Continual Domain Incremental Learning during Test-time

A Thesis

submitted to
Indian Institute of Science Education and Research Pune
in partial fulfillment of the requirements for the
BS-MS Dual Degree Programme

by

Goirik Chakrabarty

Indian Institute of Science Education and Research Pune
Dr. Homi Bhabha Road,
Pashan, Pune 411008, INDIA.

June, 2023

Supervisor: Dr. Soma Biswas

Certificate

This is to certify that this dissertation entitled ”Continual Domain Incremental Learning during Test-time ” towards the partial fulfilment of the BS-MS dual degree programme at the Indian Institute of Science Education and Research, Pune represents study/work carried out by Goirik Chakrabarty at Indian Institute of Science, Bangalore under the supervision of Dr. Soma Biswas, Associate Professor, Department of Electrical Engineering, IISc Bangalore, during the academic year 2023-2024.

Dr. Soma Biswas

Committee:

Dr. Soma Biswas

Dr. Leelavati Narlikar

This thesis is dedicated to my Parents

Declaration

I hereby declare that the matter embodied in the report entitled ”Continual Domain Incremental Learning during Test-time ” are the results of the work carried out by me at the Department of Electrical Engineering, Indian Institute of Science, Bangalore, under the supervision of Dr. Soma Biswas, and the same has not been submitted elsewhere for any other degree.

Goirik Chakrabarty

Acknowledgments

I would like to take this opportunity to express my sincere gratitude to all those who have supported me throughout my master's thesis journey.

First and foremost, I would like to thank my institute Indian Institute of Science, Education and Research, Pune for providing me with the necessary resources and environment to conduct my research. The guidance and support from my thesis advisors, Prof Soma Biswas and Prof Leelavati Narlikar, have been instrumental in shaping my research work. I am grateful for their invaluable insights, constructive feedback, and constant encouragement throughout my thesis.

I am also indebted to my PhD mentors, Jayateja Kalla and Manogna Sreenivas, for their invaluable advice, guidance, and support throughout my research work. Their expertise and knowledge have been crucial in helping me navigate through the various challenges in my research.

I would also like to express my gratitude to my friends, Purva Parmar and Khushboo Jain, for engaging in various discussions with me and for providing their invaluable assistance in proofreading this work.

My heartfelt thanks go to my friends and family, who have been a constant source of support and encouragement. Their love and encouragement have been a great motivation for me to pursue my research work with dedication and enthusiasm.

Thank you all for your support and encouragement.

Abstract

This thesis focuses on the problem of continual test time domain adaptation in deep learning, where a trained model needs to adapt to new and changing environments during deployment. The first contribution of this work is the development of a novel strategy for obtaining a signal for domain shift, which enables the model to overfit without compromising its ability to adapt to future domains. The second contribution is the presentation of a novel framework called SATA, which uses self-knowledge distillation and contrastive learning to adapt a pre-trained model to continual domain shift. The proposed framework improves the accuracy, time complexity, space complexity, and stability of the machine learning model. The research conducted in this thesis contributes to the ongoing effort to develop more robust and reliable deep learning models that can adapt to new and changing environments.

Contents

Introduction

Deep learning has achieved phenomenal success in several computer vision tasks like classification, object detection, segmentation, etc [1, 2, 3, 4, 5]. However, deep neural networks have been shown to perform poorly when tested on data from a different distribution [6] than the training data. Unsupervised Domain Adaptation (UDA) techniques have been developed to address this challenge. However, UDA techniques require access to labelled source data and unlabelled target data, which can be difficult due to various constraints such as privacy concerns and storage limitations.

Moreover, in addition to having a different distribution, the test data distribution may dynamically vary with time. For instance, in the context of autonomous driving, a model trained using data captured in clear weather may encounter changing weather conditions, such as cloudy weather, heavy rain, etc., during deployment. In such cases, it is critical to continually adapt the model during test time to ensure optimal performance in changing scenarios. Test-time adaptation of trained models has thus emerged as an important research area, where an off-the-shelf trained model is adapted to the testing data as and when they are encountered. The majority of successful frameworks for this task, [7, 8], assume that the test data belongs to a single domain, a restrictive assumption for practical applications. Researchers have recently started looking at the continual test time adaptation setting [9], where the target distribution can change over time.

Continual adaptation during test time is particularly important in real-world scenarios where the data distribution can be dynamic and challenging to predict. Thus, developing models that can adapt to new and changing environments is essential. Ongoing research is focused on developing models and algorithms that can learn to adapt and generalise to different environments, aiming to create more robust and reliable deep learning models.

In this thesis, we present two novel ideas in the scope of the problem statement of the continual

test time domain adaptation.

First, a novel strategy for getting a signal for domain shift. We observe that test time adaptation algorithms are designed to adjust a model's predictions during inference to improve its accuracy on new data. However, if these algorithms are trained on a specific domain, they may become too specialized and lose their ability to adapt to new or changing data. This is known as overfitting, and it can be difficult to reverse or undo. This observation prompted the need for a signal which can reset the adapting model to the initial model. This allows the model to overfit without the risk of performing badly in future domains.

Next, we present a novel framework Source Anchoring and Target Alignment (SATA) which utilises self-knowledge distillation and contrastive learning to adapt a pre-trained model to continual domain shift. We observe that our framework not only performs better in terms of accuracy but also improves the time complexity, space complexity and stability of the machine learning model.

We begin the thesis by introducing a relevant literature review of the field. Then we present the above two ideas in chapters 2 and 3. Both these chapters are written as self-contained scientific articles and include their own introduction, methods and results.

Chapter 1

Background Theory

In this section, we will provide an overview of the general topics closely related to this dissertation. Specifically, we will introduce the concept of domain adaptation, its variants which relax many of its assumptions and the relevant methods developed in the literature.

1.1 Related Works

1.1.1 Domain Adaptation

Domain adaptation is a subfield of machine learning that focuses on improving the performance of models when there is a shift in the distribution of the data between the training and test datasets. The goal of domain adaptation is to train a model on a source domain with labelled data and adapt it to a target domain with unlabelled data, where the distributions of the two domains differ. We aim to learn from additional target data and thereby alleviate the domain shift and improves model performance for target data. Common setups for domain adaptation are as follows:-

- Supervised domain adaptation is a type of domain adaptation where some labelled data from the target domain is available along with the labelled data from the source domain. The goal is to learn a model that can generalise well to the target domain using both the source and target labelled data.

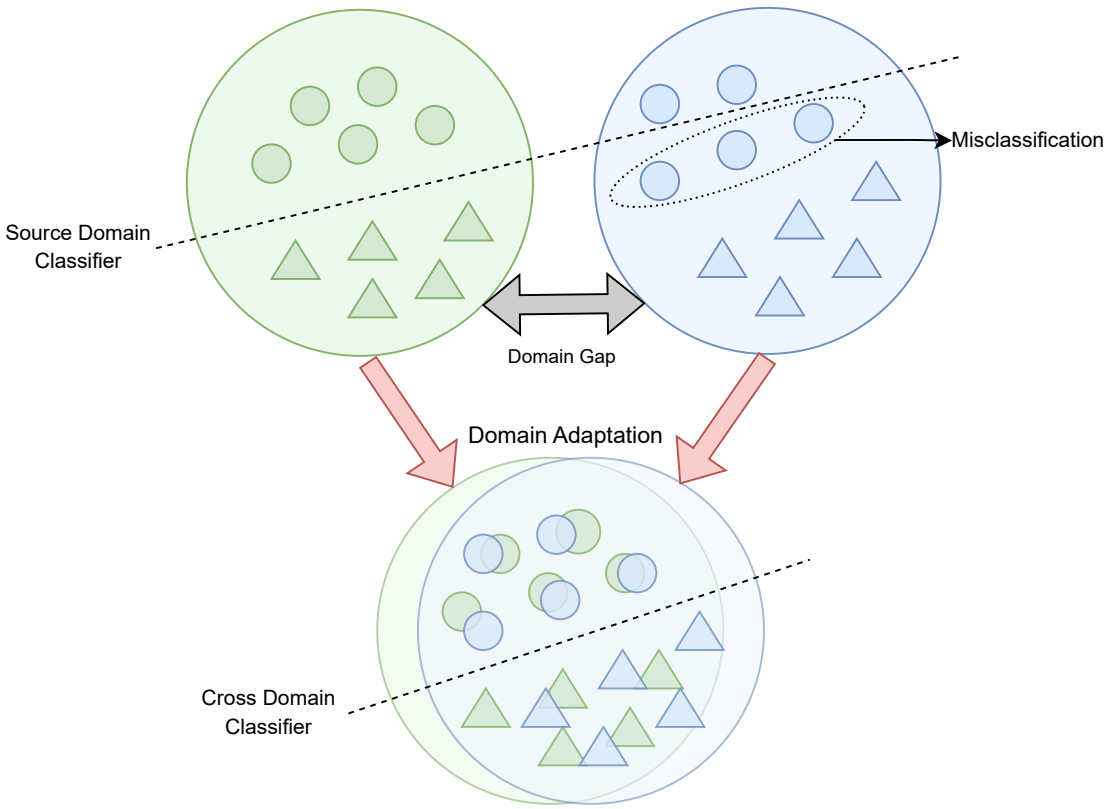


Figure 1.1: Schematic to demonstrate the process of source and target domain alignment for domain adaptation. The green domain is the source domain, and the blue domain is the target domain.

- Semi-supervised domain adaptation is a type of domain adaptation where only a small amount of labelled data from the target domain is available, along with the labelled data from the source domain. The goal is to learn a model that can make use of the limited target labelled data and generalise well to the target domain using both the source and target data.

- Unsupervised domain adaptation, as mentioned earlier, is a type of domain adaptation where no labelled data from the target domain is available. Unsupervised domain adaptation methods aim to align the source and target domain distributions and learn a model that can generalise well to the target domain using only the source domain data and the unlabelled target domain data.

1.1.2 Unsupervised Domain Adaptation

In this dissertation, we will be focusing on Unsupervised Domain Adaptation, more specifically its test time-variant as it is a more realistic setup as acquiring labelled data from the target domain can be expensive and time-consuming. Formally, we define the task of UDA as follows:

Given,

- Labelled data from the source domain,

$$\mathcal{D}_s = \{(x_i, y_i)\}_{i=1}^{n_s} \sim P_s \tag{1.1}$$

- Unlabelled data from the target domain,

$$\mathcal{D}_t = \{(x_i)\}_{i=1}^{n_s} \sim P_t \tag{1.2}$$

here $x_i \in X$ is the input data (images in our case), and $y_i \in Y$ is the ground truth (semantic labels in the case of image classification). As the distributions between the two domains $\mathcal{D}_s$ and $\mathcal{D}_t$ are assumed to be different but still similar up to the content level, it is necessary to align them in order to minimise the domain discrepancy and achieve good performance on the target domain.

There are various methods to achieve this alignment. The Deep Adaptation Network (DAN) [10] method, for example, minimises the Maximum Mean Discrepancy (MMD) [11] between the source and target domain distributions in the intermediate feature space. Adaptive Batch Normalization (AdaBN) [12, 13] and TransNorm use normalization layers to achieve the alignment, while Domain-Adversarial Training of Neural Networks (DANN) [14] uses a discriminator to align the distributions in an adversarial way.

Moreover, some methods take class information into account while aligning the distributions. For instance, [15] improve the adversarial alignment method by considering class information.

Self-training methods, on the other hand, implicitly align the distributions by making use of pseudo-labels generated by models from previous iterations. [16, 17] are some examples of self-training methods that have shown promising results in domain adaptation tasks.

1.1.3 Source-free Domain Adaptation

Source-free domain adaptation (SFDA) is a variant of domain adaptation in which the model is adapted to the target domain without access to any samples from the source domain. In other words, SFDA assumes that only the source-trained model is available for adapting to the labelled target distribution.

SFDA is a more challenging problem than traditional domain adaptation, as it requires the model to learn to recognise the domain shift using only the target domain data. Although it is possible to train on the test data in a fresh manner, it would be inefficient. Ideally, we want to use the already learnt semantic information of the source distribution. So, the challenge is to leverage source information while learning about the target domain. Various methods have been proposed to address SFDA, including deep clustering-based approaches, self-supervised learning, and unsupervised domain adaptation methods.

One of the key advantages of SFDA is that it eliminates the need for data from the source domain, which is often difficult or expensive to obtain and sometimes just unobtainable. Additionally, SFDA has the potential to generalise better to new target domains that were not seen during training, as the model is adapted to the domain shift rather than being optimised for a specific source-target domain pair. However, getting labelled target data can also be very expensive. Therefore, a variant of SFDA is SFUDA (Source free Unsupervised Domain Adaptation) which aims to adapt to a target domain using unlabelled target data.

1.1.4 (Continual) Test time Adaptation

Test-time Adaptation (TTA) is an online variant of SFUDA that adapts the model at test time using small batches of test data, as and when they become available. Here, the model's parameters or architecture are usually adjusted to handle the differences between the two domains better, thereby improving its performance on the target domain [7, 18, 8, 19, 13]. Some of these methods focus on modifying the original architecture during the source training like TTT [20], which trains the model on supervised and self-supervised tasks using source data. During testing, the self-supervised module is fine-tuned on the target data to improve performance. Recently, several researchers are focusing on the fully test time adaptation setting [7, 13], which does not assume any access to source data or the source training process making it more practical. TENT [7] adopts

entropy minimisation objective for training the BN layers, while BNStatsAdapt [13] adjusts the BN statistics during test time to align the target with the source domain.

A more realistic scenario is handled by the recently proposed continual test-time adaptation protocol [9], where the trained model should continually adapt to a dynamic environment, where the test domain can change over time. CoTTA [9] utilises weight-averaged and augmentation-averaged predictions to reduce error accumulation and also stochastically restores a small part of the neurons to the source pre-trained weights during each iteration to avoid catastrophic forgetting. This allows for long-term adaptation of all parameters in the network while preserving source knowledge. Recently, several modules have been developed which can aid the dynamic adaptation to test-data. However, these modules need to be optimised along with the model during training with source [21, 22].

Setting	Source-free	Adaptation protocol		Target domain	
		Offline	Online	Single	Continuous
UDA		✓		✓	
SFDA	✓	✓		✓	
TTA	✓		✓	✓	
CTTA	✓		✓		✓

Table 1.1: Domain adaptation protocols

1.2 Problem Setting and Motivation

Continual test time adaptation is a machine learning technique where a model is continually adapted during inference, or test time, to improve its performance. In other words, instead of training a model once and using it as is for all predictions, the model is updated in real-time as it makes predictions based on new data. Continual test time adaptation is particularly useful in situations where the distribution of the data changes over time. For example, in a recommendation system, user preferences may change over time, so the model needs to adapt to these changes to continue making accurate recommendations.

Formally, we are given a model trained using source domain data $\mathcal{D}_s = \{(x_i, y_i)\}_{i=1}^{n_s} \sim P_s$. Here, (x_i, y_i) is source data and label which is drawn from the source distribution P_s. During testing, the source data is usually not available due to privacy concerns or storage constraints. At

this stage, the model encounters test data $\mathcal{D}_t = \{x_j\}_{j=1}^{n_t} \sim P_t$. In practice, the test data can belong to a different domain compared to the source, i.e. $P_t \neq P_s$. Further, P_t can change over time such that $P_t^{(1)} \neq P_t^{(2)} \neq \ldots \neq P_s$ leading to our continual test time adaptation scenario.

1.3 Benchmark Dataset

The datasets that are generally used for testing the performance of domain adaptation and test time adaptation algorithms in the thesis are CIFAR10-C, CIFAR100-C and ImageNet-C [23]. The "C" in these datasets stand for corruption. These dataset are synthetically created by transforming the test set of CIFAR10, CIFAR100 and ImageNet using 15 different corruption which can be categorised in 4 types Noise, Blur, Weather, Digital. The 15 corruptions are, Gaussian Noise, Shot Noise, Impulse Noise; Defocus Blur, Glass Blur, Motion Blur, Zoom Blur; Snow, Frost, Fog, Brightness; Contrast, Elastic Tansform, Pixelate, Jpeg. We can see the corruption in Figure 1.2.

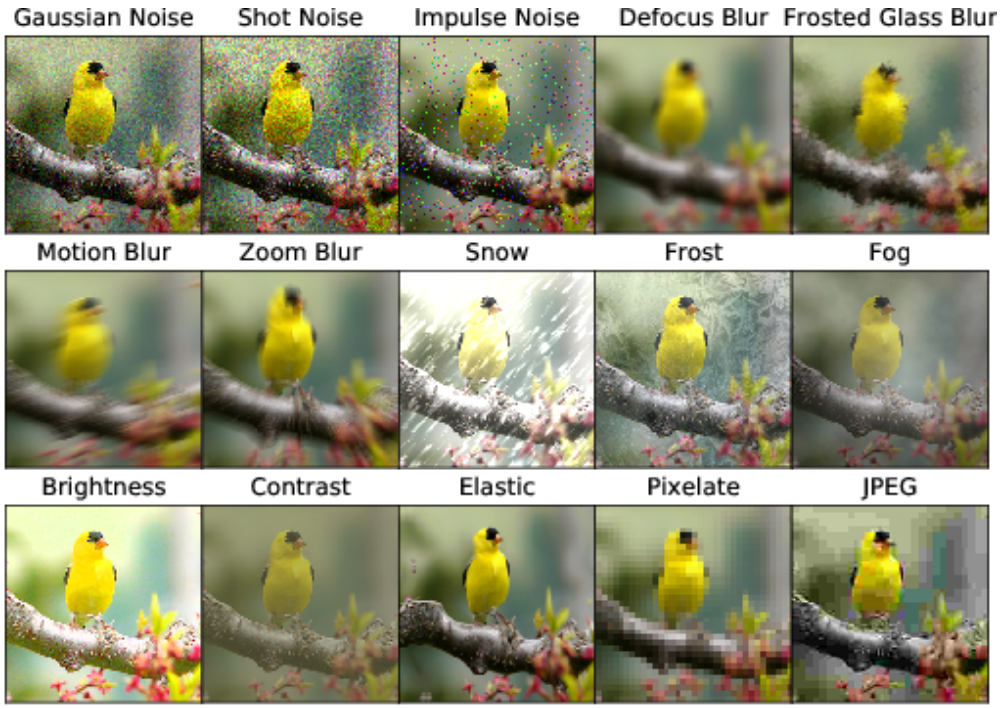


Figure 1.2: Different corruptions in ImageNet-C on which various Domain adaptation algorithms are benchmarked.

Chapter 2

A Simple Signal for Domain Shift

2.1 Introduction

The ability to continually adapt models in real-time is becoming increasingly important in today's fast-paced technological landscape. The traditional approach of single-domain test-time adaptation, while effective in certain scenarios, can limit the performance of models when deployed in dynamic and ever-changing environments.

This research broadly operates under a stringent assumption that the training and testing data come from the same distribution. This assumption can be problematic when there is a significant difference between the distribution of the training data and the distribution of the testing data, a phenomenon known as a "domain shift". This can result in reduced accuracy and performance of the model, as it has not been trained on data from the testing distribution. To mitigate this vulnerability, various domain adaptation techniques have been developed to make the models more robust to such shifts. These techniques aim to align the distributions of the training and testing data, reducing the negative impact of the domain shift on model performance.

In this work, we specifically question the differences between Test-time Adaptation (TTA) and Continual Test-time Adaptation (CTTA) and aim to bridge the gap between the two.

Why TTA can hurt CTTA? TTA methods designed for single domain adaptation tend to overfit on the current test domain which can lead to catastrophic forgetting of discriminative infor-

mation from source in time. This can be extremely harmful when the model could encounter new test domains in the future.

Can we simulate TTA setting in CTTA? We recognise that a simplistic approach to CTTA is to adapt to the test domain in a TTA manner i.e. adapt the model using a TTA algorithm and then reset the model back to the source model everytime it encounters a domain shift. This allows the model to learn representations by leveraging the benefits of single domain TTA and at the same time avoid error accumulation in time by not carrying over an overfit model to the next domain.

Restating out problem statement, we have an off-the-shelf model $h_{\boldsymbol{\theta}}$ comprising of feature extractor f and classifier g trained on a source domain $\mathcal{D}_{train}$, the objective of TTA is to adapt $h_{\boldsymbol{\theta}}$ using test batches $\mathbf{x}_t$ arriving in an online manner from a test domain $\mathcal{D}_{test}$ by minimizing a test time objective as

$$\arg\min_{\theta} \mathcal{L}_{test}(\mathbf{x}_t; \theta) \tag{2.1}$$

In standard TTA addressed in [7, 8, 24], $\mathbf{x}_t$ comes from a single test domain $\mathcal{D}_{test} \neq \mathcal{D}_{train}$. Here, we address the CTTA setting, where the test domain $\mathcal{D}_{test}$ can continuously change sequentially as $\mathcal{D}_{t,1}, \mathcal{D}_{t,2}, \mathcal{D}_{t,3}, ..., \mathcal{D}_{tN}$, where $\mathcal{D}_{t,i} \neq \mathcal{D}_{train} \forall i$.

2.2 Method

We first briefly describe some recent source-free adaptation methods, namely Tent [7] and AaD [25]. Then, we discuss the concept of Maximum mean discrepancy. Finally, we describe our Domain Shift Detection mechanism in detail.

TENT: Tent is a seminal work, which first proposed the TTA setting to online adapt any given off-the-shelf model $h_{\boldsymbol{\theta}}$. Firstly, Tent proposes to use the test feature statistics in the Batch Normalization (BN) layers instead of those estimated using the source data. Further, they fine-tune the BN's affine parameters to minimise the Shannon entropy of the test predictions, as they observe that test entropy is correlated with the test error. For a test sample x_t,

$$\mathcal{L}_{ent}(x_t) = -\sum_{c} p_c \log p_c \tag{2.2}$$

Attracting and Dispersing (AaD): This [25] is a simple and effective approach recently proposed

for SFDA. They treat SFDA as an unsupervised clustering problem where they enforce consistency between predictions of local neighbourhood features while also ensuring diversity in the feature space. The test objective for a sample x_i from a test batch $\mathbf{x}_t$ is

$$\mathcal{L}_{AaD}(\mathbf{x}_t) = \mathbf{E}_{x_i \in \mathbf{x}_t} \mathcal{L}(x_i); \qquad \text{where} \qquad \mathcal{L}(x_i) = -\sum_j p_i^T p_j + \lambda \sum_{m \in \mathbf{x}_t} p_i^T p_m \tag{2.3}$$

where p_k refers to the softmax prediction vector of the sample $x_k \in \mathbf{x}_t$.

The above mentioned methods achieve state-of-the-art performance in single domain adaptation setting. However, these methods suffer from error accumulation due to over-fitting in CTTA. We observe that source model is a more reliable starting point for adaptation than continually adapting. This is because the source model has already been trained on a large amount of data, and it has learned some general representations that can be transferred to the new domain. By adapting the source model on the new domain, the model can adjust its representations to better fit the new data while retaining the knowledge learned from the source domain.

2.2.1 Maximum Mean Discrepancy

Maximum mean discrepancy (MMD) can be defined as the distance (difference) between feature means. Let's start with the concepts used in the definition of feature means. Firstly given an $\mathcal{D}_{t,i}$, a feature map ϕ maps $\mathcal{D}_{t,i}$ to an another space $\mathcal{F}$ such that $\phi(\mathcal{D}_{t,i}) \in \mathcal{F}$. Assuming $\mathcal{F}$ satisfies the necessary conditions, and we can compute the inner product in $\mathcal{F}$:

$$\mathcal{D}_{t,i}, \mathcal{D}_{t,j} \text{ such that } \; d(\mathcal{D}_{t,i}, \mathcal{D}_{t,j}) = \langle \phi(X), \phi(Y) \rangle_{\mathcal{F}}$$

Feature means: Given a probability measure P on $\mathcal{D}_{t,i}$, feature means (or feature prototypes called in the literature) is another feature map that takes $\phi(\mathcal{D}_{t,i})$ and maps it to the means of every coordinate of $\phi(\mathcal{D}_{t,i})$:

$$\mu_p\left(\phi(\mathcal{D}_t)\right) = \left[\mathbf{E}[\phi(\mathcal{D}_{t,i}], \cdots, \mathbf{E}[\mathcal{D}_{t,N}]\right]^T \tag{1}$$

Inner product of feature means of $\mathcal{D}_{t,i} \sim P$ and $\mathcal{D}_{t,j} \sim Q$ can be written in terms of kernel

function such that:

$$\langle \mu_P \left(\phi(\mathcal{D}_{t,i}), \mu_Q \left(\phi(\mathcal{D}_{t,j})\right\rangle_{\mathcal{F}} = \mathbf{E}_{P,Q} \left[\langle \phi(\mathcal{D}_{t,i}), \phi(\mathcal{D}_{t,j})\rangle_{\mathcal{F}}\right] = \mathbf{E}_{P,Q} \left[d(\mathcal{D}_{t,i}, \mathcal{D}_{t,j})\right] \tag{2}$$

Maximum mean discrepancy: Given $\mathcal{D}_{t,i}, \mathcal{D}_{t,j}$ maximum mean discrepancy is the distance between feature means of $\mathcal{D}_{t,i}, \mathcal{D}_{t,j}$:

$$MMD^2(P, Q) = \|\mu_P - \mu_Q\|^2_{\mathcal{F}} \tag{3}$$

For convenience we have left out the $\phi(\cdot)$ parts. If we use the norm induced by the inner product such that $\|x\| = \sqrt{\langle x, x\rangle}$, the equation (3) becomes

$$MMD^2(P, Q) = \langle \mu_P - \mu_Q,\ \mu_P - \mu_Q\rangle = \langle \mu_P, \mu_P\rangle - 2\langle \mu_P, \mu_Q\rangle + \langle \mu_Q, \mu_Q\rangle$$

Using the equation (2), finally above expression becomes

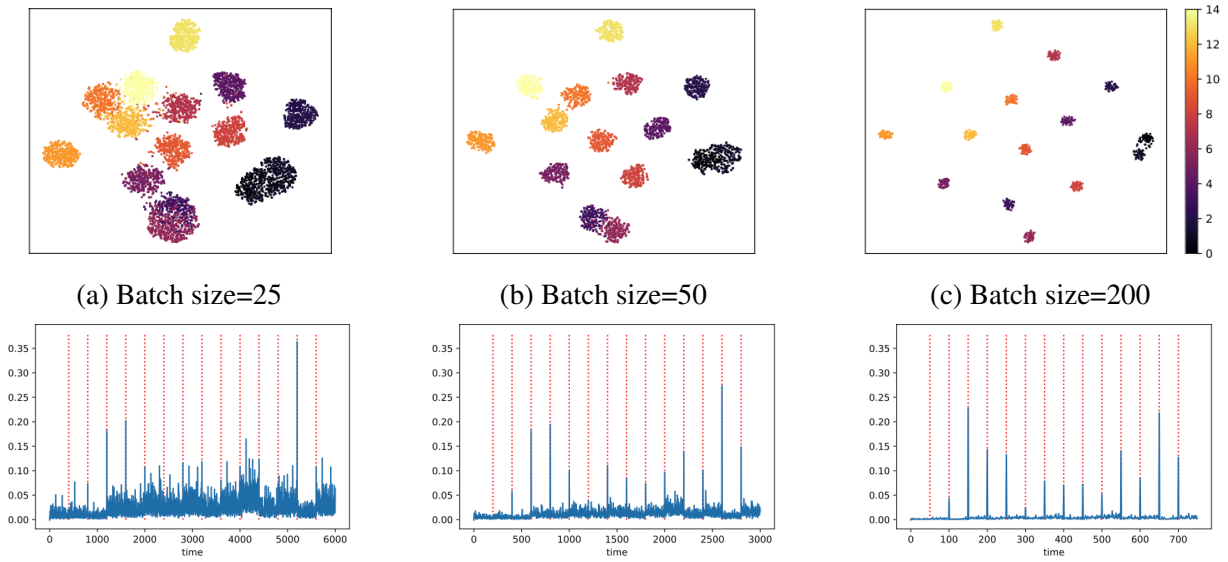
$$MMD^2(P, Q) = \mathbf{E}_P \left[d(\mathcal{D}_{t,i}, \mathcal{D}_{t,i})\right] - 2\mathbf{E}_{P,Q} \left[d(\mathcal{D}_{t,i}, \mathcal{D}_{t,j})\right] + \mathbf{E}_Q \left[d(\mathcal{D}_{t,j}, \mathcal{D}_{t,j})\right] \tag{4}$$

This is essentially the theoretical formulation that inspired us to come up with the practical implementation of our domain signal in the CTTA setting. We then use it to demonstrate our improved performance.

2.2.2 Domain Shift Detection

As mentioned earlier, using TTA methods like TENT can hurt in CTTA setting because of error accumulation. This in turn degrades the model over time. Here, we propose a simple but effective solution to this by resetting the model when a domain shift is encountered.

Can source model characterise domain shift? In CTTA, the data distribution changes over time, meaning that each batch of samples can come from a different domain. Then during inference time the domain shifts from one corruption to another. To handle this challenge, we leverage



(a) Batch size=25 (b) Batch size=50 (c) Batch size=200

(d) Batch size=25 (e) Batch size=50 (f) Batch size=200

Figure 2.1: We observe from the t-SNE plots for (a), (b) and (c) that the classes are better clustered and separated as the batch-size increases. The color of these clusters also represent the order in which 15 corruptions are seen. In (d), (e) and (f) we see the corresponding (1 - DSS) signals to the t-SNE. The red dotted lines are where the actual domain shift happens. Further 2.2 we see that this domain detection is not dependent on the order of corruptions.

the feature extractor of the source model f, which we empirically observed to capture domain information. The features of each sample $v_f = f(x)$ has two components: (i) Domain-specific component v_d which represents the part of the feature that is unique to a particular domain and distinguishes it from other domains; (ii) Class-specific component v_c that is relevant to the classification task. By separating the features into these two components, the model can learn to identify and adapt to changes in the distribution of the data between batches, while still maintaining the ability to perform well on the classification task.

We hypothesise that $\mathbf{E}(v_f) = \mathbf{E}(v_d) + \mathbf{E}(v_c)$. Given, the samples come from the same domain all sample have same domain $\mathbf{E}(v_d) = \mathbf{v}_d$, also the class specific components v_c would be uniformly spread across all classes as $\mathbf{E}(v_c) = \frac{1}{C}\sum_{k=1}^{C} v_k = \mathbf{v}_c$, where $\mathbf{v}_c$ is a constant vector and C denotes the number of classes. Hence, $\mathbf{E}(v_f) = \mathbf{v}_d + \mathbf{v}_c$. In this formulation, any change in the domain specific component $\mathbf{E}(v_d)$ can in-turn be captured by $\mathbf{E}(v_f)$, which can be empirically estimated.

In CTTA, given a test batch $\mathbf{x}_t = x_1, x_2, ..., x_N$ at time instant t, we can estimate $\mathbf{E}(v_f)(t)$

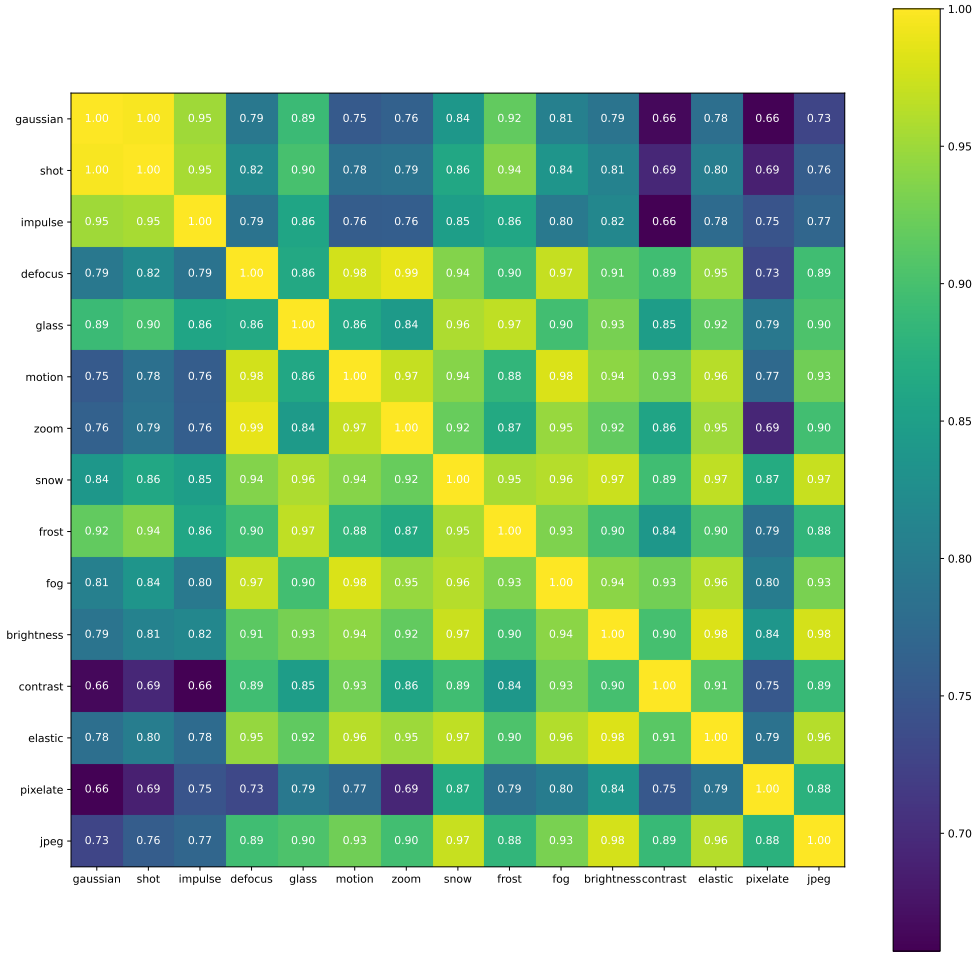


Figure 2.2: Cosine Similarity between random average batch features show that the similarity is less than the threshold between different domains thus the signal can not only be used for domain shift detection but also prompting. [26]

as the mean feature vector $\mathbf{E}(v_f)(t) = \frac{1}{N}\sum_{k=1}^{N} v_{f,i}$, where $v_{f,i} = f(x_i)$. This shows that these domain specific components can be used to identify or detect a domain shift. We empirically observe that $\mathbf{v}_c \to 0$ as $N \to \infty$. In Figure 2.1, we visualise the average batch features using different batch sizes and for 15 corruptions in the CIFAR-100C. As the batch size increases, the domain clusters become more compact, indicating the aforementioned tendency. Because of this, the domain-specific component becomes more dominant with larger batch sizes.

This naturally acts as our domain shift signal. We define the cosine similarity of consecutive batches as Domain Shift Signal (DSS), which we compute as

$$\text{DSS} = \text{CosineSimilarity}(\mathbf{E}(v_f(t)), \mathbf{E}(v_f(t-1))) \tag{2.4}$$

We us this signal to detect a change in domain using a threshold τ. When $\mathbf{E}v_f(t)$ comes from the same domain as $\mathbf{E}v_f(t-1)$, DSS is high, in turn continuing the model adaptation. Otherwise, we trigger a model reset back to the source model. We briefly describe the domain shift detection mechanism below.

Algorithm 1: Domain Shift Detection module

Input:
Source feature extractor f
Threshold for detection τ
Domain Shift Detection:
for each batch $\mathbf{x}_t$:
  $v_{f,i} = f(x_{t,i})$
  $\mathbf{E}v_f(t) = \frac{1}{N}\sum_{k=1}^{N} v_{f,i}$
  $DSS(\mathbf{E}v_f(t), \mathbf{E}v_f(t-1)) = \frac{\mathbf{E}v_f(t)^T \mathbf{E}v_f(t-1)}{||\mathbf{E}v_f(t)||||\mathbf{E}v_f(t-1)||}$
  if $DSS(\mathbf{E}v_f(t), \mathbf{E}v_f(t-1)) < \tau$:
    Reset model to source
  Continue TTA

2.3 Experiments and Results

2.3.1 Datasets

Following the protocol in [9], we use CIFAR10C and CIFAR100C [23] datasets which are designed to evaluate the robustness of classification networks. These datasets contain images that have been corrupted with 15 different types of corruption at 5 different levels of severity. In the case of the corruption benchmark, this sequence consists of all 15 corruptions, each encountered at the highest severity level 5 (maximum severity).

2.3.2 Baselines

We compare the performance of TENT and AaD in three different scenarios:

TTA: Firstly, we consider the TTA setting introduced by TENT [7] where the model is set to source whenever there is a domain shift. This domain shift information is explicitly provided to the model.

CTTA: Next, we consider the CTTA, as introduced by CoTTA [9]. Similar to the TTA setting, the continual benchmark also uses an off-the-shelf model pre-trained on the source domain. However, unlike the standard TTA setting, the continual setting does not require knowledge of when the domain changes, and instead adapts the model online to a sequence of test domains.

DSS: Finally, we use our domain shift signal to mimic the TTA setting while we are in the CTTA setting. By using the domain shift signal to dynamically set the model source even without having the underlying domain shift information. Thus the model can adapt to the changing distributions without accumulation of error, effectively mitigating the impact of domain shift.

2.3.3 Implementation details

For all the settings, TTA, CTTA and DSS we use the source model which is trained on the clean CIFAR10, CIFAR100 or ImageNet dataset. Then the algorithms are evaluated on the corruption benchmarks CIFAR10-C, CIFAR100-C or ImageNet-C[23], respectively. These datasets have 15

corruptions with severity levels ranging from 1 to 5. All experiments are conducted on the highest severity level i.e. 5. For ImageNet-C, we use the first 10,000 samples for each corruption instead of all the points in the dataset. This is the same protocol that CoTTA [9] uses for its experiments.

For the TTA task, we keep track of the domain shift and reset the model as well as the optimizer parameters to the initial states. For CTTA, we continually adapt the model and do not reset anything outside the scope of the algorithms. For DSS, we reset the model and the optimizer according to our domain shift signal.

The CIFAR10 experiments use a WideResNet-28[27] model, while the CIFAR100 experiments use a ResNeXt-29[28] architecture. The ImageNet-C experiments are done with the source model as Standard ResNet-50 trained on ImageNet. All source models are adopted from the RobustBench benchmark[29].

CoTTA[9] experiments are done using the official code base, and we use the default parameters without any further tuning as we do not change the problem set. For TENT[7], we use the code of TENT implemented in the CoTTA code base we use a learning rate of 1e-3 for all three datasets. For AaD[25], we adapt the AaD loss from its codebase. We use a learning rate of 1e-4 for CIFAR10-C and CIFAR100-C, and for ImageNet-C, we use a learning rate of 1e-7 as we observe that any higher learning rate makes AaD unstable within a single corruption. Both for TENT and AaD, only the BN-layers are learnable, and they use the mean and variance of the test batch as the BN-layer statistics.

Method	*gaussian*	*shot*	*impulse*	*defocus*	*glass*	*motion*	*zoom*	*snow*	*frost*	*fog*	*brightness*	*contrast*	*elastic*	*pixelate*	*jpeg*	Mean
Source	72.3	65.7	72.9	46.9	54.3	34.8	42.0	25.1	41.3	26.0	9.3	46.7	26.6	58.5	30.3	43.5
CoTTA	24.3	21.3	26.6	11.6	27.6	12.2	10.3	14.8	14.1	12.4	7.5	10.6	18.3	13.4	17.3	16.2
TENT-TTA	24.8	23.5	33.0	12.0	31.8	13.7	10.8	15.9	16.2	13.7	7.9	12.1	22.0	17.3	24.2	18.6
TENT-CTTA	24.8	20.6	28.6	14.4	31.1	16.5	14.1	19.1	18.6	18.6	12.2	20.3	25.7	20.8	24.9	20.7
TENT-DSS	24.8	20.6	33.0	12.0	31.8	13.7	10.8	15.9	16.2	13.7	7.9	12.1	22.0	17.3	24.2	18.4
AaD-TTA	26.7	24.8	35.0	12.4	33.9	13.8	11.5	16.7	16.9	14.4	8.2	12.5	22.8	18.7	26.2	19.6
AaD-CTTA	26.7	23.2	30.6	12.4	30.9	14.7	12.4	19.3	19.4	17.5	13.6	20.8	27.7	26.4	33.8	22.0
AaD-DSS	26.7	23.2	35.0	12.4	33.9	13.8	11.5	16.7	16.9	14.4	8.2	12.5	22.8	18.7	26.2	19.5

Table 2.1: Results as error percentages (lower is better) for CIFAR-10C

Computational Advantages: From Figure 3.4 and Table 3.8 we observe the computational advantages of using TENT-DSS or AaD-DSS compared to CoTTA. Figure 3.4 compares inference time. Here, we see that CoTTA (SoTA for CTTA setting) is computationally more expensive because (i) it needs to do 32 forward pass, (ii) update all parameters, (iii) update teacher model after

Method	*gaussian*	*shot*	*impulse*	*defocus*	*glass*	*motion*	*zoom*	*snow*	*frost*	*fog*	*brightness*	*contrast*	*elastic*	*pixelate*	*jpeg*	Mean
Source	73.0	68.0	39.4	29.3	54.1	30.8	28.8	39.5	45.8	50.3	29.5	55.1	37.2	74.7	41.2	46.4
CoTTA	40.1	37.7	39.7	26.9	38.0	27.9	26.4	32.8	31.8	40.3	24.7	26.9	32.5	28.3	33.5	32.5
TENT-TTA	37.1	34.65	33.7	25.1	37.66	27.15	25.4	30.5	31.5	33.3	23.8	27.8	32.7	28.4	36.5	31.0
TENT-CTTA	92.7	37.2	35.7	41.6	37.5	50.8	47.7	48.5	58.7	64.8	72.4	70.5	82.2	88.5	89.9	61.2
TENT-DSS	37.2	35.9	41.6	25.2	37.6	27.2	25.4	30.5	31.6	33.2	23.8	27.7	32.6	28.4	36.5	31.5
AaD-TTA	41.9	39.8	42.0	27.2	41.4	29.3	27.5	34.5	34.7	40.3	26.2	30.2	35.2	32.3	40.8	34.9
AaD-CTTA	41.9	40.1	43.5	31.7	46.8	39.2	41.6	58.2	67.7	76.2	79.1	90.1	93.0	93.8	94.6	62.5
AaD-DSS	41.9	40.1	43.5	27.2	41.4	29.3	27.5	34.5	35.0	40.3	26.2	30.2	35.2	32.3	40.8	35.0

Table 2.2: Results as error percentages (lower is better) for CIFAR-100C

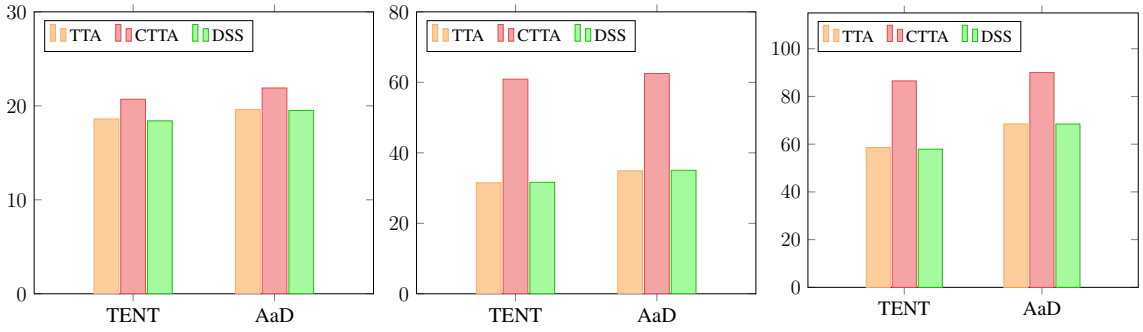
Method	*gaussian*	*shot*	*impulse*	*defocus*	*glass*	*motion*	*zoom*	*snow*	*frost*	*fog*	*brightness*	*contrast*	*elastic*	*pixelate*	*jpeg*	Mean
Source	97.8	97.1	98.2	81.7	89.8	85.2	78.0	83.5	77.1	75.9	41.3	94.5	82.5	79.3	68.6	82.0
CoTTA	83.9	79.5	76.7	79.5	76.3	67.0	57.8	62.0	59.5	50.9	40.9	62.5	49.7	44.7	48.0	62.6
TENT-TTA	73.7	71.0	72.7	74.2	74.6	60.9	52.2	54.4	58.7	43.1	32.6	74.4	46.0	42.0	48.8	58.6
TENT-CTTA	73.7	65.9	67.4	78.0	79.9	81.8	80.2	89.7	94.2	96.4	97.0	99.6	99.4	99.3	99.5	86.8
TENT-DSS	73.7	65.9	67.4	74.2	74.6	60.9	52.2	54.4	58.7	43.1	32.6	74.4	46.0	42.0	48.8	**57.9**
AaD-TTA	84.3	84.9	83.8	84.7	84.3	74.0	60.8	66.0	66.5	51.5	35.3	84.3	55.2	51.2	60.2	68.5
AaD-CTTA	84.4	84.6	83.8	83.9	86.9	82.0	75.5	89.3	95.3	93.6	93.1	99.5	99.3	99.3	99.6	90.0
AaD-DSS	84.4	84.4	83.2	84.7	84.3	74.0	60.8	66.0	66.5	51.5	35.3	84.3	55.3	51.2	60.2	68.4

Table 2.3: Results as error percentages (lower is better) for ImageNet-C

backpropagation. Next, Table 3.8 shows the memory requirements is lower in TENT and AaD compared to CoTTA due to less number of trainable parameters and models that need to be stored.

2.4 Conclusion

In this work, we propose a modular method for handling the challenge of continual test-time domain adaptation. We address the limitations of traditional single domain adaptation by developing a domain shift detection mechanism that continually measures the similarity between feature representations of consecutive batches. When a shift is detected, our method resets the model back to the source and continues test-time adaptation. Our experiments across standard datasets, batch sizes, and single domain test-time adaptation baselines demonstrate the effectiveness of our approach, making it a promising solution for the continual domain test-time adaptation problem.



(a) CIFAR-10C (b) CIFAR-100C (c) ImageNet-C

Figure 2.3: Mean error rates for CIFAR-10C, CIFAR-100C and ImageNet-C using TENT and AaD.

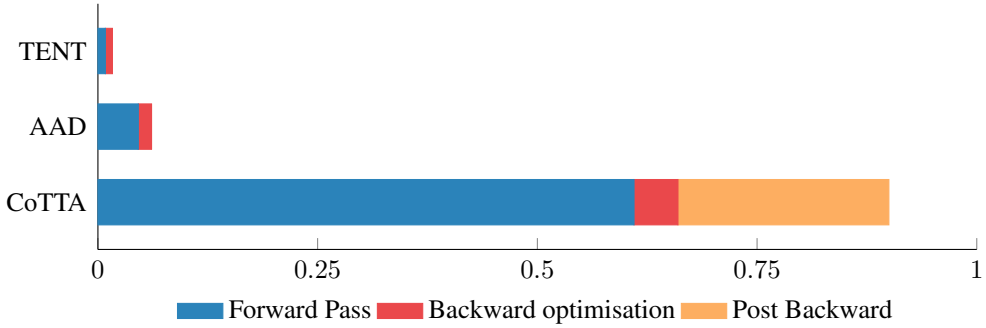


Figure 2.4: Comparison of inference time of the proposed SATA framework with the state-of-the-art CoTTA. The x-axis is time of inference per batch (sec/batch). These experiments were done on ImageNet-C using NVIDIA GeForce RTX 3090.

Method	# Parameters	# Trainable	% Trainable
TENT	25,557,032	128	0.0005
AaD	25,557,032	128	0.0005
CoTTA	76,671,096	25,557,032	33.333

Table 2.4: Number of (trainable) parameters as a proxy for the storage requirement of the respective algorithms. This table is for ImageNet-C with ResNet-50 as the backbone.

Chapter 3

SATA: Source Anchoring and Target Alignment Network for Continual Test Time Adaptation

3.1 Introduction

In this work, we propose a novel framework, termed **S**ource **A**nchored and **T**arget **A**lignment (SATA) Network, for the task of continual test-time domain adaptation. We feel that for the model to be practically useful in an online setting, it should satisfy the following requirements: *1) For online adaptation, the models should work seamlessly with different (preferably small) batch sizes which reduces the inference time and latency; 2) The updated model should continue to work well on the source domain; 3) The framework should require less storage and minimal tunable hyper-parameters, since validation sets are usually not available during test-time.* With this motivation, we propose to use source anchoring based self-distillation, which ensures that the model robustly adapts to the incoming data, while not forgetting the source domain information. The proposed SATA also utilises contrastive learning to ensure better model generalisability to unseen domains. Here, we also utilise the source prototypes for alignment of the target features to the corresponding source data, which help to conserve the semantic information learnt using the source. We propose to only update the BN affine parameters like TENT [7], which helps to avoid overfitting on the small amount of target data, in addition to reducing the storage requirements. This simple, yet

effective framework helps us take a step forward in achieving all the objectives mentioned earlier. Extensive experiments on three large-scale benchmark datasets, namely CIFAR-10C, CIFAR-100C and ImageNet-C [23] for different challenging and realistic scenarios justify the effectiveness of the proposed SATA framework. To summarize, the contributions of this work are as follows:

- We propose a novel SATA framework for the task of continual test-time domain adaptation.
- The proposed framework takes a step forward in overcoming some of the important challenges in a practical test-time adaptation setting.
- We show that the proposed source-based anchoring along with the source-guided contrastive alignment can be successfully utilised for robustly updating the model under dynamically changing test conditions.
- Extensive evaluation on challenging settings justifies its effectiveness for different scenarios.

We now discuss the related work specific to this chapter, followed by the proposed method and evaluation.

3.2 Related works

3.2.1 Knowledge and Self-distillation

Knowledge distillation is a technique used to transfer knowledge from a large, complex model (”teacher” model) to a smaller, simpler model (”student” model) [30, 31]. This is done by training the student model to mimic the predictions of the teacher model, which has already learned useful representations, rather than training the student model on the original labelled data. CoTTA also utilises distillation method to enhance the adaptation to new domains, which involves the implementation of a teacher model to make accurate predictions based on the student model.

A variant termed as self-distillation or self-knowledge distillation [32], involves training a model to mimic its own predictions. The framework in [33] shows that using a model from a previous epoch to train the same model in future epochs can increase the training efficiency and accuracy of the model.

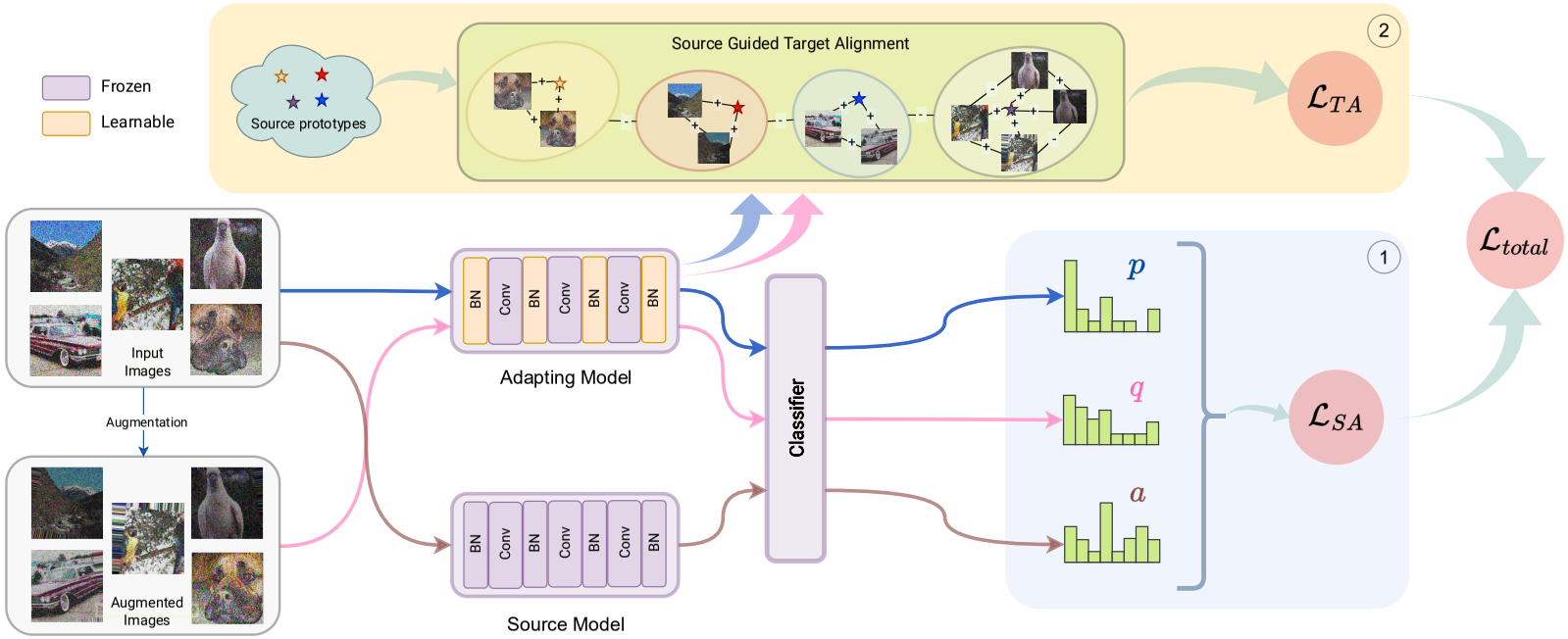


Figure 3.1: Illustration of the proposed SATA framework. The original image and its augmentation are passed through the adapting model (only BN affine parameters are updated) and the source model. ① The prediction of the input given by the source model is used as an anchor ($\mathbf{a}$) for the prediction given by the adapting model ($\mathbf{p}$ for input image and $\mathbf{q}$ for augmented image) for computing the source-anchoring loss $\mathcal{L}SA$. ② The source guided target alignment $\mathcal{L}TA$ uses the source prototypes to help maintain the semantic information learnt using the source domain and enforces clustering that is meaningful in the feature space. The augmented input is also used for both the losses.

3.2.2 Contrastive learning

Contrastive learning [34, 35, 36] has shown tremendous improvement in learning visual features for various downstream tasks. Because of its robustness researchers have used it in SFDA [37, 38, 39, 40, 41], TTA [42], Domain Generalisation (DG) [43, 44] and other studies [45, 46] to train/adapt a pre-trained model to a target domain using only unlabelled data. In previous works, contrastive learning has been used to get better feature representation. However, in this work, we align target features guided by source prototypes to get meaningful target features with respect to the source feature space. This results in meaningful clustering and good separation of classes in unseen domains.

3.3 Proposed SATA Framework

Given the model trained using the source data, the goal is to adapt it using the limited amount of test-data encountered in each batch in a dynamic environment, while satisfying the desirable criteria mentioned above. Towards this goal, we propose using (i) Source Anchorisation and (ii) Source-Guided Contrastive Alignment, which we now describe.

3.3.1 Source Anchorisation of Model

During test-time, the model has access to few test samples in a batch, which may not be representative of the corresponding target distribution. Thus, modifying the model parameters completely on the basis of the available target data may result in simultaneously overfitting on the few target samples and also catastrophic forgetting of the source information. CoTTA [9] addressed this challenge by (i) learning a teacher model by combining the source and a continuously adapting student model, wherein the teacher changes gradually for robust prediction and also using (ii) stochastic restoration to reset some of the model parameters to the source model after every batch. Though this gives impressive performance, it has two limitations, namely (i) the complete teacher,

student and source model needs to be stored and (ii) the hyperparameters required for computing the teacher model and also for stochastic restoration need to be determined, which can have different optimal values for different datasets. To overcome these challenges, in this work, we use self-distillation using the source model as the anchor [30, 33].

We denote the adapting model as f_θ, as this is the model which is constantly updated and is used to predict the target data. The weights θ of the adapting model is initialised to the weights θ_s of the off-the-shelf source model given by f_{θ_s}. Now, consider time instant k, when the model encounters a new batch denoted by B_k. Since adaptating the BN statistics has proven to be effective in capturing the data distribution characteristics [13, 9], the BN statistics of the source model (f_{θ_s}) and the adapting model (f_θ) are changed to the BN statistics of the target batch at each step. Let these models be denoted as $f^k_{\theta_s}$ and f^k_θ respectively. $f^k_{\theta_s}$ (referred to as source from now) can be thought of as a specialised model that accounts for the domain difference between the source and the specific target batch, On the other hand, the adapting model's weights are optimised after every batch using the loss function that will be described later. It should also be noted that during optimisation, only the BN parameters are updated for the adapting model [7].

The proposed self-distillation loss is inspired from the knowledge distillation loss formulation used in incremental learning [47, 48] to prevent catastrophic forgetting. In this work, self-distillation between the adapting model and the source model acts as a regulariser [49], which encourages the adapting model to mimic the source model, which is a specialised model whose response corresponds to domain invariant features. Thus, our adapting model is reinforced to learn domain invariant features [49], leading to better generalisation, which we empirically observe in Table 3.6.

The loss function is based on the prediction scores of the adapting model and source model for a given batch of test images, $B_k := \{x_1, x_2, \ldots, x_{N_k}\}$. Let p_{ij} and a_{ij} denote the j^{th} element of $f^k_\theta(x_i)$ (adapting model) and $f^k_{\theta_s}(x_i)$ (source model) respectively, which gives the prediction score of the j^{th} class for the i^{th} test image. The source-anchoring loss for a given batch is calculated as follows:

$$\mathcal{L}'_{\text{SA}}(B_k) = -\frac{1}{N_k}\sum_{i=1}^{N_k}\sum_{j=1}^{C} p_{ij}\log(a_{ij}) \tag{3.1}$$

Here, C is the number of classes, and N_k is the number of samples in the k^{th} batch. Empirically, we observe that using augmentations of the test images make the model more robust. Let q_{ij} denote the prediction score of the j^{th} class for the i^{th} augmented test image, given by the adapting model.

The complete source-anchoring loss is given by

$$\mathcal{L}_{\text{SA}}(B_k) = -\frac{1}{N_k}\sum_{i=1}^{N_k}\sum_{j=1}^{C}(p_{ij}\log(a_{ij}) + q_{ij}\log(a_{ij})) \tag{3.2}$$

This simple, yet effective self-distillation offers multiple advantages as follows: i) Since the modified source is used for anchoring, there is no hyper-parameter involved (like weights for combining student with source to form the teacher model); ii) The adapting model can be directly used for prediction continuously, without requiring any restoration to the source model; iii) Since only the BN parameters are updated, just these parameters of the source need to be stored, resulting in much lesser storage requirements compared to storing two different models.

3.3.2 Source-Guided Target Alignment

The goal is to learn a generalised model as it encounters data from different domains, thereby making the features gradually domain invariant. Here, we additionally use self-supervision (in the form of contrastive learning) for improved generalisation as used in [41, 39]. Formally, the adapting model f_θ^k can be decomposed into a feature extractor, g_ϕ^k and the fixed classifier h, i.e.

$$f_\theta^k = h \circ g_\phi^k \tag{3.3}$$

Suppose the augmented samples for the test batch data $\{x_i, y_i\}_{i=1}^{N_k}$ be denoted as $\{x_i, y_i\}_{i=N_k+1}^{2N_k}$ where $y_i = y_{N_k+i}$. We use the same augmentations for our experiments as in CoTTA [9]. These features are then passed to a projection head p_ψ so that the features are mapped to a d-dimensional hyper-sphere [34, 35].

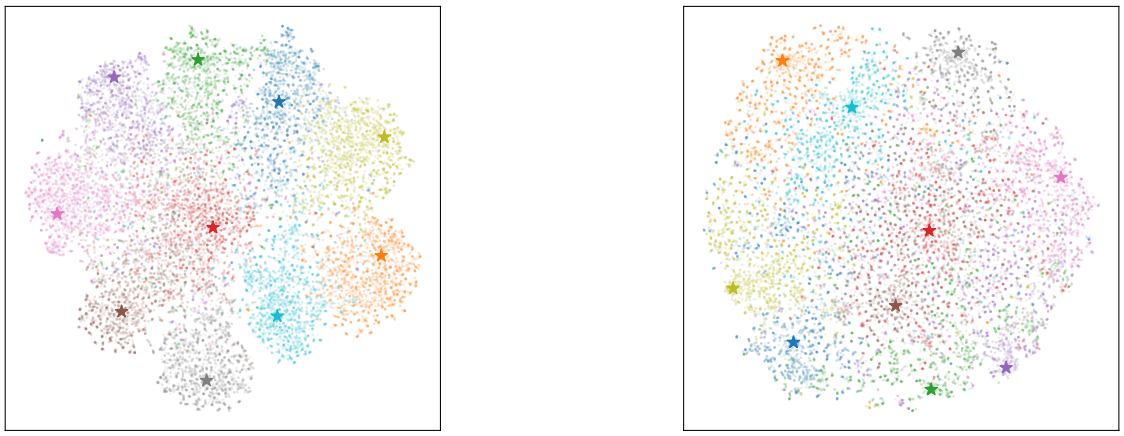
$$z_i = p_\psi \circ g_\phi^k(x_i) \tag{3.4}$$

Now, the parameters ψ and ϕ are optimised using the contrastive loss given below:

$$\mathcal{L}_{con} = \sum_{i=1}^{2N_k}\frac{-1}{|S_{-i}|}\sum_{j\in S_{-i}}\log\left(\frac{\exp(z_i.z_j/\tau)}{\sum_{k\neq i}\exp(z_i.z_k/\tau)}\right) \tag{3.5}$$

here $\tau > 0$ is the temperature hyperparameter and

$$S_{-i} = \{j \,|\, j \neq i, y_i = y_j \forall j \in \{1, \ldots, 2N_k\}\} \tag{3.6}$$



(a) (b)

Figure 3.2: t-SNE plot of the feature space using only $\mathcal{L}_{\text{TA}}$ for adaptation. We observe that the alignment and clustering is better with source prototypes (a) as compared to not using them (b). The source prototypes are represented using stars.

In this work, as in [7], only the BN layers of the feature extractor are modified to account for the changing distribution, but the classifier layer remains unchanged (to avoid overfitting on the few target samples in each batch). This ensures that the semantic information in the feature embedding space is not disturbed during the adaptation process. Thus, for correct classification, the target clusters should also align with the original source representations, which is achieved using the source-guided alignment loss. To this end, we include a third view which assigns the nearest source prototype features as an augmented view for the test time features. Specifically, let the source prototypes for the C classes be denoted as $\{\pi_i\}_{i=1}^{C}$. The source prototype views given by $\{x_i, y_i\}_{i=2N_k+1}^{3N_k}$, such that $y_i = y_{2N_k+i}$ are calculated as follows:

$$g_\phi^k(x_{2N_k+i}) = \{\pi_j | \arg\max_j (CosineSim(\pi_j, g_\phi^k(x_i)))\} \tag{3.7}$$

The effect of using prototypes as a view can be seen in Figure 3.2. We see that using this view (Eq 7 directly gives features) implicitly passes class information to the contrastive learning algorithm resulting in improved clustering. Therefore, the source-guided target alignment loss is given by:

$$\mathcal{L}_{\text{TA}} = \sum_{i=1}^{3N_k} \frac{-1}{|S_{-i}|} \sum_{j \in S_{-i}} \log \left(\frac{\exp(z_i.z_j/\tau)}{\sum_{k \neq i} \exp(z_i.z_k/\tau)} \right) \tag{3.8}$$

3.3.3 Final loss

Given any off-the-shelf pre-trained model and the source prototypes, during testing, using the current test batch, we modify the BN parameters such that the following loss is minimised:

$$\mathcal{L}_{\text{SATA}} = \mathcal{L}_{\text{SA}} + \mathcal{L}_{\text{TA}} \tag{3.9}$$

The adapting model is used for predicting the class of all the test samples. It is robust enough to be updated continuously without any restoration back to the source model.

3.4 Experimental Evaluation

Here we describe the extensive experiments performed to evaluate the effectiveness of the proposed framework.

Dataset Details: Here, we evaluate the proposed framework extensively on multiple benchmark datasets, namely, CIFAR-10C, CIFAR-100C and ImageNet-C [23]. These datasets have 10, 100 and 1000 classes respectively. All the datasets contain 15 diverse forms of corruption (noise, blur, weather, and digital) with five levels of severity, applied to the test set of all the three datasets. For all the experiments, unless mentioned otherwise, the test sequence consists of all 15 corruptions at the highest level of severity [9]. The goal is to adapt an off-the-shelf source model to this dynamically changing environment efficiently during test time.

Research Questions: The research questions that we want to answer using the experiments are the following:

i) How does the proposed framework perform on these datasets using the standard experimental protocol (higher batch size) as used in [9]?

ii) How does the model perform with lower batch sizes, which are more realistic in an online setting as in [7]?

iii) Are we able to learn a generalised model which also retains its good performance on the data

Dataset	Method	*gaussian*	*shot*	*impulse*	*defocus*	*glass*	*motion*	*zoom*	*snow*	*frost*	*fog*	*brightness*	*contrast*	*elastic*	*pixelate*	*jpeg*	Mean
CIFAR-10C	Source	72.3	65.7	72.9	46.9	54.3	34.8	42	25.1	41.3	26	9.3	46.7	26.6	58.5	30.3	43.5
	BN Stats Adapt [13]	28.1	26.1	36.3	12.8	35.3	14.2	12.1	17.3	17.4	15.3	8.4	12.6	23.8	19.7	27.3	20.4
	TENT-continual [7]	24.8	20.6	28.6	14.4	31.1	16.5	14.1	19.1	18.6	18.6	12.2	20.3	25.7	20.8	24.9	20.7
	CoTTA [9]	24.3	21.3	26.6	11.6	27.6	12.2	10.3	14.8	14.1	12.4	7.5	10.6	18.3	13.4	17.3	16.2
	SATA	23.9	20.1	28.0	11.6	27.4	12.6	10.2	14.1	13.2	12.2	7.4	10.3	19.1	13.3	18.5	**16.1** $\pm$ 0.06
CIFAR-100C	Source	73	68	39.4	29.3	54.1	30.8	28.8	39.5	45.8	50.3	29.5	55.1	37.2	74.7	41.2	46.4
	BN Stats Adapt	42.1	40.7	42.7	27.6	41.9	29.7	27.9	34.9	35	41.5	26.5	30.3	35.7	32.9	41.2	35.4
	TENT-continual	37.2	35.8	41.7	37.9	51.2	48.3	48.5	58.4	63.7	71.1	70.4	82.3	88	88.5	90.4	60.9
	CoTTA	40.1	37.7	39.7	26.9	38	27.9	26.4	32.8	31.8	40.3	24.7	26.9	32.5	28.3	33.5	32.5
	SATA	36.5	33.1	35.1	25.9	34.9	27.7	25.4	29.5	29.9	33.1	23.6	26.7	31.9	27.5	35.2	**30.3** $\pm$ 0.05
ImageNet-C	Source	97.8	97.1	98.2	81.7	89.8	85.2	78	83.5	77.1	75.9	41.3	94.5	82.5	79.3	68.6	82
	BN Stats Adapt	85	83.7	85	84.7	84.3	73.7	61.2	66	68.2	52.1	34.9	82.7	55.9	51.3	59.8	68.6
	TENT-continual	81.6	74.6	72.7	77.6	73.8	65.5	55.3	61.6	63	51.7	38.2	72.1	50.8	47.4	53.3	62.6
	CoTTA	84.7	82.1	80.6	81.3	79	68.6	57.5	60.3	60.5	48.3	36.6	66.1	47.2	41.2	46	62.7
	SATA	74.1	72.9	71.6	75.7	74.1	64.2	55.5	55.6	62.9	46.6	36.1	69.9	50.6	44.3	48.5	**60.1** $\pm$ 0.06

Table 3.1: Error percentages (lower is better) of different algorithms for CIFAR-10C, CIFAR-100C and ImageNet-C for batchsizes of 200, 200 and 64 respectively. For SATA the standard deviation is reported over 5 random seeds.

from source distribution?

iv) Are both the proposed modules important?

iv) Practical considerations - How does the model fare in terms of storage cost, inference time, number of hyperparameters to be tuned across datasets?

Implementation Details: For CIFAR-10C, we use a pre-trained WideResNet-28 [27] model from the RobustBench benchmark [50] as in [9]. The model is updated with one gradient step per iteration, and the Adam optimiser with a learning rate of 1e-3 is used. The temperature is set to the default value of 0.1. The CIFAR-100C experiment uses a pre-trained ResNeXt-29 [28] model, which is one of the default models for CIFAR-100 in the RobustBench benchmark [50]. The same hyperparameters as the CIFAR-10 experiment are used. For the ImageNet-C experiment, the standard pre-trained Resnet50 [51] model from RobustBench [50] is used. Here, SGD is used as the optimiser with a learning rate of 1e-2 as in [9]. We conduct all the experiments on an NVIDIA GeForce RTX 3090.

3.4.1 Evaluation on Standard Benchmarks

Table 3.1 reports the results on the three benckmark datasets. The batch sizes used for these experiments are 200, 200 and 64 for CIFAR-10C, CIFAR-100C and ImageNet-C[23] respectively as in [9]. All the results for the other approaches are directly taken from [9]. In the TENT-continual setup, the model continuously adapts and is not reset to the source model after each corruption. We

Dataset	Method	*gaussian*	*shot*	*impulse*	*defocus*	*glass*	*motion*	*zoom*	*snow*	*frost*	*fog*	*brightness*	*contrast*	*elastic*	*pixelate*	*jpeg*	Mean
CIFAR-10C	Source	72.3	65.7	72.9	46.9	54.3	34.8	42	25.1	41.3	26	9.3	46.7	26.6	58.5	30.3	43.5
	BN Stats Adapt	32.8	30.5	40.8	17.6	39.4	18.3	17.6	22.1	21.8	20.0	12.7	17.0	28.6	25.2	31.6	25.1
	TENT-continual	66.3	66.5	68.9	62.7	69.5	63.5	62.0	63.1	64.5	63.3	60.8	66.0	66.0	64.5	65.8	64.9
	CoTTA	55.9	55.7	56.9	52.6	58.2	53.4	51.2	55.4	54.7	53.7	51.3	55.9	55.4	54.0	53.9	54.5
	SATA	27.6	25.0	33.6	17.1	34.7	18.0	16.4	20.0	19.2	17.1	12.5	16.4	25.1	21.0	26.6	**22.0**
CIFAR-100C	Source	73	68	39.4	29.3	54.1	30.8	28.8	39.5	45.8	50.3	29.5	55.1	37.2	74.7	41.2	46.4
	BN Stats Adapt	48.7	47.4	49.3	34.5	48.7	36.1	34.7	41.8	41.3	47.7	33.3	37.1	43.0	40.2	47.7	42.1
	TENT-continual	96.7	96.9	96.7	96.7	96.8	96.8	96.7	96.5	96.9	96.9	96.7	96.9	96.8	96.6	96.7	96.8
	CoTTA	66.5	65.6	67.2	62.5	66.5	63.1	62.0	66.1	64.4	68.8	61.8	64.9	65.4	62.5	65.7	64.9
	SATA	44.2	41.8	42.1	34.5	45.1	35.7	33.9	38.7	39.4	42.9	31.7	35.8	40.7	36.3	44.1	**39.1**
ImageNet-C	Source	97.8	97.1	98.2	81.7	89.8	85.2	78	83.5	77.1	75.9	41.3	94.5	82.5	79.3	68.6	82
	BN Stats Adapt	88.6	87.7	87.9	89.6	89.0	79.5	70.1	71.8	74.4	60.9	43.0	86.5	64.4	59.9	67.9	74.8
	TENT-continual	80.7	78.2	85.1	96.9	99.1	99.5	99.4	99.6	99.6	99.5	99.5	99.7	99.6	99.5	99.6	97.5
	CoTTA	86.5	87.2	91.9	96.5	97.5	98.2	98.9	99.1	99.4	99.5	99.5	99.7	99.5	99.6	99.6	96.8
	SATA	95.0	95.0	94.4	94.3	92.0	86.8	77.7	76.2	78.3	63.4	52.2	83.6	66.3	58.8	63.9	**78.5**

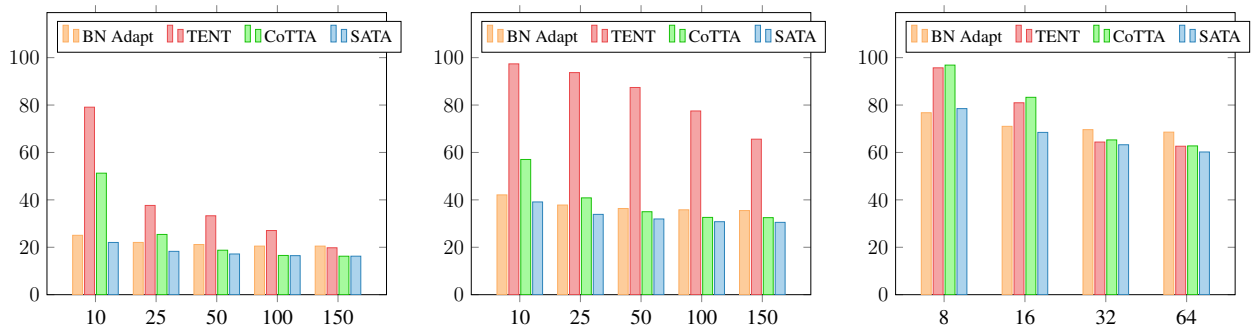
Table 3.2: Error percentages (lower is better) of different algorithms for CIFAR-10C, CIFAR-100C and ImageNet-C for batchsizes of 10, 10 and 8 respectively.

observe that for all the datasets, the proposed SATA outperforms all the other existing approaches. Specifically, for the challenging ImageNet-C, we obtain an error of 60.1%, which is 2.6% better than the previous state-of-the-art CoTTA. In addition to the gain in performance, SATA has other advantages, as elaborated on later.

3.4.2 Evaluation with Lower Batch Sizes

A practical test-time adaptation algorithm should work satisfactorily for lower batch sizes, which will decrease the average time for inference of a sample and thus lower the latency of the framework. Recently, researchers have started to address the issue of robustness across batch sizes [52], but many of these methods are not fully test time adaptation (FTTA) and requires source training for initialisation.

Here, we evaluate the robustness of the proposed SATA framework for lower batch sizes and compare the results with the state-of-the-art. Figure 3.3 shows the results for the three datasets with decreasing batch sizes. Since the results of the other approaches were not reported for other batch sizes, we ran the official codes and obtained the results reported in the table. To ensure the best results for the other methods, we tuned the appropriate hyperparameters. Specifically, for TENT, we varied the learning rate. For CoTTA, the restoration probability and model EMA factor was reduced in proportion to the decrease in batch size. Changing any other parameters



(a) CIFAR-10C (b) CIFAR-100C (c) ImageNet-C

Figure 3.3: These figures compare the robustness of the different approaches for different batch sizes. The x-axis is the batch size and y-axis is the error percentage (lower is better). We observe that the proposed SATA is robust across batch sizes.

Methods	CIFAR-10C						CIFAR-100C						IMAGENET-C			
	200	150	100	50	25	10	200	150	100	50	25	10	64	32	16	8
SALoss (original image)	20.1	20.1	20.4	20.7	21.7	24.8	32.6	32.8	33.2	34.0	35.7	40.7	62.7	64.9	69.4	77.8
SALoss (original + augmented image)	17.1	17.2	17.4	18.0	19.0	22.5	31.4	31.6	32.0	32.8	34.7	39.9	61.3	63.8	68.5	**76.8**
SATA (SALoss + TALoss)	**16.1**	**16.2**	**16.5**	**17.2**	**18.3**	**22.0**	**30.4**	**30.5**	**30.8**	**31.9**	**33.9**	**39.1**	**60.2**	**63.2**	**68.5**	78.5

Table 3.3: Mean error percentage (lower is better) demonstrating the importance of the two proposed components. Ablation is done on all the four datasets for multiple batch sizes.

did not substantially change the performance of the model. Note that for the proposed SATA, *no parameters were changed across the different datasets as well as batch sizes*. We observe that the improvement provided by our approach becomes clearer as the batch size decreases. For example, for a batch size of 10, we obtain 22.0% for CIFAR-10C dataset, which is significantly better compared to the next best obtained by BNStats. Though there is still a lot of room for improvement for all the frameworks, this experiment justifies the effectiveness of the proposed SATA for online setting.

3.4.3 Ablation Study

Here, we analyze the importance of the two losses in the proposed SATA framework. We observe from Table 3.3 that most of the performance improvement of SATA can be attributed to the source-anchoring of the test samples and its augmented version. The clustering and alignment terms further help to improve the performance, thereby achieving state-of-the-art performance for continual test-time domain adaptation under different challenging settings.

Avg. Error (%)	Source	BN Adapt	TENT (cont)	CoTTA	SATA
CIFAR-100C	33.6	29.9	74.8	26.3	**25.6**

Table 3.4: Average error over all corruptions with severity presented in the gradual test time adaptation manner. The order of corruption used here is the same as in Table 3.1.

batchsize →	200	150	100	50	25	10
TENT	92.0 (70.9)	96.2 (75.1)	97.8 (76.7)	98.2 (77.1)	98.5 (77.4)	99.0 (77.9)
CoTTA	22.8 (1.7)	23.0 (1.9)	23.9 (2.8)	27.2 (6.1)	35.5 (14.4)	58.5 (37.4)
SATA	**22.4 (1.3)**	**22.7 (1.6)**	**22.9 (1.8)**	**23.6 (2.5)**	**25.8 (4.7)**	**29.9 (8.8)**

Table 3.5: Error on CIFAR-100 test set after the model has been adapted to all the corruptions as in Table 3.1. $(.)$ is the degradation in performance on source data compared to the source model, whose error is 21.1%. This deviation can be thought of as a proxy for catastrophic forgetting.

3.5 Further Analysis

Here, we perform further analysis to evaluate the usefulness of the proposed framework and its different components. All these analysis are done on the CIFAR-100C datasets, unless stated otherwise.

Performance on gradually changing data: In the standard setup, the corruption types change abruptly with maximum severity levels. A more realistic approach will be to evaluate the performance of the approaches when the severity levels change gradually over a sequence of 15 corruption types. Thus we experiment with the gradual setup as also done in [9]. The representation below shows the order in which severity is faced for every corruption.

$$\underbrace{\ldots \rightarrow 2 \rightarrow 1}_{\text{t-1 and before}} \rightarrow \underbrace{1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 1}_{\text{t corruption type, with changing severity}} \rightarrow \underbrace{1 \rightarrow 2 \rightarrow \ldots}_{\text{t+1 and after}}$$

Table 3.4 reports the results of the proposed framework and comparisons with the existing approaches for this setup. The results suggest that BN Adapt, CoTTA, and SATA are more effective than the source and TENT (cont) approaches in dealing with corruption types that change gradually over time. In particular, our approach achieves the lowest average error rate of 25.6%, followed by the CoTTA approach with an average error rate of 26.3%. The BN Adapt approach also performs well with an average error rate of 29.9%.

Performance on source data: As the trained model gradually adapts to the changing testing conditions, we want it to be able to perform well on the original source distribution, which requires that the model has not catastrophically forgotten the original training information. For e.g., the original model trained on clear weather conditions should continue to do well for clear weather images, even though it has adapted to other conditions like rainy, foggy, etc.

To achieve these contrasting goals, it is important that the model has the right balance of stability-plasticity. The stability-plasticity trade-off refers to the balance between preserving learned knowledge and adapting to new information. Table 3.5 reports the results of using the adapted model on a held-out testing set from the source distribution on CIFAR-100C dataset. The performance of the original model trained on source gives an error of 21.1% on its test set. Thus the difference gives an estimate of the catastrophic forgetting (Table 3.5). We observe that even after adaptation, SATA is able to maintain its performance on the source distribution very well as compared to CoTTA and TENT for all batch sizes.

Generalisability of the learnt model: When the source model encounters data from different domains, we want it to gradually become more generalised, such that the features become domain invariant. To evaluate whether this happens in practice, we perform an experiment, in which the model is first adapted on the first 7 corruptions (gaussian to zoom). We then freeze the weights and evaluate its performance on the last 8 corruptions (snow to jpeg). We compare the performance of this adapted model (adapted using CoTTA and SATA) to the performance using the source model, whose BN statistics are adapted to the corresponding batch statistics. We see from the results in Table 3.6 that both CoTTA and the proposed framework have indeed learnt more generalised features and is therefore performing better than the source model (BN Adapt). To understand the generalisation capability provided by our individual losses, we also report the results using only the source-anchoring loss ($\mathcal{L}_{\text{SA}}$). We observe that this single loss term gives comparable performance as CoTTA, even without the target alignment loss.

Effect of source prototypes: In some cases, prototypes may not be available. We check the performance of our method in such cases as demonstrated in Table 3.7. We observe that without the prototypes, the performance drops slightly compared to our complete loss. This drop is small because the other loss component ($\mathcal{L}_{\text{SA}}$) will increase if the features of the adapting model are not aligned with the features of the source. Thus, even if only the pre-trained model is available, our method can be used for test time adaptation.

Method	*snow*	*frost*	*fog*	*brightness*	*contrast*	*elastic*	*pixelate*	*jpeg*	Mean
BN Adapt	35.6	34.9	42.1	26.9	31.0	36.0	33.5	41.7	35.2
CoTTA	31.3	30.5	36.7	25.2	27.8	31.5	29.1	36.4	**31.1**
SALoss	31.9	31.3	37.2	25.0	29.4	33.0	29.2	37.8	31.9
SATA	31.2	31.2	36.3	25.0	28.7	32.2	28.0	37.2	31.2

Table 3.6: This experiment on CIFAR-100C demonstrates the generalisability of the learnt model. We observe that both CoTTA and SATA yield a more generalised model after adaptation.

	CIFAR-10C	CIFAR-100C	ImageNet-C
SATA w/ prototypes	16.1	30.0	60.2
SATA w/o prototypes	16.2 (-0.1)	31.4 (-1.4)	61.1 (-0.9)

Table 3.7: Here, we see the performance comparison (error % - lower is better) between using prototypes as a view v/s not using them in the target alignment loss. (.) is the drop in performance compared to the complete loss.

Computational Advantages: The proposed SATA framework has additional advantages over the existing approaches as follows:

1) Less memory requirements due to less number of trainable parameters;

2) Faster inference time due to lesser number of forward passes.

Number of parameters to be stored: Table 3.8 reports the total number of parameters and trainable parameters for TENT, CoTTA and SATA. We observe that CoTTA has 33% more parameters than SATA and the trainable parameters in SATA is only 2.1% of those in CoTTA. Thus the proposed framework is much simpler and computationally efficient, which can be especially important in real-time applications where time and computational resources are limited.

Method	# Parameters	# Trainable	% Trainable
BN-Stats	6,900,132	0	0
TENT	6,900,132	128	0.002
CoTTA	20,700,396	6,900,132	33.333
SATA	13,947,976	147,840	1.060

Table 3.8: Number of (trainable) parameters as a proxy for the storage requirement of the respective algorithms. This table is for CIFAR-100C with ResNeXt-29 as the backbone.

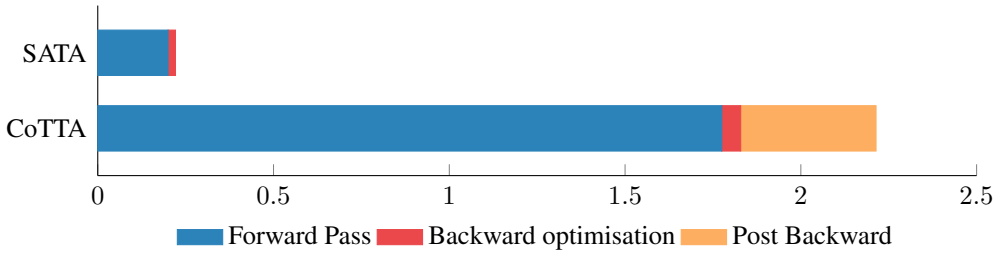


Figure 3.4: Comparison of inference time of the proposed SATA framework with the state-of-the-art CoTTA. The x-axis is time of inference per batch (sec/batch). These experiments were done on CIFAR-100C using NVIDIA GeForce RTX 3090.

Inference time for a fixed batch size: The inference time of a model is an important factor in test-time adaptation because it determines how quickly the model can be applied to new data. In Figure 3.4 we have compared the inference time (in sec) per batch (of 200 samples) for CoTTA and SATA. We see SATA is around 10 times faster than CoTTA during inference. This can be attributed to the fact that CoTTA uses 32 forward passes in the worst case for its prediction and also updates the teacher model after every step.

3.6 Conclusion

In this chapter, we proposed a novel SATA framework for the challenging task of continual test-time domain adaptation. The proposed approach modifies the batch-norm affine parameters using source anchoring-based self-distillation to ensure the model incorporates knowledge of newly encountered domains while avoiding catastrophic forgetting. Additionally, source prototype guided target alignment is proposed to maintain the already learned semantic information while grouping target samples naturally. The approach is quite robust to decreasing batch sizes, justifying its effectiveness for online application. But we observe that for very small batch sizes (e.g. 8 in ImageNet), its performance drops slightly below BN Adapt, though even for this case, it is significantly better than TENT and CoTTA. The SATA framework offers additional advantages like retaining performance on the source domain, and having minimal tunable hyper-parameters and storage requirements, in addition to achieving state-of-the-art results on all the benchmark datasets.

Chapter 4

Conclusion

In conclusion, deep learning has made tremendous strides in computer vision tasks such as classification, object detection, and segmentation. However, deep neural networks have shown poor performance when tested on data from a different distribution than the training data. To address this challenge, unsupervised domain adaptation techniques have been developed, but they require access to labelled source data and unlabelled target data. Moreover, the test data distribution may dynamically vary with time, which is critical to continually adapt the model during test time to ensure optimal performance in changing scenarios.

Continual adaptation during test time is particularly important in real-world scenarios where the data distribution can be dynamic and challenging to predict. Hence, it is a critical research area in deep learning. The ongoing research is focused on developing models and algorithms that can learn to adapt and generalise to different environments, aiming to create more robust and reliable deep learning models. The thesis contributes to this field by proposing novel ideas and frameworks to address the challenges of adapting models to changing test environments. The results are promising and open up avenues for further research and development in this area.

Bibliography

[1] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[2] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *NeurIPS*, 2015.

[3] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017.

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2018.

[5] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010.

[6] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *NeurIPS*, 32, 2019.

[7] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *ICLR*, 2021.

[8] Malik Boudiaf, Romain Mueller, Ismail Ben Ayed, and Luca Bertinetto. Parameter-free online test-time adaptation. In *CVPR*, pages 8344–8353, 2022.

[9] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *CVPR*, pages 7201–7211, 2022.

[10] Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, and Mario Marchand. Domain-adversarial neural networks. *arXiv preprint arXiv:1412.4446*, 2014.

[11] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain generalization with adversarial feature learning. In *CVPR*, 2018.

[12] Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting batch normalization for practical domain adaptation. *arXiv preprint arXiv:1603.04779*, 2016.

[13] Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. *NeurIPS*, 2020.

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

[15] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. *Advances in neural information processing systems*, 31, 2018.

[16] Yang Zou, Zhiding Yu, BVK Kumar, and Jinsong Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *Proceedings of the European conference on computer vision (ECCV)*, pages 289–305, 2018.

[17] Wilhelm Tranheden, Viktor Olsson, Juliano Pinto, and Lennart Svensson. Dacs: Domain adaptation via cross-domain mixed sampling. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1379–1389, 2021.

[18] Maohao Shen, Yuheng Bu, and Gregory Wornell. On the benefits of selectivity in pseudo-labeling for unsupervised multi-source-free domain adaptation. *arXiv preprint arXiv:2202.00796*, 2022.

[19] Yufan He, Aaron Carass, Lianrui Zuo, Blake E Dewey, and Jerry L Prince. Autoencoder based self-supervised test-time adaptation for medical image analysis. *Medical image analysis*, 72:102136, 2021.

[20] Y. Sun, X. Wang, Z. Liu, J. Miller, A. A. Efros, and M. Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *ICML*, 2020.

[21] Yulu Gan, Xianzheng Ma, Yihang Lou, Yan Bai, Renrui Zhang, Nian Shi, and Lin Luo. Decorate the newcomers: Visual domain prompt for continual test time adaptation. *arXiv preprint arXiv:2212.04145*, 2022.

[22] Taesik Gong, Jongheon Jeong, Taewon Kim, Yewon Kim, Jinwoo Shin, and Sung-Ju Lee. NOTE: Robust continual test-time adaptation against temporal correlation. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *NeurIPS*, 2022.

[23] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

[24] Dian Chen, Dequan Wang, Trevor Darrell, and Sayna Ebrahimi. Contrastive test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 295–305, 2022.

[25] Shiqi Yang, Yaxing Wang, Kai Wang, SHANGLING JUI, and Joost van de weijer. Attracting and dispersing: A simple approach for source-free domain adaptation. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[26] Yabin Wang, Zhiwu Huang, and Xiaopeng Hong. S-prompts learning with pre-trained transformers: An occam's razor for domain incremental learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 5682–5695. Curran Associates, Inc., 2022.

[27] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, pages 87.1–87.12, 2016.

[28] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, pages 1492–1500, 2017.

[29] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*, 2020.

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

[31] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *IJCV*, 129:1789–1819, 2021.

[32] Li Yuan, Francis EH Tay, Guilin Li, Tao Wang, and Jiashi Feng. Revisit knowledge distillation: a teacher-free framework. 2019.

[33] Chenglin Yang, Lingxi Xie, Chi Su, and Alan L Yuille. Snapshot distillation: Teacher-student optimization in one generation. In *CVPR*, pages 2859–2868, 2019.

[34] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. pages 1597–1607. PMLR, 2020.

[35] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *NeurIPS*, 33:18661–18673, 2020.

[36] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *NeurIPS*, 33:9912–9924, 2020.

[37] Jiaxing Huang, Dayan Guan, Aoran Xiao, and Shijian Lu. Model adaptation: Historical contrastive learning for unsupervised domain adaptation without source data. *NeurIPS*, 34:3635–3649, 2021.

[38] Haifeng Xia, Handong Zhao, and Zhengming Ding. Adaptive adversarial network for source-free domain adaptation. In *ICCV*, pages 9010–9019, 2021.

[39] Rui Wang, Zuxuan Wu, Zejia Weng, Jingjing Chen, Guo-Jun Qi, and Yu-Gang Jiang. Cross-domain contrastive learning for unsupervised domain adaptation. *IEEE Transactions on Multimedia*, 2022.

[40] Shiqi Yang, Yaxing Wang, Kai Wang, Shangling Jui, et al. Attracting and dispersing: A simple approach for source-free domain adaptation. In *NeurIPS*, 2022.

[41] Xuejun Zhao, Rafal Stanislawski, Paolo Gardoni, Maciej Sulowicz, Adam Glowacz, Grzegorz Krolczyk, and Zhixiong Li. Adaptive contrastive learning with label consistency for source data free unsupervised domain adaptation. *Sensors*, 22(11):4238, 2022.

[42] Dian Chen, Dequan Wang, Trevor Darrell, and Sayna Ebrahimi. Contrastive test-time adaptation. In *CVPR*, pages 295–305, 2022.

[43] Daehee Kim, Youngjun Yoo, Seunghyun Park, Jinkyu Kim, and Jaekoo Lee. Selfreg: Self-supervised contrastive regularization for domain generalization. In *ICCV*, pages 9619–9628, 2021.

[44] Xufeng Yao, Yang Bai, Xinyun Zhang, Yuechen Zhang, Qi Sun, Ran Chen, Ruiyu Li, and Bei Yu. Pcl: Proxy-based contrastive learning for domain generalization. In *CVPR*, pages 7097–7107, 2022.

[45] Sagar Vaze, Kai Han, Andrea Vedaldi, and Andrew Zisserman. Generalized category discovery. In *CVPR*, pages 7492–7501, 2022.

[46] Yixin Fei, Zhongkai Zhao, Siwei Yang, and Bingchen Zhao. Xcon: Learning with experts for fine-grained category discovery. *arXiv preprint arXiv:2208.01898*, 2022.

[47] Songlin Dong, Xiaopeng Hong, Xiaoyu Tao, Xinyuan Chang, Xing Wei, and Yihong Gong. Few-shot class-incremental learning via relation knowledge distillation. In *AAAI*, volume 35, pages 1255–1263, 2021.

[48] Minsoo Kang, Jaeyoo Park, and Bohyung Han. Class-incremental learning by knowledge distillation with adaptive feature consolidation. In *CVPR*, pages 16071–16080, 2022.

[49] Hossein Mobahi, Mehrdad Farajtabar, and Peter Bartlett. Self-distillation amplifies regularization in hilbert space. *NeurIPS*, 33:3351–3361, 2020.

[50] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. In *NeurIPS*, 2021.

[51] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[52] Hyesu Lim, Byeonggeun Kim, Jaegul Choo, and Sungha Choi. TTN: A domain-shift aware batch normalization in test-time adaptation. In *ICLR*, 2023.