7-2023

Continual Domain Adaptation through Knowledge Distillation

Georgi Thomas
gt4330@rit.edu

Continual Domain Adaptation through Knowledge Distillation

Georgi Thomas

Continual Domain Adaptation through Knowledge Distillation

Georgi Thomas

July 2023

A Thesis Submitted
in Partial Fulfillment
of the Requirements for the Degree of
Master of Science
in
Computer Engineering

RIT | Kate Gleason College of Engineering

Department of Computer Engineering

Continual Domain Adaptation through Knowledge Distillation

Georgi Thomas

Committee Approval:

Dr. Andreas Savakis *Advisor* Date
Department of Computer Engineering

Dr. Dongfang Liu Date
Department of Computer Engineering

Dr. Andres Kwasinski Date
Department of Computer Engineering

Acknowledgments

I want to begin by thanking my advisor Dr. Andreas Savakis for all his support and guidance throughout my academic and research studies. I also want to thank Dr. Dongfang Liu and Dr. Andres Kwasinski for serving as members of my thesis committee. I would like to thank the Vision and Image Processing Lab, especially Rahi, Mihir, and Rajat for their help in my research endeavors. Finally, I want to thank my family and friends for their constant and continued support throughout this entire experience.

To my parents, who taught me that through hard work I could have my cake and eat it too.

Abstract

Domain Adaptation (DA) techniques aim to overcome the domain shift between a source domain used for training and the target domain used for testing and deployment. Domain adaptation methods assume the entire target domain is accessible during the adaptation process. We use efficient architectures in the continual data-constrained DA paradigm where the unlabeled data in the target domain is received continually in batches.

In recent years, Vision Transformers have emerged as an alternative to traditional Convolutional Neural Networks (CNNs) as the feature extraction backbone for image classification and other computer vision tasks. Within the field of DA, these attention-based architectures have proven to be more powerful than their traditional counterparts. However, they possess a larger computational overhead due to their model size. We design a novel framework, called **C**ontinual Domain **A**daptation through **K**nowl**E**dge Distillation (CAKE), that uses knowledge distillation (KD) to transfer to a CNN the more complex Vision Transformer's knowledge. By doing so, CNN-based adaptation obtains a similar performance to transformer-based adaptation while reducing the computational overhead. With this framework, we selectively choose samples from these batches to store in a buffer for selective replay. We mix the samples from the buffer with the incoming samples to incrementally update and adapt our model.

We show that distilling to a smaller network after adapting a larger model allows the smaller network to achieve better accuracy than if the smaller network adapted to the target domain. We also demonstrate that CAKE outperforms state-of-the-art unsupervised domain adaptation methods without full access to the target domain or any access to the source domain.

Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction

Computer Vision models are trained on a specific dataset or source domain, and once deployed they may be exposed to other domains. The source domain refers to the dataset where the model is trained and the target domain is the domain where the model is deployed and can make predictions. The domain shift between the source and target domains causes a drop in classification accuracy. Domain adaptation methods try to mitigate the domain gap between the source domain and the target domain, namely adversarial and source-free. Adversarial methods leverage the source domain to learn more general features across different domains. Source-free approaches include Hypothesis Transfer Learning [9] and Source Hypothesis Transfer [10] to transfer the source-trained model for target adaptation without access to the source domain..

Convolutional Neural Networks (CNN) have been the state-of-the-art backbone to extract features for image classification. In recent years, Vision Transformers have shown to be more powerful in various computer vision tasks. They have been shown to perform better than CNNs in domain adaptation due to their ability to generalize better over multiple domains. However, they have a significant computational load over traditional CNNs.

1.2 Thesis Contributions

- We propose to use knowledge transfer through knowledge distillation to leverage the feature extraction power of a larger attention-based model as a teacher and the efficiency of ResNet-50 as a student to adapt to new domains.
- We explore transformers and attention-based convolutional models to serve as teachers and help the student model adapt to the target domain.
- We use knowledge distillation within a newer continual learning paradigm in domain adaptation where target data is obtained in small batches. To do this, we propose using a buffer to selectively replay target samples during the adaptation process.

1.3 Motivation

Image classification is a fundamental task in computer vision and has many applications across many fields such as detecting cancer or object recognition for self-driving. However, with a changing domain, it is necessary to have a model that can continuously adapt to these new changes without reducing performance. This is particularly important in fields such as autonomous driving where new regions and new areas must be analyzed in real-time. This creates a necessity for a smaller network that can be run on edge devices and adapt efficiently. Harnessing the generalization capabilities of the attention-based architectures, the teacher model can distill the new domain to the student more efficiently while reducing the footprint of the deployed model. We title this framework CAKE, for **C**ontinual Domain **A**daptation through **K**nowl**E**dge Distillation.

1.4 Document Structure

The remainder of this document is structured as follows: Chapter 2 discusses the background material and contains an overview of related works in domain adaptation and knowledge distillation. Chapter 3 covers the proposed methodology and paradigms being analyzed. Chapter 4 presents the implementation details and datasets used. Chapter 5 discusses preliminary experiments that have been run to set up the framework and justifications for the framework. Chapter 6 presents our results compared to the state-of-the-art results, and also discusses different ablation studies that were run. Chapter 7 contains our concluding remarks and provides possibilities for future work.

Chapter 2

Background

2.1 Convolutional Neural Networks

Traditionally, Convolutional Neural Networks [11] (CNNs) have been the de facto architecture used to extract features from an image. CNNs are used in various computer vision tasks such as image classification and segmentation, object detection, and optical flow. CNNs are composed of primarily 3 types of layers: convolution, pooling, and fully connected layers. An example of this can be seen in Figure 2.1.

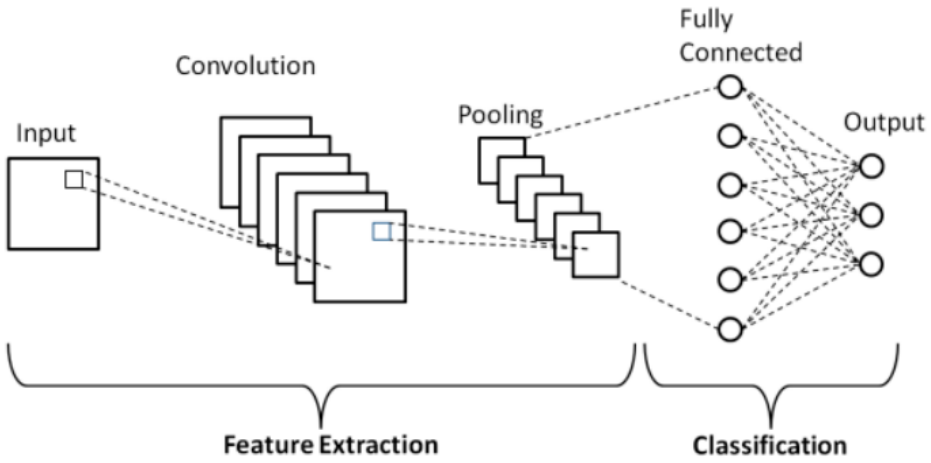

Figure 2.1: CNN Architecture Example for Image Classification [1] Image Source: [2].

The input layer contains a fixed size for an image which can be adjusted as necessary to fit the specific task. The convolution layer convolves the image with various filters using shared weights to extract features. Each kernel is responsible for extract-

ing features of the specific layer of input data. The pooling layers are non-trainable layers applied in a shifted window manner to reduce the features and generate a down-sampled version of the feature space. There are different techniques for implementing the pooling layers such as max pooling and average pooling. Max pooling involves extracting the maximum value from the values considered. Average pooling returns the average of the data seen by the pooling regions. A fully connected network is used to understand complex representations of features to extract outputs that closely relate to the ground truth.

While many popular CNN-based architectures exist, such as AlexNet [12], VGGNet [13], and EfficientNet [14], we focus on ResNet-50 [3] as it is the most common feature extractor used in domain adaptation image classification tasks.

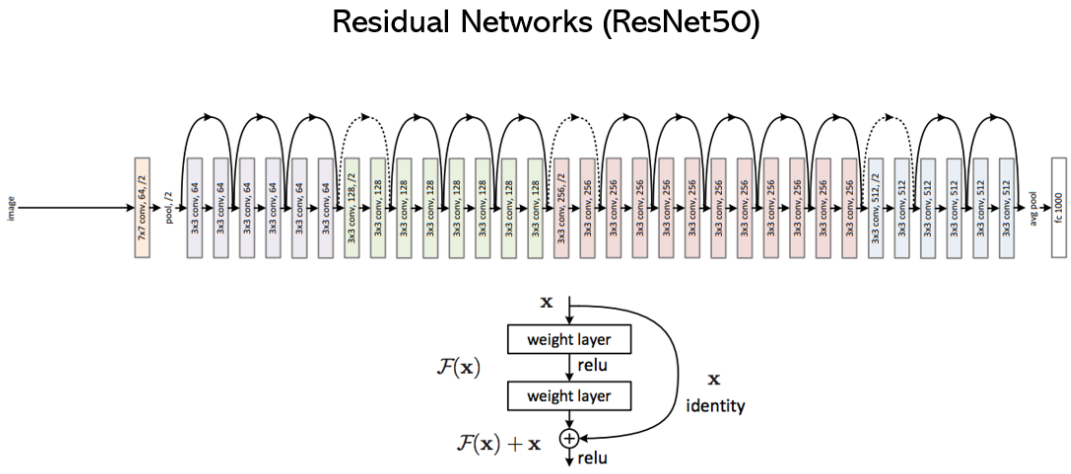

Figure 2.2: Model Overview of ResNet Architecture and Design of Skip Connection. Image Source: [3]

ResNet or Residual Network is a deep learning CNN-based model designed to resolve or reduce the problem of vanishing gradient through skip connections. ResNet accomplishes this by stacking multiple identity mappings to skip layers to use the activation of the previous layer. By skipping these layers, the model can converge must faster by pseudo-compressing the network. When the network is then retrained,

the layers are expanded and the network can extract a better feature space from the image.

2.2 Attention-Based Architectures

In recent years, attention-based architectures such as transformers have been gaining popularity across the field of machine learning due to their success in larger language models. Contrary to other architectures, transformers operate on the entire input sequence in parallel as opposed to sequentially. In the Natural Language Processing (NLP) domain, this was beneficial as it could draw comparisons between different words that may be distant from each other. Implementation of Vision Transformers (ViT) was introduced by [4]. Figure 2.3 depicts an overview of the ViT architecture as well as the design of the transformer encoder block.

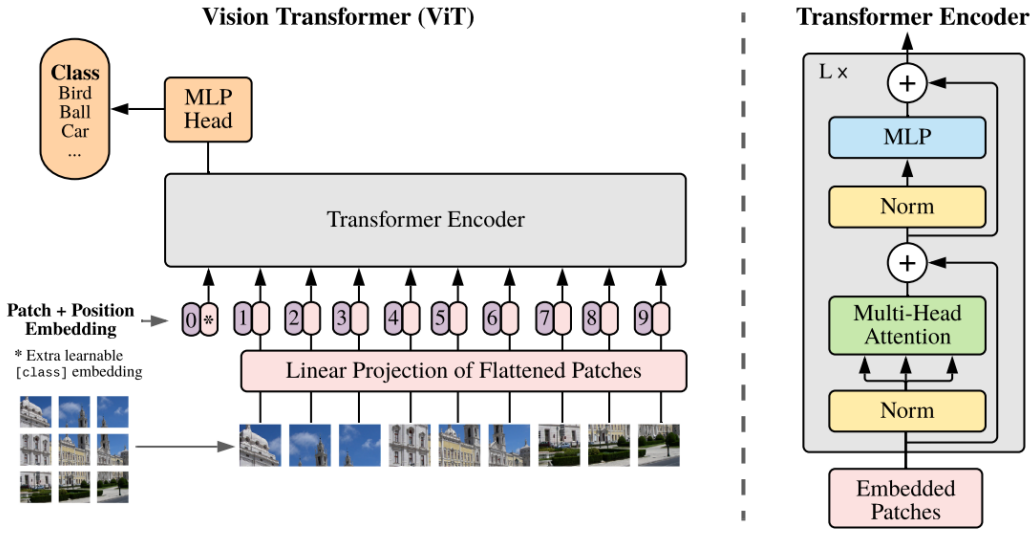

Figure 2.3: Model Overview of Vision Transformer Architecture and the Transformer Encoder Block. Image Source: [4]

Implementing self-attention directly to the image would attempt to draw relations between individual pixels, however, this vastly increases the complexity of the model. Instead, ViT [4] splits the image into image patches and treats these image patches as

words or tokens. ViT then applies a trainable linear projection layer across the image patches to create patch embeddings. ViT uses the patch embeddings and applies a positional embedding over them before passing them to the transformer encoder. The position embeddings are then used to retain positional information. The transformer encoder consists of alternating layers of Multi-Head Self-Attention (MSA) blocks and Multi-Layer Perception (MLP) layers. In this format, the MSA layer extracts global features by computing attention against all patches, while the MLP layer encodes the local features within each patch.

While ViT was revolutionary in the computer vision domain, it suffered from shortcomings such as its quadratic computational complexity of the self-attention mechanism. Some of these issues were addressed by the Shifted window (Swin) Transformer [5] through the introduction of hierarchical feature maps and shifted window attention. Hierarchical feature maps are the intermediate tensors generated by layers that are merged from one layer to each successive layer. This effectively downsamples the feature space and reduces the dimension of the feature map to compress the latent representation. As mentioned previously, another key introduction made by the Swin transformer is the change in the way the self-attention mechanism is used.

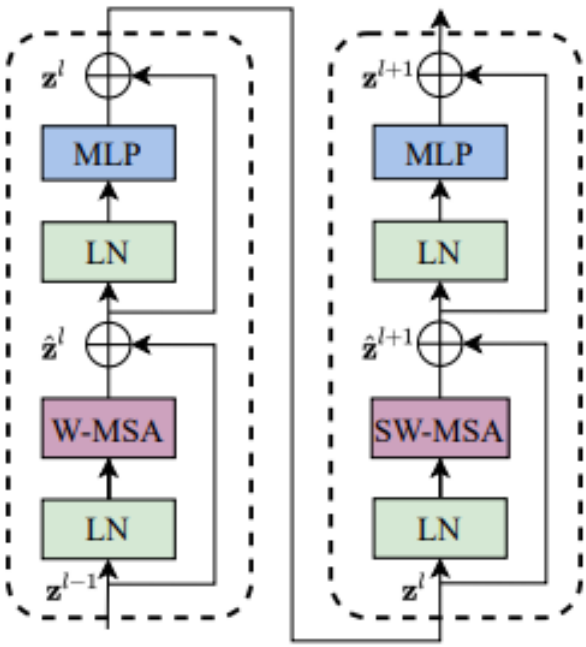

Figure 2.4: Swin Transformer Block with 2 Sub-Units. Image Source: [5]

The Swin transformer implements two types of MSA blocks: W-MSA and SW-MSA, where W-MSA denotes the window-based MSA and SW-MSA denotes the shifted-window MSA. With a window approach, the attention is applied to each window as opposed to the entire patch itself. The windows are arranged in an evenly partitioned manner so that they don't overlap. Due to the fixed window size, the computational complexity drops from quadratic to linear. However, since attention is not computed across multiple windows, it limits the modeling power of the architecture. To resolve this, the Swin transformer applies the SW-MSA and cyclic shift to introduce cross-window connections by shifting the windows toward the top left corner. The displaced patches are then fit to the windows with incomplete patches. With the cyclic shift, there may be windows that consist of patches that are not adjacent to each other, so a mask is applied to limit the self-attention computation for those windows.

With the growth of Vision Transformers, a new CNN architecture was introduced, known as ConvNeXt [15], which modernized ResNet through multiple strate-

gies. These strategies include moving up the depthwise convolution, adjusting the kernel size of the convolution layer, and the addition of a convolutional layer for spatial downsampling. Moving up the depthwise convolutional layer parallels the transformer blocks used in vision transformers. The depthwise convolution operation is similar to the self-attention operation of using a weighted sum to extract information in the spatial dimension. In vision transformer blocks, the MSA is kept before the MLP layer. To replicate this with ConvNeXt, they use an inverted bottleneck that emulates the vision transformers' enhanced capability to extract local features. Traditionally, with CNNs, the norm was to stack smaller convolutional layers of size 3x3, but ConvNeXt changes the kernel size to 7x7 to better extract global features. Inspired by the Swin transformer, a separate downsampling convolutional layer of size 2x2 is added to extract finer features and reduce the feature space. They [15] show that by modernizing the training methods and adapting the convolutional models to parallel transformer blocks, CNNs can compete with vision transformers in different computer vision tasks.

2.3 Image Classification

Image classification is used to identify an image out of a set of target classes. A standard image classification model is composed of a feature extraction backbone and a classification head. The training dataset is often preprocessed through different image augmentations to artificially expand the dataset and help the model extract better features and prevent over-fitting. The class of the object in question is used to label the image, and the feature extraction backbone attempts to extract features that are unique to the different classes. During the final classification step, the features of the image are fed to a classifier to extract the class label. While the final outputs are determined at the final layers in the classification head, much of the performance comes from a richer feature space.

2.4 Knowledge Distillation

An effective way to improve the accuracy or performance of classification tasks is to train various models and then use a weighting algorithm to refine predictions. However, this is very computationally intensive and inefficient. To tackle this issue, a model compression algorithm known as knowledge distillation [16] was proposed to transfer the knowledge from a larger model to a smaller model.

The smaller model (student) learns from the pre-trained larger model (teacher) by mimicking the output and leverages this methodology of model compression to achieve similar performance [16]. This also allows the student model to learn different features than it would have on its own. There exist three types of knowledge distillation: offline distillation, online distillation, and self-distillation. In offline knowledge distillation [17], the pre-trained teacher model is used as a black box model by extracting soft labels to teach the student model. With online knowledge distillation [18], the teacher can be pre-trained or it can be an untrained model. Using online knowledge distillation, an end-to-end framework is created where the soft labels go directly to the student model as the teacher model updates to learn the new data. Over time, the student model will become better at making the same predictions as the teacher. With self-distillation [19], the same model is used for the teacher and student. With this method, the old activations and the outputs from previous epochs are transferred to later epochs to train the student model. This distillation methodology behaves similarly to a momentum encoder [20] by regularizing the model based on past predictions to explore richer features.

2.5 Unsupervised Domain Adaptation

Deep learning is the state-of-the-art method used in computer vision to understand and classify images. However, these models are very reliant on their training data,

and a shift in the domain of the image can result in a significant loss in performance. A domain shift or domain variance occurs when there is a significant difference in the data distributions of the source and target domains.

Unsupervised domain adaptation is an extension of transfer learning where the target domain is unlabelled. There exist three types of domain adaptation: closed-set, open-set, and partial-set domain adaptation. With open-set [21] domain adaptation, there are many images in the target dataset, but only some of them belong to classes of interest. In closed-set domain adaptation, there are no discrepancies in the number of classes between the source and target domains. Partial-set domain adaptation is a paradigm in which the target domain has fewer classes of interest than the source domain.

Many UDA methods utilize adversarial methods [22, 23] to learn and align the feature spaces between the source and target domains. However, recent methods have used a source-free methodology for test time adaptation. This significantly reduces the data storage footprint and prevents the extraction of information in the source domain which could contain sensitive information.

2.6 Continual Learning

Continual learning is a paradigm where a model learns from data sequentially, as data samples are acquired without access to earlier samples. In this paradigm, the model can forget the knowledge that it had learned in preceding tasks. This forgetting of data is known as catastrophic forgetting. Continual learning can be sectioned into two main categories: incremental batch learning and streaming learning. Incremental batch learning is a paradigm in which data is incrementally added in batches for training. Once a particular batch has been trained on by the model, it is discarded and the next batch is fetched to train on. The model is allowed to train on any given batch for multiple epochs before the next batch is fetched. The paradigm of streaming

learning [24, 25] is an extension of incremental batch learning where training samples are fed to the model one sample at a time, and the model is only trained for one epoch over the entire training dataset.

We use batch streaming [26] with domain adaptation as a combination of incremental batch learning and streaming learning. Similar to incremental batch learning, the target domain is split into small batches that are input sequentially to the network. Following the paradigm of streaming learning, the model is allowed to adapt to one batch of data at a time. Once that batch is adapted, the next batch is fetched and provided to the network. This batch streaming process can be shown in Figure 2.5.

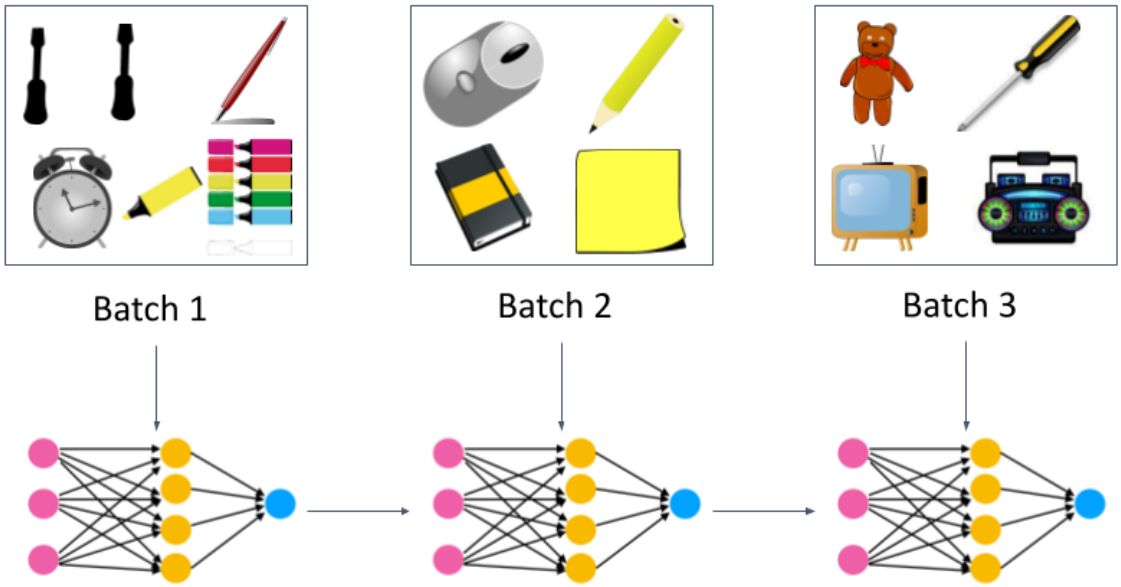

Figure 2.5: Visual Depiction of Batch Streaming Learning Paradigm.

This paradigm is noted to be more challenging than the standard domain adaptation pipeline due to the streaming nature and limited availability of target data.

2.7 Related Work

Existing methods in UDA use adversarial methods [27], which require the target models to access some of the source data. Source-free unsupervised domain adaptation methods include Source HypOthesis Transfer (SHOT) [28] and Contrastive Test-Time Adaptation (ADAContrast) [29]. All these networks use ResNet-50 as the target backbone for feature extraction.

SHOT uses a source-trained model which comprises a feature extractor and a classifier head. When the model is deployed in the target domain, the classifier head is frozen. SHOT implements a clustering algorithm using the cosine similarity function to extract pseudo labels for adaptation to the target domain. During adaptation, the feature extractor learns the features based on the pseudo-labels. By extension, this improves the clustering algorithm as it can extract better pseudo-labels in the target domain. SHOT implements information maximization to force the labels of the target model to be unambiguous and uniform.

ADAContrast [29] implements a similar methodology where the source model is transferred to the target domain without any data from the source domain to align its features. ADAContrast creates a copy of the source encoder and implements it as a momentum encoder, which regularizes the prediction model through contrastive loss. ADAContrast uses this momentum encoder which is jointly applied with the self-training to exclude same-class negative pairs. It uses the same clustering algorithm as the SHOT framework to test its pseudo-labels, however, a large difference is the image augmentations applied for the clustering and the classifier. It uses weaker augmentations for the clustering algorithm and a stronger augmentation for the classifier head to enforce a weak-strong consistency during self-training.

Knowledge distillation has also been applied to domain adaptation through a Distill and Fine Tune (DINE) framework [30]. This framework uses a two-step process

by pretraining a teacher model on the source domain and using knowledge distillation to distill target information to a custom target model. The target model then adapts to the target domain using information maximization loss. DINE uses the traditional ViT as the teacher, and ResNet-50 as the student. The framework they use for adaptation involves using the source-trained teacher model as a black box predictor from the source domain.

Continual Unsupervised Domain Adaptation [26] (ConDA) proposes a continual UDA framework where they use a class-balanced buffer and selectively replay target samples during the adaptation process. Similar to SHOT, ConDA uses a clustering algorithm to extract pseudo-labels to adapt the model and use information maximization to increase variation in the output labels. Since the pseudo-labeling algorithm may assign incorrect labels, ConDA uses high-confidence samples to store to the buffer and replay. Inspired by these frameworks, we design a novel framework named CAKE.

Chapter 3

CAKE Methodology

In this chapter, we present a novel framework, called **C**ontinual Domain **A**daptation through **K**nowl**E**dge Distillation (CAKE).

3.1 Continual Domain Adaptation through Knowledge Distillation

The CAKE framework combines knowledge distillation with continual unsupervised domain adaptation to create a more efficient, tuned target model. Attention-based architectures excel in cross-domain image classification, compared to their CNN counterparts, such as ResNet. Additionally, in the case of attention-based architectures, running the network on low-power and low-resource devices may not be possible. Therefore, this thesis aims to leverage the performance of attention-based architectures with the lower resource usage of ResNet to deploy in the target domain. An overview of the CAKE framework is illustrated in Figure 3.1.

We focus on the K-way cross-domain image classification task aiming to address a more realistic closed-set UDA setting where the data is received continually in batches and without access to the whole target domain. For an unsupervised domain adaptation task, we are provided with n_s labeled samples $\{(x_s, y_s) \in (X_s, Y_s)\}$ from the source domain, D_s, and n_t samples $\{x_t \in X_t\}$ from the target domain, D_t. The goal is to learn a mapping $(f_t : X_t \rightarrow Y_t)$ to determine the corresponding labels

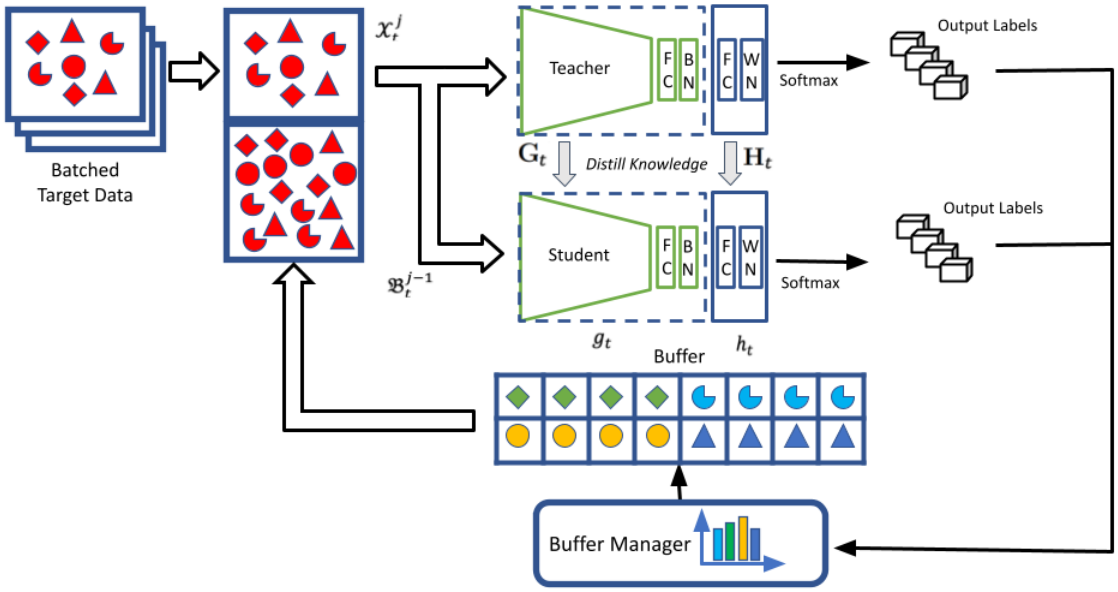

Figure 3.1: An overview of the CAKE method. We use a source-trained teacher and student to initialize the adaptation to the target domain. The teacher network extracts and smooths labels extracted from the samples to teach the smaller student network through knowledge distillation. The buffer manager stores the confident samples denoted by the teacher to replay back to the network in future batches.

$\{y_t \in Y_t\}$ for the target domain. Under the closed-set setting, we assume that the number of classes in the source domain is the same as in the target domain, namely that $C_t = C_s$.

3.2 Continual Domain Adaptation

Under the continual paradigm, the target domain is not available all at once. We consider the batch streaming approach to split the target domain into smaller batches that are sequentially sent to the network. We further split these batches into multiple mini-batches to train the target model.

To perform continual domain adaptation, we take inspiration from the buffer strategy used by ConDA [26]. With continual learning, neural networks are susceptible to catastrophic forgetting and we look to mitigate this through memory replay. Memory

replay is a process inspired by the biological phenomenon of hippocampal replay, in where, there is a re-occurrence of cell activations that had occurred during awake phases, but at a much faster rate [31].

In the CAKE framework, we design a class-balanced buffer to store samples and their predicted class labels to selective replay. The buffer is populated with the first batch of target samples. Our model only requires access to the samples stored in the buffer and the incoming samples for each successive batch of target data in the adaptation process. The selective replay prevents the model from overfitting to new batches of data by replaying older samples.

The class-balanced buffer is defined as B_t with states $\{B_t^1, B_t^2, B_t^3, ..., B_t^m\}$ where m is the current batch of data from the target domain for balanced training. We allot an equal number of buffer slots for each class relative to the buffer length and the number of classes. The buffer is populated with data after the network is trained on a batch of target data and stores the samples as well as their respective labels as predicted by the teacher model. Our model then only needs access to the data stored in the buffer as well as the incoming batch data.

The network adapts on each batch X_t^m and outputs $\{Y_t^m, S_t^m\}$, where S_t^m is the softmax classification score. We compute the soft labels for the buffer samples for the current state of the model. The buffer takes in the current samples and soft labels. Initially, the incoming batch samples are grouped based on the output label Y_t^m and are sorted based on the confidence S_t^m. The buffer manager picks the high-confidence samples denoted by the teacher model. This allows the teacher model to better adapt to the target domain which in turn allows the student model to learn better labels from the teacher.

3.3 Source Training

We train the teacher and student networks on the source data X_s in the source domain D_s to learn the feature mapping ($f_t : X_s \rightarrow Y_s$). We design our models for the teacher and student networks with a feature extractor and a classification head. We denote feature extractors for the teacher and student networks as G_s and g_s respectively. Likewise, we denote the classification head for the teacher and student as H_s and h_t. We define the source teacher to be F_s, where $F_s = H_s(G_s)$ and the source student to be f_s, where $f_s = h_s(g_s)$. Within the feature extraction module, we include a batch normalization layer, and in the hypothesis module, we include a weight normalization layer. A depiction of these models can be seen in Figure 3.2.

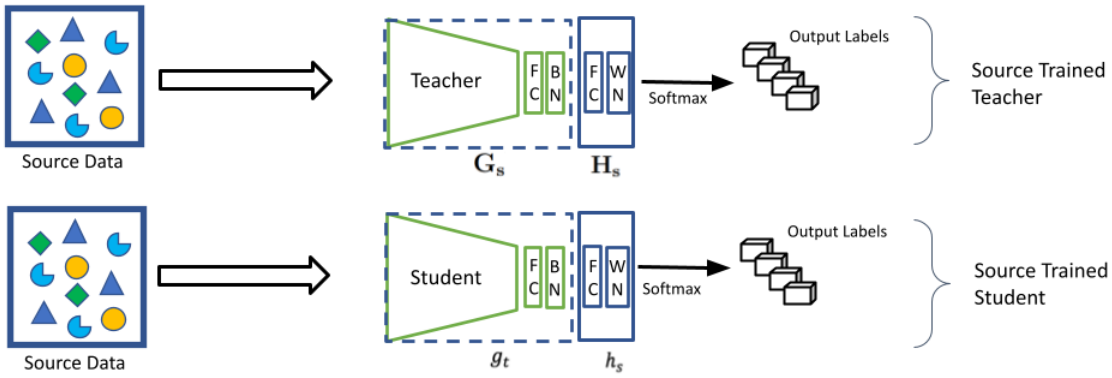

Figure 3.2: Source training of teacher and student models

After feature extraction, the classifier head returns k logits, where k is the total number of classes within the dataset. We train on the source data by minimizing the cross-entropy loss in conjunction with label smoothing for the source training procedure. We use label smoothing to increase the model's ability to properly cluster

and separate the different classes. Smoothed labels help to decrease the gaps between the predictions and prevent the model from being overconfident. We use q as the one-hot encoding of the output, where q is defined to be '1' for the intended class and '0' for any other class. We define the true labels as q_k and the smoothed label as q_k^{ls}. The equation for the label smoothing is defined as

$$q_k^{ls} = (1 - \alpha)q_k + \alpha/k \tag{3.1}$$

where α is the label smoothing parameter predetermined to be 0.1. We use the softmax probability

$$\delta_k(a) = \frac{exp(a_k)}{\sum_i exp(a_i)} \tag{3.2}$$

where $\delta_k(a)$ defines the k-th element in the softmax output of a K-dimensional vector a. We implement this into the cross-entropy loss function which becomes

$$\mathcal{L}_{src}(f_s; X_s, Y_s) = -\mathbb{E}_{(x_s,y_s)\in\{X_s,Y_s\}} \sum_{k=1}^{K} q_k^{ls}\ log\ \delta_k(f_s(x_s)) \tag{3.3}$$

3.4 Target Adaptation

We adopt the Information Maximization loss from [32] to increase the diversity among predictions of the teacher and student model during adaptation. Information Maximization(IM) is a combination of entropy loss $\mathcal{L}_{ent}$ [33] and diversity loss $\mathcal{L}_{div}$. We define $\mathcal{L}_{ent}$ and $\mathcal{L}_{div}$ as follows:

$$\mathcal{L}_{ent}(f_t; X_t) = -\mathbb{E}_{(x_t)\in\{X_t\}} \sum_{k=1}^{K} \delta_k(f_t(x_t))\ log\ \delta_k(f_t(x_t)) \tag{3.4}$$

$$\mathcal{L}_{div}(f_t; X_t) = \sum_{k=1}^{K} q_k\ log\ (q_k) \tag{3.5}$$

We denote f_t to be some target model adapting to the target data and q_k as the mean softmax output of the target data seen by the model. The equal diversity loss $\mathcal{L}_{div}$ aims to make network predictions diverse for all classes to prevent similar one-hot encodings of the observed target data. We then define IM loss as $\mathcal{L}_{IM}$, where

$$\mathcal{L}_{IM}(f_t; X_t) = \mathcal{L}_{ent} + \mathcal{L}_{div} \tag{3.6}$$

We use L_{IM} during the adaptation process for the teacher and the student models. We use it for the teacher to refine the extracted pseudo-labels to teach the student and during student adaptation to increase the variation in the outputs. From this point on, we define the target teacher to be F_t, where $F_t = H_t(G_t)$ and the target student to be f_t, where $f_t = h_t(g_t)$.

3.4.1 Teacher Model Adaptation

Unsupervised domain adaptation frameworks typically utilize a clustering method to self-supervise the pseudo-label extraction. However, these methods heavily rely on rich latent representations from the feature extraction backbone. With the superiority of attention-based architectures, we look to leverage the outputs generated by the attention-based architectures and replace the pseudo-labeling clustering algorithm with online knowledge distillation. We apply the clustering algorithm inspired by SHOT [10] to the teacher models for adaptation to the target domain and for teaching the student better labels.

For generating the pseudo-labels for teacher adaptation, we first determine the initial centroids, $c_k^{(0)}$, using the softmax output of the target samples,

$$c_k^{(0)} = \frac{\sum_{x_t \in X_t} \delta_k(p_t(x_t))\ (F_t(x_t))}{\sum_{x_t \in X_t} \delta_k(F_t(x_t))} \tag{3.7}$$

where p_t describes the previously learned target hypothesis and F_t are the current

predictions. We use the cosine distance function and minimize the distance between samples where $D(a, b)$ is the cosine distance function and a and b are the two samples.

$$\hat{y}_t^{(0)} = arg_k min(D(F_t(x_t), c_k^{(0)}) \tag{3.8}$$

After extracting the initial pseudo-labels, the cluster centers are recomputed as follows.

$$c_k^{(1)} = \frac{\sum_{x_t \in X_t} \mathbb{1}(\hat{y}_t = k)\ (F_t(x_t))}{\sum_{x_t \in X_t} \mathbb{1}(\hat{y}_t = k)} \tag{3.9}$$

The final pseudo-labels are then computed with the updated cluster centers using

$$\hat{y}_t^{(1)} = arg_k min(D(F_t(x_t),\ c_k^{(1)}) \tag{3.10}$$

where the $y_t^{(1)}$ are the pseudo-labels extracted from the input target data X_t.

As shown below, we use the pseudo-labels from the clustering algorithm to minimize the cross-entropy loss, $\mathcal{L}_{Tce}$, for the teacher model using the target samples.

$$\mathcal{L}_{Tce}(F_s; X_s, Y_s) = -\mathbb{E}_{(x_t, \hat{y}_t) \in \{X_t, \hat{Y}_t\}} \sum_{k=1}^{K} \mathbb{1}_{[k=\hat{y}_t]}\ log\ \delta_k(F_t(x_t)) \tag{3.11}$$

Using the teacher model's cross-entropy loss, $\mathcal{L}_{Tce}$, with the IM loss, $\mathcal{L}_{IM}$, our final adaptation objective function becomes,

$$\mathcal{L}_{F_t} = \mathcal{L}_{Tce} + \mathcal{L}_{IM} \tag{3.12}$$

It should be noted that during the teacher's adaptation process, the classifier head is frozen. This helps the model extract better features that can then be used to extract better pseudo-labels with the clustering algorithm. This leads the teacher network to learn good network predictions as a unified approach to better adapt to the target domain.

3.4.2 Student Model Adaptation

In order to adapt the student model to the target domain, we use knowledge distillation by training the student model to learn the labels from the teacher network. Unlike the teacher model, we leave the classifier head unfrozen to allow the entire network to fit the teachings of the teacher model.

We utilize the Kullback-Leibler loss or $\mathcal{L}_{kl}$ defined as follows.

$$\mathcal{L}_{kl}(F_t(x_t)||f_t(x_t)) = \sum_{x_t \in X_t} F_t(x_t)\ log\left(\frac{F_t(x_t)}{f_t(x_t)}\right) \tag{3.13}$$

We treat the labels from the teacher model as strong labels and use the following consistency loss,

$$\mathcal{L}_{KD}(f_t; X_t, F_t) = \mathbb{E}_{x_t \in X_t}\ \mathcal{L}_{kl}\ (F_t(x_t)\ ||\ f_t(x_t)) \tag{3.14}$$

The issue with using this consistency loss is that KD is often used in a supervised setting. However, with UDA, the teacher outputs for some target instances may be inaccurate. Inspired by [30], we use Adaptive Label Smoothing (AdaLS) to generate a revised output of $\hat{p}$. We have the teacher F_t revise the output p with the top-r values. We consider T_p^r as the set of the top-r labels of classes in the original output p.

$$\hat{p}(r) = \begin{cases} p_i, & i \in T_p^r \\ (1 - \sum_{j \in T_p^r} p_j)\ /\ (K - r), & otherwise \end{cases} \tag{3.15}$$

We empirically select $r = 3$ to choose the top 3 classes and smooth out the remaining labels. We use these refined pseudo-labels to reduce the noisiness of the labels extracted from the teacher and impose a uniform distribution on the labels similar to label smoothing. Smoothing to highlight the most confident classes allows the student to learn from the samples that the teacher is confident about rather than

learning from noisy labels that may occur with the domain shift.

We further regularize the knowledge learned by the student through MixUp [34] and employ the following interpolation consistency training loss defined by [35].

$$\begin{aligned}\mathcal{L}_{mix}(f_t; X_t) = \mathbb{E}_{x_t^i, x_t^j \in X_t}\ \mathbb{E}_{\lambda=0.3} \\ \mathcal{L}_{KL}(Mix_\lambda\ (f_t(x_t^i), f_t(x_t^j)), f_t/(Mix_\lambda(x_t^i, x_t^j)))\end{aligned} \tag{3.16}$$

where MixUp, $Mix_\lambda(a, b)$ is defined to be as follows

$$Mix_\lambda(a, b) = \lambda \cdot a + (1 - \lambda) \cdot b \tag{3.17}$$

Similar to the teacher model, we apply IM loss to the student to encourage the label distribution to be uniform and unambiguous. Our final objective function for the student network becomes

$$\mathcal{L}_{f_t} = \mathcal{L}_{KD} + \mathcal{L}_{mix} + \mathcal{L}_{IM} \tag{3.18}$$

Chapter 4

Implementation Details

In this chapter, we discuss the various architectures used, datasets tested, different parameters, and continual settings.

4.1 CAKE Models

For the student model shown in 3.1, we use the traditional ResNet-50 backbone for a direct comparison to other domain adaptation methods that use the same backbone. For the teacher model, we primarily focus on ConvNeXt due to its lower data requirements relative to transformer architectures while still keeping the attention mechanism. However, we also test Swin and ViT as other potential teachers and compare their results. The parameter size of each model that uses an image size of 224×224 is shown in Table 4.1.

Model	Image Size	Num Parameters
ResNet	224×224	23M
ConvNeXt	224×224	89M
ViT	224×224	86M
Swin	224×224	88M

Table 4.1: Information about each backbone architecture used.

4.2 DataSets

To analyze and test our models, we use three standard domain adaptation datasets: Office-31 [6], OfficeHome [7], and DomainNet-126 [8].

4.2.1 Office 31

The Office31 dataset [6] is a small domain adaptation dataset consisting of 4110 images from 31 different categories. This dataset consists of 3 domains: Amazon (**A**), DSLR (**D**), and Webcam (**W**). Seen in Figure 4.1 are 4 samples of the classes across 3 domains.

Figure 4.1: Sample Images from Office-31 [6] Dataset

4.2.2 OfficeHome

The OfficeHome dataset [7] is a smaller domain adaptation dataset consisting of 15,500 images from 65 different categories. There are 4 domains in this dataset: Art (**Ar**), Clipart (**Cl**), Real World (**Rw**), and Product (**Pr**). The 4 domains and 16 of the 65 classes are shown below in Figure 4.2.

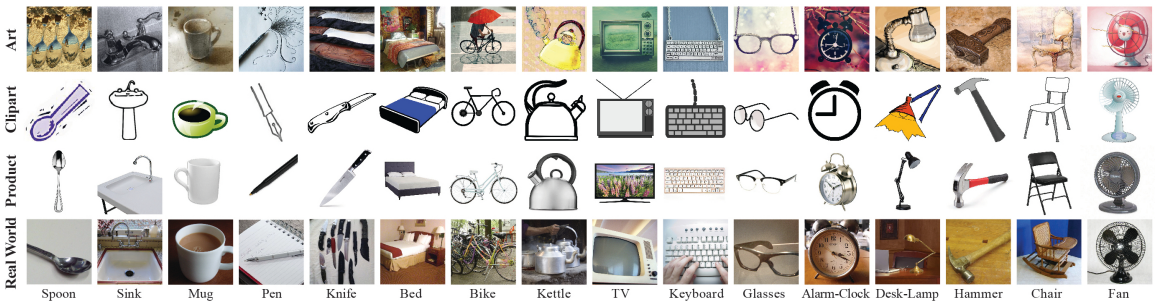

Figure 4.2: Sample Images from OfficeHome [7] Dataset

4.2.3 DomainNet-126

DomainNet-126 is a subset of the DomainNet [8] dataset. The DomainNet dataset is a large UDA dataset with over 600,000 images across 6 domains: Clipart, Infograph, Painting, Quickdraw, Real, and Sketch. Following [36], we use 126 classes from 4 domains: Real (**R**), Clipart (**C**), Painting (**P**), and Sketch (**S**) for evaluation. Shown in Figure 4.3 are 16 of 126 classes across those 4 domains.

Figure 4.3: Sample Images from DomainNet-126 [8] Dataset

4.3 Parameter Settings

CAKE is implemented in PyTorch and uses Timm (PyTorch Image Model Library) [37] to load models for analysis, training, and testing. We use the stochastic gradient descent (SGD) [38] optimizer for training on all datasets. We fix the student's learning rate to 0.001 and the teacher's learning rate to 0.001. The layers following the student backbone have a learning rate of 10 times the learning rate of the backbone. The

classifier of the teacher architecture is frozen and is not updated. However, the bottleneck layer of the teacher network has a learning rate of 0.001. Since we run CAKE in an online setting, there is no need for a parameter for a number of epochs as the model only receives the data once.

4.4 Continual Adaptation Setup

For the Office dataset, we use a mini-batch size of 32 and an incoming batch size of 32. We set the buffer size to 124 for 4 samples per class in a fully-balanced buffer. For experiments on the OfficeHome dataset, we use a mini-batch buffer size of 32 and an incoming batch size of 128. We set the buffer size to 520 samples resulting in 8 samples per class when the buffer is fully balanced.

4.5 Evaluation Metrics

The primary metric used to compare different models and different architectures is the mean accuracy. Mean accuracy measures the model's performance on different image sets relative to the ground truth. It is defined by the number of labels the model correctly classified divided by the total number of predictions performed by the model. We test accuracy on domain generalization and domain adaptation. Generalization is defined as testing a source-trained model on the target data, where inferences of the class labels are derived from the source knowledge. Adaptation is fine-tuning the source-trained model to the target data without knowledge of true labels. We use the source-trained models as a way to compare how much the model performance has improved after adaptation.

Chapter 5

Benchmarking and Adapt to Distill Experiments

5.1 Benchmarking Experiments

In order to have a fair comparison between both scenarios, we use three different backbones acting as the teacher models for our training pipeline: ViT, Swin, ConvNeXt. Given its simplicity and distinction across different domains, we use the OfficeHome dataset for our preliminary experiments. We also use the DomainNet-126 dataset due to its larger number of training samples. In order to establish a baseline for the performance of the different backbones, we compare multiple models that were trained only on the source domain. Tables 5.1 and 5.2 contain the generalization performance of the source-trained models on the target without any adaptation.

Table 5.1: Generalization performance Mean percent accuracy for source-trained models tested on the target domain for OfficeHome.

Backbone	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.
ResNet-50	44.5	65.6	74.1	54.0	61.2	64.7	52.1	41.2	73.3	65.1	46.2	78.0	60.0
ConvNeXt	72.9	86.0	89.3	82.6	85.2	87.5	80.2	67.9	89.9	82.6	70.2	89.7	82.0
ViT	50.5	82.5	86.8	74.3	82.7	84.3	73.2	50.4	88.2	77.7	49.8	87.6	74.0
SWIN	70.1	85.2	89.0	82.3	86.8	88.1	80.0	66.4	89.4	83.3	68.4	90.4	81.6

Table 5.2: Generalization performance Mean percent accuracy for source-trained models tested on the target domain for DomainNet-126.

Backbone	C → P	C → R	C → S	P → C	P → R	P → S	R → C	R → P	R → S	S → C	S → P	S → R	Avg.
ResNet-50	47.7	61.3	48.7	55.0	74.8	50.0	57.6	63.4	48.6	57.1	52.5	60.2	56.4
ConvNeXt	74.0	83.3	72.6	75.8	88.2	70.8	74.8	79.2	68.5	76.8	76.4	83.5	77.0
ViT	70.7	80.7	67.4	70.7	85.9	60.3	67.7	74.5	58.2	74.7	74.2	82.0	72.2
SWIN	74.2	83.7	71.5	76.3	88.0	68.5	75.1	78.5	67.6	77.4	75.9	83.8	76.7

With these baselines established, we adapt the source-trained models to each target domain using the adaptation strategy mentioned in Section 3.4.1 with IM and Mixup losses, and the pseudo labeling algorithm. We show the updated performance of each backbone on the target domains in Tables 5.3 and 5.4 for OfficeHome and DomainNet-126 datasets respectively.

Table 5.3: Adaptation performance Mean percent accuracy using the SHOT framework for target-adapted models tested on the target domain for OfficeHome.

Backbone	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.
ResNet-50	48.9	72.0	75.5	64.8	76.5	75.2	67.7	53.9	82.3	72.2	58.6	82.7	69.2
ConvNeXt	79.2	92.4	92.2	88.4	92.1	92.3	86.6	77.6	92.3	87.6	78.7	92.8	87.7
ViT	69.8	89.6	90.0	84.8	89.6	89.5	83.8	67.5	90.5	84.5	69.7	92.3	83.4
SWIN	76.2	91.5	91.7	87.4	92.0	92.0	86.6	77.0	91.7	87.9	76.6	94.2	87.1

Table 5.4: Adaptation performance Mean percent accuracy using the SHOT framework for target-adapted models tested on the target domain for DomainNet-126.

Backbone	C → P	C → R	C → S	P → C	P → R	P → S	R → C	R → P	R → S	S → C	S → P	S → R	Avg.
ResNet-50	61.2	77.3	59.7	68.4	80.1	61.1	68.1	66.3	58.2	68.8	63.2	76.4	67.4
ConvNeXt	79.2	90.1	76.5	79.4	90.4	77.1	79.5	81.4	76.1	79.5	80.5	89.5	81.6
ViT	76.2	86.9	71.8	77.5	87.9	70.9	74.7	77.1	66.5	80.2	78.2	87.7	77.9
SWIN	79.8	90.0	75.0	80.9	90.2	76.2	81.2	81.1	74.5	82.0	80.1	89.8	81.7

Comparing results in Tables 5.1, 5.2, 5.4, and 5.4, we see that ResNet performs significantly worse than the other architectures across the dataset in generalization and adaptation tasks. Attention-based architectures like ConvNeXt and Swin show significantly better performance on the generalization compared to the adapted ResNet

model. We then ask the question: is there a way to leverage these powerful architectures to improve the performance of the ResNet model?

5.2 Adapt to Distill Experiments

Using insight from DINE [30], we discover that knowledge distillation is an effective method of model compression. However, we encounter an interesting question: which approach is superior - adapting the source-trained teacher before distilling to the student model, or distilling the source-trained teacher and adapting with the student model? In essence: is it better to adapt-to-distill or distill-to-adapt?

We begin by testing the first option: adapting to distill. We first adapt the attention-based architecture using the teacher's adaptation process mentioned in Section 3.4.1. We then distill that knowledge down to the ResNet model. For the distill-to-adapt experiments, we distill the source-trained teacher's knowledge to the ResNet model. We then adapt this ResNet model using the teacher's adaptation process. We provide a quantitative comparison of the OfficeHome and DomainNet-126 datasets in Tables 5.5 and 5.6, respectively.

The "Source-Trained Teacher Distilled to ResNet" displays the student performance in the distill-to-adapt process before adapting using the student model. The "Adaptation with ResNet using SHOT" presents the results of the final performance of the student network in the distill-to-adapt framework. The "Target-Adapted Teacher Distilled to ResNet" displays the performance of the student model after learning from the adapted teacher in the adapt-to-distill mode.

Table 5.5: Accuracy of ResNet-50 for Adapt to Distill or Distill to Adapt on OfficeHome.

Teacher / Mode	Source-Trained Teacher Distilled to ResNet	Adaptation with ResNet using SHOT	Target-Adapted Teacher Distilled to ResNet
ViT	74.1	74.8	80.0
Swin	75.9	76.2	82.4
ConvNeXt	73.0	76.2	82.3

The results in Table 5.5 illustrate that when using the source-trained teacher, ResNet adapts to the target data better than if it had adapted by itself. When that same ResNet model adapts to the target domain, there is an increase in performance ranging from 0.3% to 3.2%. However, if we adapt the teacher network before distilling, there is an average accuracy increase of 5.9% - 9.3%.

Table 5.6: Accuracy of ResNet-50 for Adapt to Distill or Distill to Adapt on DomainNet-126.

Teacher / Mode	Source-Trained Teacher Distilled to ResNet	Adaptation with ResNet using SHOT	Target-Adapted Teacher Distilled to ResNet
ViT	75.9	73.5	78.0
Swin	77.5	74.3	80.1
ConvNeXt	77.2	74.5	80.1

The DomainNet-126 results in Table 5.6 reveal similar results to the OfficeHome experiments. When distilling from the source-trained teacher, ResNet learns the target data better than if it had adapted to the target data by itself. However, every instance of ResNet adapting from the distilled model showed a negative transfer, which means that the performance of the model decreased after the adaptation process. This is because the generalization capability of the attention-based architecture is

much better than the pseudo-labeling and clustering algorithm derived using ResNet. Adapting to distill using the DomainNet dataset resulted in an average increase of 2.1% - 2.9%.

It can be concluded from these experiments that it is much better to adapt to distill, instead of distilling to adapt. Since the attention-based models generalize better on the target data, they will adapt to the target data more effectively. By leveraging the knowledge gained from the attention-based architecture while adapting ResNet, we help ResNet adapt to the target domain better than if it had adapted on its own. With confidence in our methods from these results, we move to implement knowledge distillation in the continual paradigm.

Chapter 6

Results for the CAKE Framework

In this section, we evaluate the CAKE framework against current state-of-the-art (SOTA) adaptation methods and compare our results against established benchmarks as well as the accuracy of the teacher models. In order to understand the importance of different parameter settings and components, we conduct ablation studies.

6.1 Comparison to State-of-the-Art

To test our framework, we use three standard domain adaptation datasets: Office-31 [6], OfficeHome [7], and DomainNet-126 [8]. We present two modes of adaptation: full and continual. In the results tables, "Full" denotes a mode where the model has access to the full target data and continual (Cont.) is the paradigm where the model only has access to the data one batch at a time.

6.1.1 Results for Office-31 Dataset

Table 6.1 presents the results of the CAKE architecture compared to other architectures on the Office-31 dataset. The CAKE architecture uses an incoming batch size of 32 and a buffer size of 124 (4 samples per class) for the continual mode and uses the top 3 class predictions for knowledge distillation.

Table 6.1: CAKE results and comparison to state-of-the-art on the Office-31 dataset. * is used to denote the SOTA adversarial unsupervised domain adaptation methods that require access to both source and target domains during adaptation. Bold indicates the top performance, while the underlined values indicate the second-best performance.

Method	Target	A→D	A→W	D→A	D→W	W→A	W→D	Mean
FixBi*	Full	95.0	96.1	78.7	**99.3**	79.4	**100.0**	91.4
CoVi*	Full	95.0	**97.6**	77.5	**99.3**	78.4	**100.0**	91.8
SHOT	Full	94.0	90.1	74.7	98.4	74.3	99.9	88.6
DINE	Full	<u>95.5</u>	94.8	**81.2**	<u>98.5</u>	<u>82.0</u>	99.7	91.9
SHOT	Cont.	84.7	85.3	69.8	97.9	65.5	99.2	83.7
ConDA	Cont.	84.7	88.7	72.8	98.2	70.0	99.8	85.7
CAKE	Cont.	**96.6**	<u>96.9</u>	<u>80.0</u>	**99.3**	**82.9**	**100.0**	**92.6**

Based on the results shown in Table 6.1, we see that CAKE outperforms all other continual methods in all adaptation categories by a significant margin. It is consistently on par or better compared to the SOTA frameworks. When we compare CAKE with methods such as FixBi and CoVi which have access to the source domain during adaptation, we see that we outperform them by 0.8% on average. We attribute this to CAKE's methodology of leveraging the attention-based architecture for the adaptation process. Despite DINE leveraging ViT for the preliminary adaptation step, we still outperform DINE by 0.7% on average. Since we distill during the adaptation process, and the teacher model improves on the target data, we have much better pseudolabels used for adapting the student model.

Table 6.2: Comparison on Office-31. Accuracy comparison between the teacher and student models of the CAKE architecture.

CAKE Model	A→D	A→W	D→A	D→W	W→A	W→D	Mean
Teacher (ConvNeXt)	93.2	95.1	82.1	98.7	83.8	99.8	92.1
Student (ResNet-50)	96.6	96.9	80.0	99.3	82.9	100.0	92.6

In order to understand the performance drop of knowledge distillation, we contrast the accuracies of the teacher and student model in Table 6.2. We see in Table 6.2 that the student has surpassed the teacher, possibly due to the bigger model overfitting on the source data. We observe that before adaptation, the teacher performs better than the student in most domain transfers. The student leverages better generalization of the teacher to get a step up by utilizing better labels from the teacher during distillation. We conjecture that increased performance comes from the information maximization loss, which reduces ambiguity in the output label distribution.

6.1.2 Results for OfficeHome Dataset

In this section, we compare the performance of the CAKE framework against the SOTA methods on the OfficeHome dataset. The CAKE architecture uses a continual batch size of 128, and a buffer size of 520 (8 samples per class), and uses the top 3 class predictions for distillation.

Table 6.3: CAKE results and comparison to state-of-the-art on the OfficeHome dataset. An * denotes the SOTA adversarial unsupervised domain adaptation methods that require access to both source and target domains during adaptation. Bold indicates the top performance, while the underlined values indicate the second-best performance.

Method	Target	Ar→Cl	Ar→Pr	Ar→Rw	Cl→Ar	Cl→Pr	Cl→Rw	Pr→Ar	Pr→Cl	Pr→Rw	Rw→Ar	Rw→Cl	Rw→Pr	Mean
FixBi*	Full	58.1	77.3	80.4	67.7	79.5	78.1	65.8	57.9	81.7	76.4	62.9	86.7	72.7
CoVi*	Full	58.5	78.1	80.0	68.1	80.0	77.0	66.4	60.2	82.1	76.6	63.6	86.5	73.1
SHOT	Full	57.1	78.1	81.5	68.0	78.2	78.1	67.4	54.9	82.2	73.3	58.8	84.3	71.8
DINE	Full	64.9	**87.4**	**88.8**	**80.5**	**89.6**	**87.8**	**79.0**	62.9	**89.1**	**81.5**	64.6	90	80.5
SHOT	Cont.	49.3	71.0	75.0	59.9	70.1	70.2	58.7	47.2	76.7	69.4	54.0	79.6	65.1
ConDA	Cont.	54.9	75.2	79.4	65.9	75.3	77.0	64.5	53.5	80.0	73.0	55.9	81.8	69.7
CAKE	Cont.	**72.9**	85.6	86.6	79.1	86.2	86.7	76.6	**70.0**	86.9	80.2	**71.6**	**90.3**	**81.1**

In the OfficeHome dataset, as seen in Table 6.1, CAKE outperforms all other continual methods by a significant margin by leveraging knowledge distillation. It should be noted that with the OfficeHome dataset, domain transfers to Clipart perform the worst among all architectures. Even though our framework does not match SOTA in all categories compared to frameworks utilizing the full target domain, we outperform

all other frameworks when adapting to Clipart by more than 6%. Despite only having access to one batch of target data at a time, CAKE outperforms DINE by 0.6% on mean accuracy. Another observation that Table 6.1 also shows is that even without using source data or an intermediate domain, we outperform the adversarial methods by 8% on average.

We compare the accuracies of the teacher and the student to understand the distillation process and how much performance could be affected during distillation. These results are presented in Table 6.4. We see that, despite ConvNeXt being 3 times the size of ResNet, its performance decreases only by 3% when adapting to the OfficeHome dataset.

Table 6.4: Comparison on OfficeHome. Accuracy comparison between the teacher and student models of the CAKE architecture.

CAKE	Ar→Cl	Ar→Pr	Ar→Rw	Cl→Ar	Cl→Pr	Cl→Rw	Pr→Ar	Pr→Cl	Pr→Rw	Rw→Ar	Rw→Cl	Rw→Pr	Mean
Teacher (ConvNeXt)	75.3	88.6	89.6	83.6	87.3	88.8	82.0	74.1	89.7	84.4	75.0	91.3	84.1
Student (ResNet-50)	72.9	85.6	86.6	79.1	86.2	86.7	76.6	70.0	86.9	80.2	71.6	90.3	81.1

6.1.3 Results for DomainNet-126 Dataset

In this section, we compare the performance of the CAKE framework against the SOTA methods on the DomainNet-126 dataset. In these experiments, we utilize an incoming batch size of 256, and a buffer size of 1024 (8 samples per class), and use the top 3 class predictions for distillation.

Table 6.5: CAKE results and comparison to the state-of-the-art on the DomainNet dataset. Bold indicates the top performance, while the underlined values indicate the second-best performance.

Method	Target	C→S	P→C	R→C	R→P	R→S	S→P	Mean
SHOT	Full	60.0	68.5	68.2	66.4	58.3	63.2	64.1
AdaContrast	Full	58.0	68.6	70.2	69.8	61.5	65.9	65.7
AdaContrast	Cont.	53.4	60.8	61.1	66.9	54.5	62.7	59.9
T^3 AR	Cont.	60.9	66.8	70.2	70.0	59.8	64.1	65.3
DePT-G	Cont.	<u>72.5</u>	<u>74.2</u>	<u>74.8</u>	<u>78.3</u>	<u>68.5</u>	<u>77.2</u>	<u>74.3</u>
CAKE	Cont.	**74.4**	**79.3**	**79.4**	**79.6**	**74.4**	**77.3**	**77.4**

In the DomainNet-126 dataset, as seen in Table 6.5, CAKE outperforms AdaContrast's continual and full modes by a significant margin by leveraging an attention-based architecture as opposed to a secondary ResNet model. AdaContrast uses a FIFO architecture to store incoming samples but only uses those samples for clustering and generating pseudolabels. As it does not adapt to those images, the continual paradigm's data constraint negatively impacts AdaContrast despite having access to those samples. By using the memory replay, CAKE can increase the amount of data it uses to adapt.

The DePT-G framework utilizes a Vision transformer backbone for adaptation and testing, however, CAKE significantly outperforms them despite using a weaker testing backbone. We attribute this to our mitigation of the data dependency through memory replay. Additionally, we compare CAKE to T^3 AR, since it also uses memory replay to mitigate the more difficult data paradigm. CAKE outperforms T^3 AR by a significant margin because of leveraging the generalization capability of the more powerful architecture to generate better labels for adaptation.

We compare the performance of the teacher and the student to understand the distillation process and the effects of the model compression. These comparisons are

presented in Table 6.6.

Table 6.6: Comparison on DomainNet-126. Accuracy comparison between the teacher and student models of the CAKE architecture.

CAKE	C→S	P→C	R→C	R→P	R→S	S→P	Mean
Teacher (ConvNeXt)	78.1	79.5	79.3	82.1	77.2	80.9	79.5
Student (ResNet-50)	74.4	79.3	79.4	79.6	74.4	77.3	77.4

The results show that despite ConvNeXt being 3 times the size of ResNet, its performance decreases only by 2.1% when adapting to the OfficeHome dataset.

6.2 Ablation Studies

6.2.1 Ablation Studies of Teacher Models

We benchmark Swin and ViT against ConvNeXt in their capabilities as teachers on the Office-31 dataset. These results can be seen in Table 6.7.

Table 6.7: Comparing CAKE Teacher Architectures on Office-31. We use the teachers presented in the preliminary experiments and benchmark their capabilities as teachers.

CAKE Teacher	A→D	A→W	D→A	D→W	W→A	W→D	Mean
ConvNeXt	96.6	**96.9**	**80.0**	**99.3**	**82.9**	**100.0**	**92.6**
ViT	93.2	95.9	78.6	98.9	80.3	**100.0**	91.1
Swin	**97.0**	96.6	76.7	**99.3**	79.7	**100.0**	91.5

We find that ConvNeXt, similar to the preliminary experiments in Section 5, serves as the best teacher as it can generalize quickly and effectively. Additionally, since it is not a Vision Transformer architecture, it can efficiently adapt to the smaller dataset and serve as a better teacher.

As done with Office-31, we benchmark the teacher models on OfficeHome. These results can be seen in Table 6.8. The results are consistent with our preliminary

experiments that use the full target domain. Additionally, we find that ConvNeXt can more efficiently adapt to the target domain and effectively distill that knowledge to the student.

Table 6.8: Comparing CAKE Teacher Architectures on OfficeHome.

CAKE Teacher	Ar→Cl	Ar→Pr	Ar→Rw	Cl→Ar	Cl→Pr	Cl→Rw	Pr→Ar	Pr→Cl	Pr→Rw	Rw→Ar	Rw→Cl	Rw→Pr	Mean
ConvNeXt	**72.9**	**85.6**	86.6	**79.1**	86.2	**86.7**	76.6	**70.0**	**86.9**	80.2	**71.6**	**90.3**	**81.1**
ViT	67.7	85.5	84.9	77.8	86.2	85.5	**78.3**	66.7	86.8	**81.2**	69.4	89.8	80.0
Swin	69.5	85.5	**86.9**	77.3	**86.6**	86.3	74.3	65.8	**86.9**	79.8	69.0	90.1	79.8

6.2.2 Ablation Studies of r-Values

In addition to benchmarking different teachers, we examine the effect of adjusting the number of predictions extracted from the teacher for student training. We vary the number of top predictions from the teacher to the student. A value of $r = 1$ selects the top class in the prediction and smooths out the other labels, while a value of $r = \mathrm{K}$ uses the full prediction vector. The results for the CAKE framework on the Office-31 and OfficeHome datasets are shown in Table 6.9 and Table 6.10, respectively.

Table 6.9: Study on the Adaptive Label Smoothing. Results for the continual closed-set UDA on the Office-31 dataset using the CAKE framework by adjusting the r-value.

Method	A→D	A→W	D→A	D→W	W→A	W→D	Mean
CAKE ($r = 1$)	96.0	**97.0**	79.8	99.1	81.5	**100.0**	92.2
CAKE ($r = 3$)	**96.6**	96.9	**80.0**	**99.3**	**82.9**	**100.0**	**92.6**
CAKE ($r = \mathrm{K}$)	95.4	96.8	79.6	98.9	81.4	**100.0**	92.0

Table 6.10: Study on the Adaptive Label Smoothing. Results for the continual closed-set UDA on the OfficeHome dataset using the CAKE framework by adjusting the r-value.

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	Mean
CAKE ($r = 1$)	72.6	**86.1**	**87.3**	**79.3**	**86.4**	**86.7**	76.3	**70.2**	86.4	79.9	**73.2**	87.7	81.0
CAKE ($r = 3$)	**72.9**	85.6	86.6	79.1	86.2	**86.7**	**76.6**	70.0	**86.9**	**80.2**	71.6	**90.3**	**81.1**
CAKE ($r = \mathrm{K}$)	70.4	84.3	85.7	77.5	85.2	85.1	74.8	68.9	85.7	78.5	72.0	89.4	79.8

Through adjustments of the r-value, we find that CAKE where $r = 3$ helps the model to perform better. With $r = \mathrm{K}$, we find a drop in performance. We attribute this to the noisier labels of the raw output especially since we don't have access to the full target data, lowering the teacher's adaptive capabilities.

We also conduct ablation studies on the OfficeHome dataset to determine the contribution of the different components of CAKE. We note in Section 3.4 that the $\mathcal{L}_{mix}$ and $\mathcal{L}_{IM}$ are some key techniques to apply during target adaptation. The teacher model is likely to generate inaccurate or noisy labels due to the domain shift and the unlabelled target data without these structural regularizations. Therefore, we compare three main methods: CAKE (w/o $\mathcal{L}_{mix}$ and $\mathcal{L}_{IM}$), CAKE (w/o $\mathcal{L}_{mix}$), and CAKE. We test the improvement of the student when applying these techniques.

Table 6.11: Ablation Study of CAKE on OfficeHome Dataset Results for the continual closed-set UDA on the OfficeHome dataset using the CAKE framework by testing the different components.

CAKE 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	Mean
w/o $\mathcal{L}_{mix}$ & $\mathcal{L}_{IM}$	70.0	84.7	85.4	76.8	85.0	85.2	73.5	67.8	85.1	77.5	70.8	84.8	78.9
w/o $\mathcal{L}_{mix}$	70.6	84.6	85.7	77.3	85.3	85.2	75.0	68.3	85.7	78.3	**71.6**	89.8	79.8
CAKE w/ $\mathcal{L}_{f_t}$	**72.9**	**85.6**	**86.6**	**79.1**	**86.2**	**86.7**	**76.6**	**70.0**	86.9	**80.2**	**71.6**	**90.3**	**81.1**

In 11 out of 12 domain tasks, the performance of CAKE improves when using $\mathcal{L}_{mix}$. Without $\mathcal{L}_{IM}$, the accuracy drops even more. This verifies the importance of those components which help to adapt the model with the noisy target labels.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This thesis presents CAKE, a novel framework for continual domain adaptation through knowledge distillation. CAKE improves upon existing source-free and continual domain adaptation methods by leveraging knowledge distillation to extract better labels when adapting to the target domain. We demonstrate that using a strong teacher as well as distillation with structural regularization can significantly improve the performance of domain adaptation tasks. Despite not having access to the full target domain at once or access to the source domain, we demonstrate that CAKE can outperform other frameworks by leveraging an attention-based architecture. We demonstrate how to leverage the feature extraction and generalization capabilities of attention-based architectures and use that to better adapt a ResNet model. Our results show that CAKE outperforms SOTA UDA methods on various datasets at a fraction of the data storage requirements. By improving the accuracy of these domain adaptation tasks, we are able to continue expanding the application areas.

7.2 Future Work

While it achieved state-of-the-art results, CAKE can be expanded with newer models as teachers and smaller models as students. With the current progression of computer vision research, novel models and architectures are being released swiftly. A future study looking into new architectures may reveal a significant increase in performance and efficiency.

In this work, we do not consider partial-set UDA where all the classes seen in the source domain may not be found in the target domain. Within the continual adaptation process, we identify the buffer as a space for improvement as it can store samples in the buffer over various incoming batches. ADAContrast [29] proposes using a FIFO architecture to continually receive samples to extract pseudolabels. This can be applied to the buffer to store samples, which will increase the amount of data the models learn. Additionally, the condition for storing samples in the buffer is a possibility for improvement. By applying techniques from noisy label learning [39], the buffer can select samples that the student is not confident about and the teacher is confident of. This can better leverage the teacher network as it results in better samples and labels for the student to adapt.

Bibliography

[1] Phung and Rhee, "A high-accuracy model average ensemble of convolutional neural networks for classification of cloud image patches on small datasets," *Applied Sciences*, vol. 9, p. 4500, 10 2019.

[2] M. Gurucharan, "Basic cnn architecture: Explaining 5 layers of convolutional neural network," *URL: https://www. upgrad. com/blog/basic-cnn-architecture*, 2020.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[5] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022.

[6] S. Herath, M. Harandi, and F. Porikli, "Learning an invariant hilbert space for domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3845–3854.

[7] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan, "Deep hashing network for unsupervised domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5018–5027.

[8] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang, "Moment matching for multi-source domain adaptation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1406–1415.

[9] I. Kuzborskij and F. Orabona, "Stability and hypothesis transfer learning," in *International Conference on Machine Learning*. PMLR, 2013, pp. 942–950.

[10] J. Liang, D. Hu, and J. Feng, "Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation," in *International Conference on Machine Learning*. PMLR, 2020, pp. 6028–6039.

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[13] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[14] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

[15] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 11 976–11 986.

[16] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[17] X. Cheng, Z. Rao, Y. Chen, and Q. Zhang, "Explaining knowledge distillation by quantifying the knowledge," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 12 925–12 935.

[18] Q. Guo, X. Wang, Y. Wu, Z. Yu, D. Liang, X. Hu, and P. Luo, "Online knowledge distillation via collaborative learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 020–11 029.

[19] J. Yang, X. Peng, K. Wang, Z. Zhu, J. Feng, L. Xie, and Y. You, "Divide to adapt: Mitigating confirmation bias for domain adaptation of black-box predictors," *arXiv preprint arXiv:2205.14467*, 2022.

[20] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9729–9738.

[21] P. Panareda Busto and J. Gall, "Open set domain adaptation," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 754–763.

[22] S. Cui, S. Wang, J. Zhuo, C. Su, Q. Huang, and Q. Tian, "Gradually vanishing bridge for adversarial domain adaptation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 12 455–12 464.

[23] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7167–7176.

[24] J. Gama, *Knowledge discovery from data streams*. CRC Press, 2010.

[25] J. Gama, R. Sebastiao, and P. P. Rodrigues, "On evaluating stream learning algorithms," *Machine learning*, vol. 90, pp. 317–346, 2013.

[26] A. M. N. Taufique, C. S. Jahan, and A. Savakis, "Conda: Continual unsupervised domain adaptation," *arXiv preprint arXiv:2103.11056*, 2021.

[27] Y. Ganin, 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*, vol. 17, no. 1, pp. 2096–2030, 2016.

[28] J. Liang, D. Hu, Y. Wang, R. He, and J. Feng, "Source data-absent unsupervised domain adaptation through hypothesis transfer and labeling transfer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 11, pp. 8602–8617, 2021.

[29] D. Chen, D. Wang, T. Darrell, and S. Ebrahimi, "Contrastive test-time adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 295–305.

[30] J. Liang, D. Hu, J. Feng, and R. He, "Dine: Domain adaptation from single and multiple black-box predictors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 8003–8013.

[31] D. Ji and M. A. Wilson, "Coordinated memory replay in the visual cortex and hippocampus during sleep," *Nature neuroscience*, vol. 10, no. 1, pp. 100–107, 2007.

[32] W. Hu, T. Miyato, S. Tokui, E. Matsumoto, and M. Sugiyama, "Learning discrete representations via information maximizing self-augmented training," in *International conference on machine learning*. PMLR, 2017, pp. 1558–1567.

[33] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," in *Advances in Neural Information Processing Systems*, L. Saul, Y. Weiss, and L. Bottou, Eds., vol. 17. MIT Press, 2004. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2004/file/96f2b50b5d3613adf9c27049b2a888c7-Paper.pdf

[34] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[35] V. Verma, K. Kawaguchi, A. Lamb, J. Kannala, A. Solin, Y. Bengio, and D. Lopez-Paz, "Interpolation consistency training for semi-supervised learning," *Neural Networks*, vol. 145, pp. 90–106, 2022.

[36] J. Liang, D. Hu, and J. Feng, "Domain adaptation with auxiliary target domain-oriented classifier," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 16 632–16 642.

[37] R. Wightman, "Pytorch image models," https://github.com/rwightman/pytorch-image-models, 2019.

[38] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, pp. 400–407, 1951.

[39] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari, "Learning with noisy labels," *Advances in neural information processing systems*, vol. 26, 2013.