be easily adapted for the target domain if the target domain is shifted to space formed by the entire source domain. Domain adaptation with static model has a fixed parameter for adapting the target domain, on the other hand dynamic transfer with dynamic model adapts model parameters

according to per sample.

4.2 Our Approach

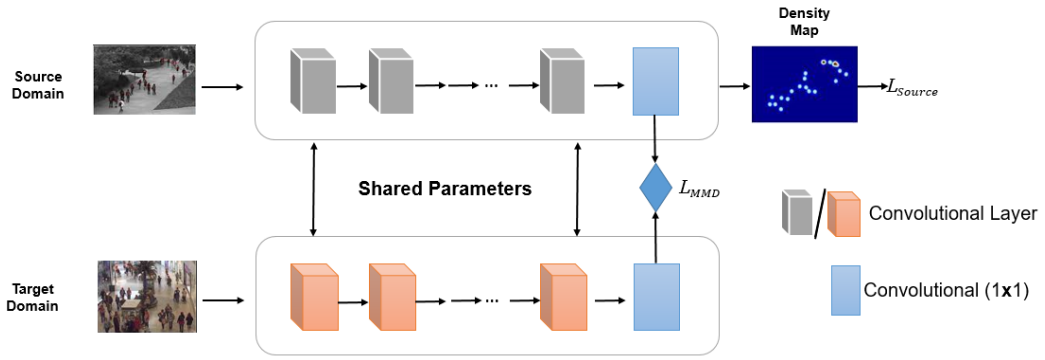

Figure 4.1: **Overview of our approach.** We generate density map of source domain image and calculate supervised loss. Then we extract features from (1x1) convolution layer for both source, and target domain and we compute the mmd loss. Throughout the process, we update the model for combined loss of supervised and mmd loss.

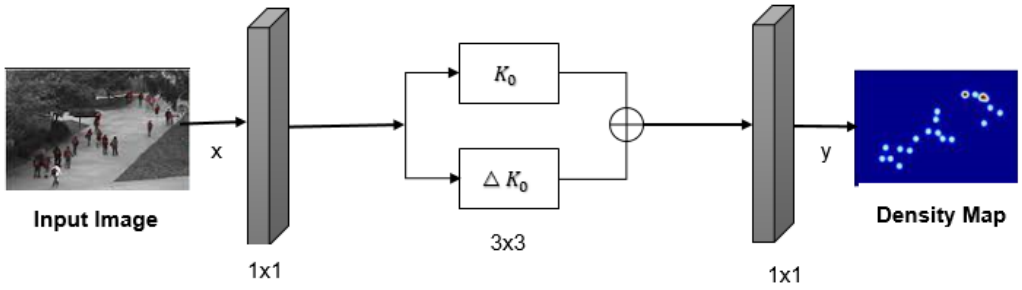

Figure 4.2: **Architecture of the dynamic crowd counting network.** Static part of the network is replaced by a combination of static kernel matrix and a dynamic residual matrix.

4.2.1 Network Architecture

Although this problem is fit to any backbone architecture, we use Resnet-101 [20] as the backbone of our network. For the dynamic transfer, we added the dynamic residual on the 3×3 kernel of each bottleneck block. We use θ to denote model parameter. For example, for an input image x, we denote $\delta_\theta(x)$ to extract features and $\phi(\delta_\theta(x))$ to get the final density map of the image. We update the parameter of $\delta_\theta(x)$ based on supervised loss for source domain and domain adaptation loss. The overview of our approach is shown in Fig. 4.1.

4.2.2 Supervised Loss:

This loss function calculates the performance of the model on the labeled source domain D_s. We denotes this loss as follows:

$$\mathcal{L}_{source} = \frac{1}{N_s} \sum_{i=1}^{N_s} ||\delta_\theta({x^s}_i - {y^s}_i)||^2{}_F \tag{4.1}$$

Here, ${y_i}^s$ denotes the ground truth of x_i image and $||.||_F$ refers the Frobenius norm.

4.2.3 Domain Adaptation Loss

We optimize our network parameter based on the domain adaptation loss that measures the distance of the feature space between the source and target domain. We use MMD [18] as the domain adaptation algorithm following the implementation of [24]. The loss function is represented as below:

$$\mathcal{L}_{UDA} = \mathcal{L}_{source} + \alpha \mathcal{L}_{mmd} \tag{4.2}$$

Here, $\mathcal{L}_{source}$ is the supervised loss defined in Eq. 4.1. The last term $\mathcal{L}_{mmd}$ is for calculating domain adaptation loss. The main objective of this loss function is to build an embedding bringing the feature of source and target data to a Reproducing Kernel Hilbert Space (RKHS). After that, it computes the distance between the mean embeddings. The equation for calculating MMD loss is represented by the following:

$$\mathcal{L}_{mmd}(\theta) = || \sum_{i=1}^{N_s} \frac{\Pi({\delta_i}^s)}{N_s} - \sum_{k=1}^{N_t} \frac{\Pi({\delta_k}^t)}{N_t} ||^2 \tag{4.3}$$

Where $\Pi(.)$ refers the mapping between each data point and RKHS.

4.2.4 Dynamic Transfer

Previous domain adaptation learns a static model by setting a fixed mapping across the domains. On the other hand, dynamic transfer with domain adaptation adapts model per sample basis which increases the effectiveness of the model. For this reason, inspired by [35], we use a model composed of a static model and dynamic residual blocks as follows:

$$\delta_\theta(x) = \delta_0 + \triangle\delta_\theta(x) \tag{4.4}$$

where δ_0 is a static part of the network and $\triangle\delta_\theta(x)$ is a dynamic residual component of the network. Dynamic transfer is implemented by applying Eq. 4.4 to each convolutional kernel of the crowd counting network. Network convolution is represented as follows:

$$K(x) = K_0 + \triangle K(x) \tag{4.5}$$

Where, K_0 is a static network kernel matrix and $\triangle K(x)$ is dynamic residual kernel matrix. The dynamic residual component is implemented by adding residual blocks to

Method	City		PETS		Mall	
	MAE	MSE	MAE	MSE	MAE	MSE
No Adapt	26.6	36.8	19.3	22.2	8.5	9.2
DA	20.5	27.8	14.7	17.3	3.2	4.0
DA + DRT	16.3	23.5	11.1	13.8	3.1	3.9

Table 4.1: **The performance of no adaptation (NoAdapt), domain adaptation(DA) and domain adaptation with dynamic neural network(DRT)** We use WorldExpo10 as the source domain dataset and City, PETS, Mall as target domain datasets

the different network layers and this part is directly dependent on the input sample x. Dynamic transfer helps to train the model according to the sample of images instead of the domain. As every domain is a group of image samples, so adapting a dynamic model to a domain is achieved by adapting the model per sample. As long as the target domain is related to the distribution of source domain, the model can easily learn the target domain. The architecture of the dynamic neural network is illustrated in Fig. 4.2

4.3 Experiment

We evaluate our approach on several crowd-counting datasets and compare the results with some baseline methods.

Method	Source	Target	MAE	MSE
No Adapt	Mall	ShanghaiTechPartA	214.4	336.7
DA	Mall	ShanghaiTechPartA	189.9	336.2
DA + DRT	Mall	ShanghaiTechPartA	186.1	329.8
No Adapt	UCSD	ShanghaiTechPartA	217.8	313.6
DA	UCSD	ShanghaiTechPartA	207.0	287.8
DA + DRT	UCSD	ShanghaiTechPartA	195.1	281.4
No Adapt	Mall	ShanghaiTechPartB	73.9	88.5
DA	Mall	ShanghaiTechPartB	64.1	69.7
DA + DRT	Mall	ShanghaiTechPartB	57.2	65.8

Table 4.2: **The performance of no adaptation (NoAdapt), domain adaptation(DA) and domain adaptation with dynamic neural network(DRT) for different source and target dataset.** First we consider Mall as the source domain and ShanghaiTechPartA, ShanghaiTechPartB as the target domain datasets. Then we use UCSD for the source domain and ShanghaiTechPartA as the target domain data.

4.3.1 Datasets

We use different crowd-counting datasets for evaluating the performance of our approach. For the simplicity of our problem, we consider one dataset as the source domain and another dataset as the target domain.

WorldExpo'10 [71]: At first we use WorldExpo'10 as a source domain data. It is a diverse dataset with multiple scenes. Particularly this dataset consists of 3980 labeled images from 108 different scenes. We keep the resolution of the images at 576 $\times$ 720.

Mall [11]: We also use Mall dataset which consists of 800 training images and 1200 test images. Image resolution is fixed at 640 × 480.

ShanghaiTech Part A [73]: This dataset consists of dense crowd images with a standard split of 300 training images and 182 test images.

UCSD[10]: This dataset has relatively sparse crowd with ranging from 11 to 46 persons in an image. It has a total of 2000 images from one surveillance camera.

PETS [16]: Crowd images of this dataset is collected from multiple views. We consider each view as one scene. The training set of this dataset contains 1105 images and the test set has 794 images. The image resolution is set at 288 × 384.

FDST [15]: This dataset has 13 different camera scenes. The training set contains 9000 image frames from 60 videos and the test set has 6000 frames from 40 videos. We use the resolution of the images as 1920 × 1080.

City [28]: This dataset is comprised of images from 55 different camera scenes. We consider 43 scenes as the training set and 12 scenes as the test set.

4.3.2 Implementation Details

We implement this crowd-counting problem considering Resnet-101 [20] as a backbone network. This network is initialized with ImageNet [14] pre-trained weights. At first, we train the network with only the source domain dataset by optimizing $\mathcal{L}_{source}$ defined in Eq. 4.1. After that, we fix the parameter of the backbone network except for the density map estimator network and a conv layer. Then, we extract features from the conv layer and calculate mmd loss for these features. We use Adam [29] optimizer with an initial learning rate 1e-5 to train the network and the learning rate

is decayed by .995 after every 5 epochs. We train the network for total 210 epochs and a batch size of 1 for all datasets.

4.3.3 Evaluation Metrics

We use the standard evaluation metric namely Mean Absolute Error(MAE) and Mean Square Error(MSE) for evaluating the performance of our method. We define the metris as follows:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i|, MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \tag{4.6}$$

Where, $\hat{y}_i$ is predicted density map and y_i denotes the provided ground truth of the i-th image.

4.3.4 Experimental Results

In this thesis, we present how a dynamic neural network with domain adaptation improves the performance of a crowd-counting model. A crowd-counting model works well when the training and test images are from the same domain. But when we train a model for one domain and test it using images from a different domain, there is a significant performance drop because of the domain shift between training and test data. To solve this problem, we apply domain adaptation with the dynamic neural network. Table 4.1 shows the better performance of the dynamic neural network with domain adaptation over two other benchmark methods. In the 1st column, we present the different methods that we use in our experiments. Then, in other columns, we consider a target domain dataset and record the scores for evaluation metrics MAE

and MSE. We consider WorldExpo'10 as the source domain and City, PETS, and Mall as the target domain dataset. At first, we experiment without any adaptation named as NoAdapt. For this method, we train the crowd-counting method using a dataset and directly test it using a different dataset. Then, we use the domain adaptation (DA) algorithm with labeled images from the source domain and some unlabeled data from the target domain. Finally, we apply dynamic neural network with domain adaptation mentioned in Table 4.1 and Table 4.2 as DA + DRT. We also present the results in Table 4.2. Where we consider Mall and UCSD as the source domain datasets and we use ShanghaiTech Part A and ShanghaiTech Part B as the target domain datasets. From the observations of the results from Table 4.1 and Table 4.2, domain adaptation performs better than without adaptation. This is because we are using the target domain dataset to adapt the model for both the source and target domain. Finally, our proposed approach dynamic neural network with domain adaptation outperforms all other methods.

Chapter 5

Conclusion and Future Work

In this thesis, we present effective domain adaptation approaches for image categorization and crowd-counting problem. First, we address the problem of black-box model adaptation for a compact model. In this problem, we assume that a black-box model is provided by the client. We can not access any internal information of the model. We provide an approach that can adapt data from a new target domain without using source data or any internal information of the base model. Our final target model is also a compact model that can be easily deployed to lightweight devices. Second, we apply dynamic transfer for domain adaptation in the crowd counting problem. Our goal is to produce a crowd counting model that can effectively work on the target domain data. In dynamic transfer, a crowd counting model adapts the model parameters according to per sample of the data. For this reason, this adaptation algorithm increases the efficacy of the crowd counting method.

In terms of future direction, we would like to explore dynamic transfer between synthetic to real world scene adaptation. In addition to this, in this thesis, it is

assumed that the black-box model provides the predicted class scores. An even more challenging setting is where the black box only provides the predicted class label. We plan to address this more challenging case as future work.

Bibliography

[1] `https://github.com/tim-learn/SHOT`.

[2] `https://github.com/peterliht/knowledge-distillation-pytorch`.

[3] `https://pytorch.org/vision/master/_modules/torchvision/models/mobilenetv2.html`.

[4] Mathilde Bateson, Hoel Kervadec, Jose Dolz, Hervé Lombaert, and Ismail Ben Ayed. Source-relaxed domain adaptation for image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 490–499. Springer, 2020.

[5] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3722–3731, 2017.

[6] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. *Advances in neural information processing systems*, 29:343–351, 2016.

[7] Joan Bruna, Christian Szegedyszegedy, Ilya Sutskever, Ian Goodfellow, Wojciech Zaremba, Rob Fergus, and Dumitru Erhan. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

[8] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *Proceedings of the IEEE/CVF international conference on computer vision workshops*, pages 0–0, 2019.

[9] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Michael I Jordan. Partial transfer learning with selective adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2724–2732, 2018.

[10] Antoni B Chan, Zhang-Sheng John Liang, and Nuno Vasconcelos. Privacy preserving crowd monitoring: Counting people without people models or tracking. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–7. IEEE, 2008.

[11] Chen Change Loy, Shaogang Gong, and Tao Xiang. From semi-supervised to transfer counting of crowds. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2256–2263, 2013.

[12] Hanting Chen, Yunhe Wang, Chang Xu, Zhaohui Yang, Chuanjian Liu, Boxin Shi, Chunjing Xu, Chao Xu, and Qi Tian. Data-free learning of student networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3514–3522, 2019.

[13] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11030–11039, 2020.

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

[15] Yanyan Fang, Biyun Zhan, Wandi Cai, Shenghua Gao, and Bo Hu. Locality-constrained spatial transformer network for video crowd counting. In *2019 IEEE international*

conference on multimedia and expo (ICME), pages 814–819. IEEE, 2019.

[16] James Ferryman and Ali Shahrokni. Pets2009: Dataset and challenge. In *2009 Twelfth IEEE international workshop on performance evaluation of tracking and surveillance*, pages 1–6. IEEE, 2009.

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

[18] Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex Smola. A kernel method for the two-sample-problem. *Advances in neural information processing systems*, 19, 2006.

[19] Matan Haroush, Itay Hubara, Elad Hoffer, and Daniel Soudry. The knowledge within: Methods for data-free model compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8494–8502, 2020.

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

[21] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

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

[23] Mohammad Asiful Hossain, Mahesh Kumar, Mehrdad Hosseinzadeh, Omit Chanda, and Yang Wang. One-shot scene-specific crowd counting. In *BMVC*, page 217, 2019.

[24] Mohammad Asiful Hossain, Mahesh Kumar Krishna Reddy, Kevin Cannons, Zhan Xu,

and Yang Wang. Domain adaptation in crowd counting. In *2020 17th Conference on Computer and Robot Vision (CRV)*, pages 150–157. IEEE, 2020.

[25] Yunzhong Hou and Liang Zheng. Visualizing adapted knowledge in domain transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13824–13833, 2021.

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

[27] Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama. Learning discrete representations via information maximizing self-augmented training. In *International conference on machine learning*, pages 1558–1567. PMLR, 2017.

[28] Di Kang, Debarun Dhar, and Antoni Chan. Incorporating side information by adaptive convolution. *Advances in Neural Information Processing Systems*, 30, 2017.

[29] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[30] Andreas Krause, Pietro Perona, and Ryan Gomes. Discriminative clustering by regularized information maximization. *Advances in Neural Information Processing Systems*, 2010.

[31] Jogendra Nath Kundu, Akshay Kulkarni, Amit Singh, Varun Jampani, and R Venkatesh Babu. Generalize then adapt: Source-free domain adaptive semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7046–7056, 2021.

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

[33] Vinod K Kurmi, Venkatesh K Subramanian, and Vinay P Namboodiri. Domain impres-

sion: A source data free domain adaptation method. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 615–625, 2021.

[34] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of IEEE*, 1998.

[35] Yunsheng Li, Lu Yuan, Yinpeng Chen, Pei Wang, and Nuno Vasconcelos. Dynamic transfer for multi-source domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10998–11007, 2021.

[36] Zeqi Li, Ruowei Jiang, and Parham Aarabi. Semantic relation preserving knowledge distillation for image-to-image translation. In *European Conference on Computer Vision*, pages 648–663. Springer, 2020.

[37] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 6028–6039. PMLR, 2020.

[38] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. *Advances in neural information processing systems*, 30, 2017.

[39] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. *Advances in neural information processing systems*, 29:469–477, 2016.

[40] Yuang Liu, Wei Zhang, and Jun Wang. Source-free domain adaptation for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1215–1224, 2021.

[41] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In *32nd International Conference on Neural Information Processing Systems*, 2018.

[42] Raphael Gontijo Lopes, Stefano Fenu, and Thad Starner. Data-free knowledge distillation for deep neural networks. *arXiv preprint arXiv:1710.07535*, 2017.

[43] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representation by inverting them. *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[44] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015.

[45] Paul Micaelli and Amos Storkey. Zero-shot knowledge transfer via adversarial belief matching. *arXiv preprint arXiv:1905.09768*, 2019.

[46] Rafael Müller, Simon Kornblith, and Geoffrey Hinton. When does label smoothing help? *arXiv preprint arXiv:1906.02629*, 2019.

[47] Gaurav Kumar Nayak, Konda Reddy Mopuri, Vaisakh Shaj, Venkatesh Babu Radhakrishnan, and Anirban Chakraborty. Zero-shot knowledge distillation in deep networks. In *International Conference on Machine Learning*, pages 4743–4751. PMLR, 2019.

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

[49] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3967–3976, 2019.

[50] Xingchao Peng, Ben Usman, Neela Kaushik, Judy Hoffman, Dequan Wang, and Kate Saenko. Visda: The visual domain adaptation challenge. *arXiv preprint arXiv:1710.06924*, 2017.

[51] Mahesh Kumar Krishna Reddy, Mohammad Hossain, Mrigank Rochan, and Yang Wang. Few-shot scene adaptive crowd counting using meta-learning. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2814–2823, 2020.

[52] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. FitNets: Hints for thin deep nets. In *International Conference on Learning Representations*, 2015.

[53] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 2015.

[54] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *European conference on computer vision*, pages 213–226. Springer, 2010.

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

[56] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[57] Yuan Shi and Fei Sha. Information-theoretical learning of discriminative clusters for unsupervised domain adaptation. *arXiv preprint arXiv:1206.6438*, 2012.

[58] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shematikov. Membership inference attacks against machine learning models. In *IEEE Symposium on Security and Privacy*, 2017.

[59] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2107–2116, 2017.

[60] K Simonyan and A Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations*, 2015.

[61] Prabhu Teja Sivaprasad and Francois Fleuret. Uncertainty reduction for model adaptation in semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, number CONF. IEEE, 2021.

[62] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. *arXiv preprint arXiv:1611.02200*, 2016.

[63] Simen Thys, Wiebe Van Ranst, and Toon Goedemé. Fooling automated surveillance cameras: adversarial patches to attack person detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[64] F Tramèr, D Boneh, A Kurakin, I Goodfellow, N Papernot, and P McDaniel. Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Representations*, 2018.

[65] Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1365–1374, 2019.

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

[67] Xudong Wang, Zhaowei Cai, Dashan Gao, and Nuno Vasconcelos. Towards universal object detection by domain attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7289–7298, 2019.

[68] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kris-

ten Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8817–8826, 2018.

[69] Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. *Advances in Neural Information Processing Systems*, 32, 2019.

[70] Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8715–8724, 2020.

[71] Cong Zhang, Hongsheng Li, Xiaogang Wang, and Xiaokang Yang. Cross-scene crowd counting via deep convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 833–841, 2015.

[72] Jeffrey O Zhang, Alexander Sax, Amir Zamir, Leonidas Guibas, and Jitendra Malik. Side-tuning: A baseline for network adaptation via additive side networks. In *European Conference on Computer Vision*, 2020.

[73] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 589–597, 2016.