A Simple Signal for Domain Shift

Goirik Chakrabarty
IISER, Pune
goirik.chakrabarty@students.iiserpune.ac.in

Manogna Sreenivas Soma Biswas
IISc, Bangalore
{manognas, somabiswas}@iisc.ac.in

Abstract

Test time domain adaptation has come to the forefront as a challenging scenario in recent times. Although single domain test-time adaptation has been well studied and shown impressive performance, this can be limiting when the model is deployed in a dynamic test environment. We explore this continual domain test time adaptation problem here. Specifically, we question if we can translate the effectiveness of single domain adaptation methods to continuous test-time adaptation scenario. We take a step towards bridging the gap between these two settings by proposing a domain shift detection mechanism and hence allowing us to employ the current test-time adaptation methods even in a continual setting. We propose to use the given source domain trained model to continually measure the similarity between the feature representations of the consecutive batches. A domain shift is detected when this measure crosses a certain threshold, which we use as a trigger to reset the model back to source and continue test-time adaptation. We demonstrate the effectiveness of our method by performing experiments across datasets, batch sizes and different single domain test-time adaptation baselines.

1. Introduction

The ability to continually adapt models in real-time is becoming increasingly important in today's fast-paced technological landscape. Research in deep learning [7, 17, 9, 5, 12, 22] 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.

Test-time adaptation (TTA) is a critical area of research aimed at addressing distribution shifts between the training and testing data. Current TTA methods[25, 30, 21] primarily address the single domain shift scenario. A more recent and practical scenario is that of Continual Test Time Adaptation(CTTA). This problem is particularly important for deep models operating in environments where the test conditions frequently change. TTA methods aim to adapt a given source model with respect to any testing sample from an unseen target domain using methods like entropy minimization[25]. While effective in stationary test environments, these[25, 30, 21] can become unstable in CTTA. This instability arises from two factors: (1) *Error accumulation:* In continually changing environments, the distribution shift makes the pseudo-labels less reliable, resulting in early prediction mistakes that accumulate errors over time. (2) *Catastrophic forgetting:* As the model continually adapts to new distributions, it struggles to retain knowledge from the source domain, leading to a phenomenon known as catastrophic forgetting.

Contrary to TTA, in CTTA, researchers aim to create algorithms that enable the model to incrementally adapt to new test domains while retaining its ability to perform well in the source domain and previously encountered test domains. This is a more complex and dynamic task that requires specialized techniques to handle the continuous evolution of test conditions and data distributions. However, our approach successfully bypasses the complexity of CTTA methods and successfully employs the simplicity of TTA methods in CTTA setting.
We summarize our contributions below:

- We propose a Simple Signal to Domain Shift, to employ efficient TTA methods in CTTA setting.
- We empirically demonstrate that the features obtained from the source model can be used to uniquely identify domains in feature space.
- We perform extensive experiments on multiple datasets and incorporate our module with multiple TTA methods to demonstrate its effectiveness in the CTTA setting.

2. Related Work

The study of robustness of deep networks against distribution shifts has rapidly evolved in recent years, broadly covering the following topics:

Unsupervised Domain Adaptation (UDA): This setting assumes access to labeled source domain data along with unlabeled target domain data during training. UDA methods [11, 19, 23, 20, 28, 24] primarily aim to align the two domains so that the supervision from source domain can be transferred to that of target.

Domain Generalization (DG): DG methods [15, 14, 32] use multiple source domains to learn robust domain-invariant representations so that the model can better generalize to unseen test domains.

UDA and DG focus on training robustness and the goal of learning domain-invariant representations. In contrast, SFDA and TTA aim to adapt off-the-shelf models using unlabeled target domain data. SFDA allows for the collection of ample target domain data for offline adaptation, while TTA is a more realistic setting where the model must be adapted using test data that is only available online and can only be seen once.

Source Free Domain Adaptation (SFDA): Contrary to UDA and DG, SFDA methods [16, 29, 30, 3] attempt to adapt a source model using abundant unlabeled target domain data. This data is assumed to be available offline and can be shown to the model multiple times for adaptation.

Test Time Adaptation (TTA): This setting was first proposed by [25] with the objective of leveraging the test data coming in an online manner to adapt a given off-the shelf model. The key challenges here are: (1) No access to labels and therefore inability recognise and correct wrong predictions; (2) No access to source data; (3) Viewing data in an online manner i.e. you have access to each test minibatch only once.

Continuous Test Time Adaptation (CTTA): Taking another step forward from TTA towards reality, a recent work [26] formalized the CTTA setting where the test domain can dynamically change in time. The seminal approach [26] reduces error accumulation through weight-averaged and augmentation-averaged predictions and avoids catastrophic forgetting through stochastic restoration of source pre-trained weights. However, this method is computationally very taxing as also acknowledged by the authors. Contrary to CoTTA [26], RMT [8] and SATA [2], we propose a simple strategy to detect domain shift, thus enabling us to employ light weight single domain TTA methods like TENT [25] in CTTA setting.

More recent works[18, 10] also explore the CTTA setting. As ViDA uses Vision Transformers instead of CNNs and VDP requires source data to initialize the prompts, we cannot directly compare with these methods.

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

Table 1: Domain adaptation protocols

3. Motivation

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 information 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.

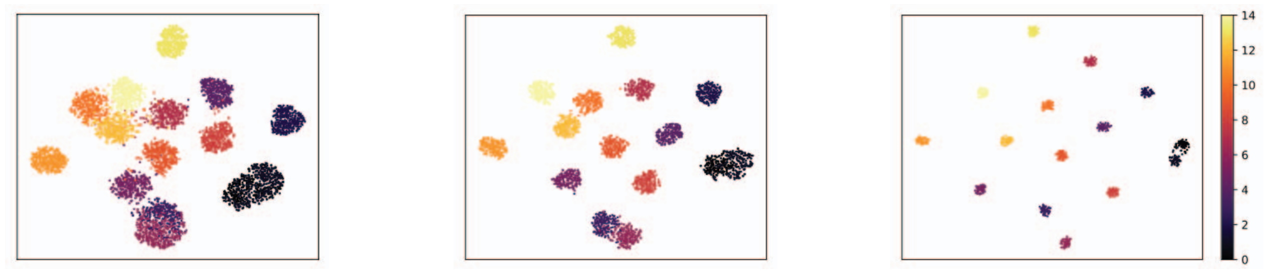
4. Problem setting

Given an off-the shelf model $h_{\boldsymbol{\theta}} = f \circ g$ 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 $\mathcal{L}_{test}(\mathbf{x}_t;\theta)$. In the single domain TTA addressed in [25, 1, 4], $\mathbf{x}_t$ 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 $\mathcal{D}_{t1}, \mathcal{D}_{t2}, \mathcal{D}_{t3}, ..., \mathcal{D}_{tN}$, where $\mathcal{D}_{ti} \neq \mathcal{D}_{train} \forall i$. We address this more challenging and realistic CTTA scenario leveraging the ideas of TTA through a Domain Shift Signal.

5. Method

We first briefly describe the single domain TTA method TENT [25] and SFDA method AaD [30]. Then, we describe how to employ these methods in the CTTA setting using the proposed Domain Shift Signal.

TENT: Tent is a seminal work, which first proposed the TTA setting to online adapt any given off-the-shelf model $h_{\boldsymbol{\theta}}$. In general, the Batch Normalization (BN) layers in a model estimate the feature statistics during training, which


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

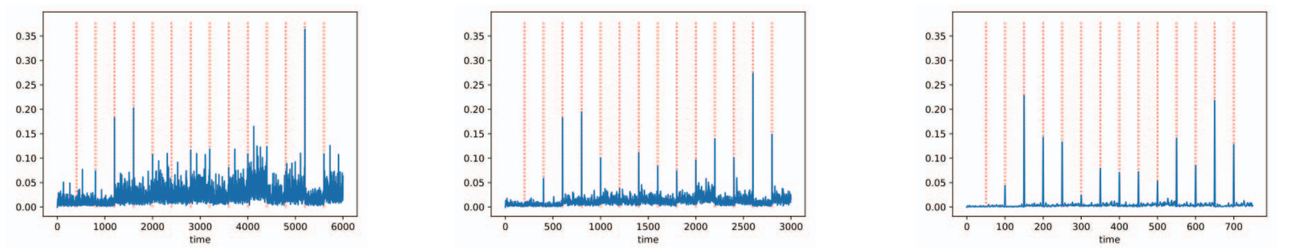

(d) Batch size=25 (e) Batch size=50 (f) Batch size=200

Figure 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 DSS signals to the t-SNE. The red dotted lines are where the actual domain shift happens.

is then used during testing. However, when $\mathcal{D}_{test} \neq \mathcal{D}_{train}$, the estimated statistics are no longer appropriate. Tent proposes to correct this by using the feature statistics of the test data instead. Further, they fine-tune the BN's affine parameters to minimize the Shannon entropy, as they observe that test entropy is correlated with the test error. For a test sample x_i, the optimization objective is

$$\mathcal{L}_{ent}(x_i) = -\sum_c p_c \log p_c \tag{1}$$

where p_c is the softmax score of class c for the sample x_i.

Attracting and Dispersing (AaD): AaD 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

$$\mathcal{L}(x_i) = -\sum_j p_i^T p_j + \lambda \sum_{m \in \mathbf{x}_t} p_i^T p_m \tag{2}$$

Here $\mathcal{N}_i$ is the set of neighbours of x_i and p_m refers to the softmax prediction vector of a sample $x_m \in \mathbf{x}_t$.

Efficient Anti-forgetting Test-time Adaptation (EATA): EATA proposes active sample selection, identifying reliable and non-redundant samples for model adaptation, minimizing entropy loss during TTA. Additionally, a Fisher regularizer is introduced to prevent drastic parameter changes and alleviate the forgetting problem by estimating Fisher importance from test samples with pseudo labels. This method improves out-of-distribution test performance while maintaining in-distribution data accuracy. The test objective for a sample x_i from a test batch $\mathbf{x}_t$ is

$$\mathcal{L}_{EATA}(x_i) = S(x_i)\mathcal{L}_{ent}(x_i) + \beta\mathcal{R}(\theta, \theta_0) \tag{3}$$

where $S(x_i)$ is the sample importance, $\mathcal{R}()$ is the Fisher regularization, θ and θ_0 are the current model parameters and source model parameters respectively.

While these methods achieve state-of-the-art performance in single domain adaptation setting, they suffer from error accumulation due to over-fitting in CTTA. We observe that source model is a more reliable starting point for adaptation on domain shift than a continually adapting model. 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.

5.1. 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 characterize 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 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 hypothesize 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 $\mathbf{E}v_f(t)$ 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 1, we visualize 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

$$\text{DSS} = 1 - \cos(\mathbf{E}v_f(t), \mathbf{E}v_f(t-1)) \tag{4}$$

This signal is depicted in the lower row of figure 1. We can clearly see that there are spikes which align well with the true domain shift shown by the dotted red lines. When $\mathbf{E}v_f(t)$ comes from the same domain as $\mathbf{E}v_f(t-1)$, DSS is low. Otherwise, we need to trigger a model reset back to the source model. In order to detect the domain shift, we propose to obtain a smooth estimate of the DSS by calculating its moving average. Further, we use this moving average to define a Domain Shift Detector(DSD) as follows:

$$\text{DSD} = \mathbb{I}_{\{\text{DSS} > k \cdot \text{MovingAverage(DSS)}\}} \tag{5}$$

The proposed module is completely decoupled from the underlying TTA method. Our objective is to employ any single domain TTA method[25, 30, 21] in a CTTA scenario through the means of the Domain Shift Detection module. By simply resetting the model to source on domain shift, we simulate a single domain TTA setting. Our method improves the performance of any TTA method in CTTA setting as it mitigates error accumulation. We summarize the proposed algorithm.

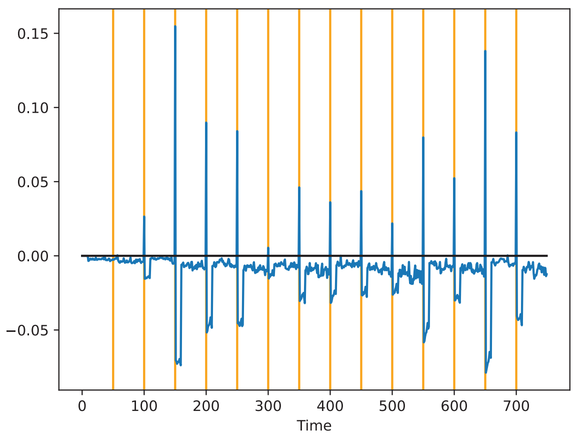


Figure 2: The blue line is the function DSS - $k\cdot$ MovingAverage(DSS). The orange lines represent the true domain shift. The zero crossing of the blue line is the indicator for domain shift as described in Equation 5.

Algorithm 1: Domain Shift Detection module

```
Input:
Source feature extractor $f$
Domain Shift Detection:
for each batch $\mathbf{x}_t$:
    $v_{f,i} = f(x_{t,i})$
    $\mathbf{E}v_f(t) = \frac{1}{N}\sum_{k=1}^{N} v_{f,i}$
    $\text{DSS}(\mathbf{E}v_f(t), \mathbf{E}v_f(t-1)) = 1 - \frac{\mathbf{E}v_f(t)^T \mathbf{E}v_f(t-1)}{||\mathbf{E}v_f(t)||||\mathbf{E}v_f(t-1)||}$
    $\text{DSD} = \mathbb{I}_{\{\text{DSS} > k \cdot \text{MovingAverage(DSS)}\}}$
    if DSD == 1:
        Reset model to source
    Continue TTA
```

6. Experiments and Results

We now describe the datasets, baseline TTA methods and implementation details of the experiments done to determine the effectiveness of our proposed method.

	Method	*gaussian*	*shot*	*impulse*	*defocus*	*glass*	*motion*	*zoom*	*snow*	*frost*	*fog*	*brightness*	*contrast*	*elastic*	*pixelate*	*jpeg*	Mean
ImageNet-C	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
	BN Stats	84.9	84.0	84.8	84.9	84.5	73.3	61.1	65.8	68.2	51.9	35.0	83.0	56.3	51.2	60.0	68.6
	CoTTA[26]	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
	RMT[8]	79.9	76.3	73.1	75.7	72.9	64.7	56.8	56.4	58.3	49.0	40.6	58.2	47.8	43.7	44.8	59.9
	SATA[2]	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
	TENT-TTA[25]	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 (+28.9)
	AaD-TTA[30]	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	84.6	83.8	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 (+21.6)
	EATA-TTA[21]	76.3	69.4	63.1	71.3	68.7	59.8	52.4	55.3	59.1	47.2	33.4	61.0	47.2	42.8	46.5	56.9
	EATA-CTTA	76.3	66.5	65.0	73.1	69.1	62.1	53.5	58.9	59.3	48.1	35.9	62.8	47.5	43.9	47.5	58.0
	EATA-DSS	76.3	66.4	63.1	71.3	68.7	59.8	52.4	55.3	59.1	47.2	33.4	61.0	47.2	42.8	46.5	**56.7** (+1.3)
CIFAR-100C	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
	BN Stats	42.3	40.7	43.2	27.7	41.8	29.8	27.9	35.0	34.7	41.8	26.4	30.2	35.6	33.1	41.2	35.4
	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
	RMT	40.2	36.2	36.0	27.9	33.9	28.4	26.4	28.7	28.8	31.1	25.5	27.1	28.0	26.6	29.0	**30.2**
	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
	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	37.1	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 (+29.7)
	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 (+27.5)
	EATA-TTA	37.2	36.1	35.4	25.4	37.7	27.6	25.6	30.2	31.8	35.0	24.1	28.0	33.0	29.4	37.1	31.6
	EATA-CTTA	37.2	33.1	36.0	27.8	37.6	29.6	27.0	32.6	31.5	35.2	26.6	29.1	33.4	29.6	37.5	32.2
	EATA-DSS	37.2	33.1	35.4	25.4	37.7	27.6	25.6	30.2	31.8	35.0	24.1	28.0	33.0	29.4	37.1	31.4 (+0.8)
CIFAR-10C	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
	BN Stats	28.3	26.2	36.2	12.7	35.1	13.9	12.2	17.5	17.7	15.0	8.3	13.0	23.6	19.7	27.4	20.4
	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
	RMT	24.1	20.2	25.7	13.2	25.5	14.7	12.8	16.2	15.4	14.6	10.8	14.0	18.0	14.1	16.6	17.0
	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**
	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 (+0.2)
	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 (+1.5)
	EATA-TTA	24.6	22.4	32.1	11.3	31.9	13.0	10.8	16.2	16.0	13.2	8.0	10.6	21.5	17.0	23.7	18.1
	EATA-CTTA	24.6	19.1	27.7	12.8	29.4	14.5	12.1	16.3	15.8	15.2	9.3	13.0	21.6	16.1	20.8	17.9
	EATA-DSS	24.6	19.1	32.1	11.3	31.9	13.0	10.8	16.2	16.0	13.2	8.0	10.6	21.5	17.0	23.7	17.9 (0.0)

Table 2: Results as error percentages (lower is better) for all the datasets. The number in bracket shows the improvement in the performance of the underlying method with respect its CTTA version. In all cases we observe that the DSS version works at par with the TTA version and is an improvement over the CTTA version.

6.1. Datasets

In our study, we follow the experimental protocol outlined in reference [26] to evaluate the robustness of classification networks. We use CIFAR-10C, CIFAR-100C and ImageNet-C datasets which are designed to evaluate the robustness of classification networks. These datasets contain images that have been corrupted with 15 different types of corruptions at 5 different levels of severity. All experiments are conducted at the severity level 5. For ImageNet-C, we use the first 10,000 samples for each corruption instead of all the points in the dataset.

6.2. Baselines

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

(i) TTA: Here, the model is is set to source and the optimizer parameters are set to the initial states whenever there is a domain shift. This domain shift information is explicitly provided to the model following [25].

(ii) CTTA: Next, we consider the CTTA, as introduced by CoTTA [26]. Similar to the TTA setting, the continual benchmark uses an off-the-shelf model pre-trained on the source domain. However, this setting does not assume knowledge of when the domain changes and instead adapts the model online to a sequence of test domains. We continually adapt the model and do not reset anything outside the scope of the algorithms

(iii) DSS: Finally, we use our domain shift signal to mimic the TTA setting while in the CTTA setting to dynamically reset the model to the off-the-shelf model without having the underlying domain shift information. Thus the model can adapt to the changing distributions without error accumulation, effectively mitigating the negative impact of continuous domain shift. Here, we reset the model and the optimizer according to our domain shift signal.

6.3. 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[13], respectively. The CIFAR10 experiments use a WideResNet-28[31] model, while the CIFAR100 experiments use a ResNeXt-29[27] 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[6].

CoTTA[26] 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[25] and EATA[21], we use a learning rate of 1e-3 for all three datasets. For AaD[30], 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. For all methods, only the BN-layers are learnable, and they use the mean and variance of the test batch as the BN-layer statistics. We set k (Equation 5) to 3 for all experiments. We hope these details will help in the reproducibility of our results.

6.4. Computational Advantages

From Figure 3 and Table 3 we observe the computational advantages of using TENT-DSS or AaD-DSS compared to CoTTA and RMT. Figure 3 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 backpropagation. Next, Table 3 shows the memory requirements is lower in TENT and AaD compared to CoTTA and RMT due to less number of trainable parameters and models that need to be stored. These experiments were done on ImageNet-C using NVIDIA GeForce RTX 3090.

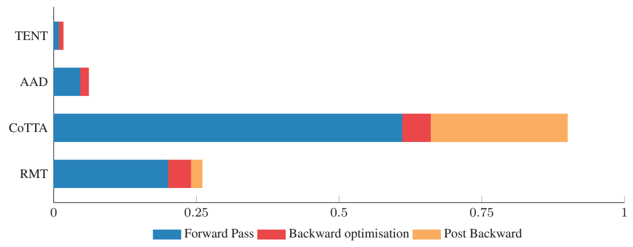


Figure 3: Comparison of inference time of the proposed SATA framework with the state-of-the-art CoTTA. The x-axis is time of inference per batch (sec/batch).

Method	# Parameters	# Trainable	% Trainable
TENT	25,557,160	128	0.0005
AaD	25,557,160	128	0.0005
CoTTA	76,671,096	25,557,032	33.333
RMT	51,114,064	25,557,032	50.000

Table 3: 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.

6.5. Discussion

In Table 2 we see that indeed when TTA methods are directly taken to CTTA setting their performance dramatically reduces. This is due to catastrophic forgetting caused by overfitting on current domain. The DSS approach brings these methods back to TTA setting without explicit domain information. We observed that the inclusion of the DSS module in TENT and EATA resulted in comparable or even better performance compared to the SoTA methods. This potentially brings the choice of algorithm in real life setting down to the computational requirement of the algorithms in which case the TTA methods are more efficient compared to the CTTA methods by orders of magnitude as shown in Table 3 and Figure 3.

7. Conclusion

In this work, we address the limitations of traditional single domain adaptation by developing a domain shift detection mechanism. We show that the source model can indeed characterize domain shift, by continually measuring the similarity between mean feature representations of consecutive batches. We leverage this simple signal to detect a domain shift, upon which our method resets the model back to the source and continues test-time adaptation. Incorporating this mechanism in TTA methods overcomes the problem of error accumulation and catastrophic forgetting in CTTA setting. Our experiments across standard datasets and single domain test-time adaptation baselines demonstrate the effectiveness of our approach, making it a promising solution for the CTTA problem.

Acknowledgements: This research is partially funded through a grant from SERB, Government of India (Grant No. SPF/2021/000118).

References

[1] Malik Boudiaf, Romain Mueller, Ismail Ben Ayed, and Luca Bertinetto. Parameter-free online test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8344–8353, 2022. 2

[2] Goirik Chakrabarty, Manogna Sreenivas, and Soma Biswas. Sata: Source anchoring and target alignment network for continual test time adaptation. *arXiv preprint arXiv:2304.10113*, 2023. 2, 5

[3] 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. 2

[4] 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. 2

[5] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 2017. 1

[6] 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. 6

[7] 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. 1

[8] Mario Döbler, Robert A Marsden, and Bin Yang. Robust mean teacher for continual and gradual test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7704–7714, 2023. 2, 5

[9] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010. 1

[10] Yulu Gan, Yan Bai, Yihang Lou, Xianzheng Ma, Renrui Zhang, Nian Shi, and Lin Luo. Decorate the newcomers: Visual domain prompt for continual test time adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 7595–7603, 2023. 2

[11] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35, 2016. 2

[12] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 1

[13] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019. 6

[14] Daehee Kim, Youngjun Yoo, Seunghyun Park, Jinkyu Kim, and Jaekoo Lee. Selfreg: Self-supervised contrastive regularization for domain generalization. In *ICCV*, 2021. 2

[15] Pan Li, Da Li, Wei Li, Shaogang Gong, Yanwei Fu, and Timothy M. Hospedales. A simple feature augmentation for domain generalization. In *ICCV*, 2021. 2

[16] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *ICML*, 2020. 2

[17] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *ECCV*, 2014. 1

[18] Jiaming Liu, Senqiao Yang, Peidong Jia, Ming Lu, Yandong Guo, Wei Xue, and Shanghang Zhang. Vida: Homeostatic visual domain adapter for continual test time adaptation. *arXiv preprint arXiv:2306.04344*, 2023. 2

[19] Mingsheng Long, ZHANGJIE CAO, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In *NeurIPS*, volume 31, 2018. 2

[20] Zhihe Lu, Yongxin Yang, Xiatian Zhu, Cong Liu, Yi-Zhe Song, and Tao Xiang. Stochastic classifiers for unsupervised domain adaptation. In *CVPR*, pages 9108–9117, 2020. 2

[21] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 16888–16905. PMLR, 17–23 Jul 2022. 1, 4, 5, 6

[22] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *NeurIPS*, 2015. 1

[23] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *CVPR*, 2018. 2

[24] Hui Tang, Ke Chen, and Kui Jia. Unsupervised domain adaptation via structurally regularized deep clustering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8725–8735, 2020. 2

[25] D. Wang, E. Shelhamer, S. Liu, B. Olshausen, and T. Darrell. Tent: Fully test-time adaptation by entropy minimization. In *ICLR*, 2021. 1, 2, 4, 5, 6

[26] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7201–7211, 2022. 2, 5, 6

[27] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, pages 1492–1500, 2017. 6

[28] Ruijia Xu, Guanbin Li, Jihan Yang, and Liang Lin. Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation. In *ICCV*, October 2019. 2

[29] Shiqi Yang, Joost van de Weijer, Luis Herranz, Shangling Jui, et al. Exploiting the intrinsic neighborhood structure for source-free domain adaptation. *Advances in neural information processing systems*, 34:29393–29405, 2021. 2

[30] Shiqi Yang, Yaxing Wang, Kai Wang, Shangling Jui, et al. Attracting and dispersing: A simple approach for source-free domain adaptation. In *Advances in Neural Information Processing Systems*, 2022. 1, 2, 4, 5, 6

[31] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, pages 87.1–87.12, 2016. 6

[32] K. Zhou, Y. Yang, Y. Qiao, and T. Xiang. Domain generalization with mixstyle. In *ICLR*, 2021. 2