

Adaptive Cascading Network for Continual Test-Time Adaptation

Kien X. Nguyen*
 Department of Computer and
 Information Sciences
 University of Delaware
 Newark, Delaware, USA
 kxnguyen@udel.edu

Fengchun Qiao*
 Department of Computer and
 Information Sciences
 University of Delaware
 Newark, Delaware, USA
 fengchun@udel.edu

Xi Peng
 Department of Computer and
 Information Sciences
 University of Delaware
 Newark, Delaware, USA
 xipeng@udel.edu

ABSTRACT

We study the problem of continual test-time adaptation where the goal is to adapt a source pre-trained model to a sequence of unlabelled target domains at test time. Existing methods on test-time training suffer from several limitations: (1) Mismatch between the feature extractor and classifier; (2) Interference between the main and self-supervised tasks; (3) Lack of the ability to quickly adapt to the current distribution. In light of these challenges, we propose a cascading paradigm that simultaneously updates the feature extractor and classifier at test time, mitigating the mismatch between them and enabling long-term model adaptation. The pre-training of our model is structured within a meta-learning framework, thereby minimizing the interference between the main and self-supervised tasks and encouraging fast adaptation in the presence of limited unlabelled data. Additionally, we introduce innovative evaluation metrics, *average accuracy* and *forward transfer*, to effectively measure the model’s adaptation capabilities in dynamic, real-world scenarios. Extensive experiments and ablation studies demonstrate the superiority of our approach in a range of tasks including image classification, text classification, and speech recognition. Our code is publicly available at <https://github.com/Nyquixt/CascadeTTA>.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning algorithms**.

KEYWORDS

Continual Test-time Adaptation, Self-supervised Learning, Transfer Learning

ACM Reference Format:

Kien X. Nguyen, Fengchun Qiao, and Xi Peng. 2024. Adaptive Cascading Network for Continual Test-Time Adaptation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24)*, October 21–25, 2024, Boise, ID, USA. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3627673.3679801>

*Both authors contributed equally to this research.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CIKM '24, October 21–25, 2024, Boise, ID, USA

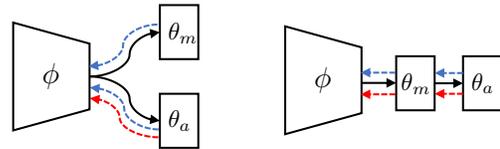
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-0436-9/24/10

<https://doi.org/10.1145/3627673.3679801>

ϕ : Feature Extractor θ_m : Main Classifier θ_a : Auxiliary Classifier

→ Forward ← Backward (Pre-train) ← Backward (Test)



(a) Parallel Paradigm

(b) Cascading Paradigm

Figure 1: (a) Parallel paradigm for test-time training [40], and (b) our proposed cascading paradigm for continual test-time training. The proposed cascading paradigm efficiently mitigates the mismatch between the feature extractor and main classifier, enabling long-term model adaptation.

1 INTRODUCTION

Intelligent systems operating in real-world settings frequently encounter non-stationary data distributions that evolve over time. For example, self-driving cars would face dynamically changing environments due to weather, lighting conditions, geographic locations, etc. This variability poses significant challenges to machine learning models as a highly accurate model on training data may fail catastrophically on shifted distributions [11? ? ? ?]. These challenges necessitate model retraining to assimilate new distributions, also known as continual learning [31]. However, most existing continual learning approaches are designed primarily for fully labeled datasets, which is prohibitively expensive.

Test-time adaptation [34, 40, 45?] is a promising approach to adapt a pre-trained model to unlabelled target data without access to training data. The essence of these methods lies in the incorporation of a self-supervised learning (SSL) task, e.g., predicting image rotation, minimizing the entropy loss, etc. Recently, Wang et al. [46] has extended the problem setting to continual test-time adaptation, where there is a sequence of target distributions at test time, to better reflect the real-world scenario. However, recent work Liu et al. [29] shows that the unconstrained model update from the SSL task may interfere with the main task. This interference results in the accumulation of prediction errors and a gradual deviation from the model’s true predictive mechanism, preventing long-term model adaptation. Moreover, existing methods lack the ability of quickly adapting (e.g., adapting with few gradient steps) the model to the current distribution and demand statistically sufficient data (e.g. large batch size). This issue is crucial for continual and non-stationary settings when test data arrive in an online manner with small batches. In addition, test-time training methods [3, 29, 40] typically employ a parallel paradigm (Fig. 1 (a)) with an auxiliary

classifier for the SSL task, thus only enabling feature update while keeping the main classifier fixed at test time. Continually updating the model in such a manner would lead to a growing misalignment between the feature extractor and the main classifier, leading to deteriorated test accuracy.

To address the issues, we propose a cascading paradigm for continual test-time adaptation. In contrast to previous parallel paradigm which only enables feature update, the proposed cascade paradigm (Fig. 1 (b)) synchronously modulates the feature extractor and main classifier at test-time, mitigating the mismatch between them and enables long-term model adaptation. To optimize the proposed cascading paradigm, we organize the model pre-training in a meta-learning framework, minimizing the interference between the main and SSL tasks and encouraging fast adaptation with limited amounts of unlabelled data.

Given the complexities identified in continual test-time adaptation, we recognize the need for more refined metrics in addition to those tailored for standard test-time adaptation [13]. Taking inspiration from the continual learning literature [43], we introduce two new evaluation metrics to the continual test-time adaptation setting: (1) *average accuracy* identifies whether the model has drifted away from the true prediction mechanisms at the end of the adaptation process, and (2) *forward transfer* evaluates the model’s capability to leverage knowledge from the past domains to adapt to the current one. To summarize, our contribution is multi-fold:

- We propose a cascading paradigm tailored for continual test-time adaptation to efficiently eliminate the mismatch between the feature extractor and classifier at test-time, enabling long-term model adaptation.
- We organize the model pre-training in a meta-learning framework to align the main and SSL tasks, meanwhile encouraging fast adaptation to target distributions in the presence of limited unlabelled data.
- We introduce new evaluation metrics, namely average accuracy and forward transfer, to further understand the model’s behavior in long-term adaptation.
- Extensive experiments demonstrate the superiority of our approach in a wide scope of tasks including image classification, text classification, and speech recognition.

2 RELATED WORK

Test-time Adaptation. Several methods have been developed to adapt pre-trained models to test data from shifted distributions without accessing source data. Liang et al. [27] used information maximization and pseudo-labeling for implicit alignment between target and source domains. Sun et al. [40] adapts the feature extractor using self-supervised tasks like image rotation prediction. Batch normalization methods, including re-estimating target domain normalization statistics [25] and introducing entropy-based updates [45], are also utilized. However, these methods often lack flexibility for continually changing distributions. Recently, Wang et al. [46] formulated continual test-time adaptation (CoTTA) to address the issue of catastrophic forgetting for non-stationary distributional shifts. Despite its effectiveness, CoTTA updates all parameters of model, degrading adaptation efficiency and risking overfitting to

data streams. This problem is also known as batch dependency, or *over-adaptation on previous test batches* [52].

Continual Learning. The objective is to learn progressively from tasks in sequence without erasing previously gained knowledge [7]. Existing methods can be divided into three categories: regularization-based [18, 26], memory replay [5, 35], and parameter isolation methods [1, 30]. Different from continual test-time adaptation, they focus on learning new tasks with fully labelled datasets. Several methods are proposed to adapt models to a sequence of unlabelled target domains, also known as continual domain adaptation [15, 49]. Liu et al. [28] suggested a meta-adaptation framework capable of capturing the evolving pattern of the target domain. Su et al. [39] proposed gradient regularized contrastive learning to learn discriminative and domain-invariant representations simultaneously. Lao et al. [20] introduced a modularized two-stream system which can handle both task and domain shifts. However, these methods rely on the co-existence of both the source and target domains, and cannot be applied directly in our problem. Lifelong domain adaptation [16, 36] lifts this restriction but necessitates estimating internal source distribution for adaptation.

Meta-learning. Meta-learning [37] is a long-standing topic on learning models to generalize over a distribution of tasks. Model-Agnostic Meta-Learning (MAML) by Finn et al. [9] was proposed to adapt the model to new tasks within a few gradient steps. The key idea is to learn a good initialization from which the model is able to be quickly adapted to new tasks with few-shot examples. Several approaches [?] have been proposed to use meta-learning to address distributional shifts. Li et al. [23] suggested an episodic training paradigm to improve models’ generalization capability. Balaji et al. [2] came up with MetaReg that meta-learned a regularization function that can generalize to new domains. Dou et al. [8] proposed to incorporate global and local constraints to learn semantic feature spaces in a modified MAML framework. MT3 [3] was formulated to incorporate meta-learning into test-time adaptation by learning task-specific model parameters for different tasks. However, MT3 requires the meta-model to be accessed at test time which makes continual adaptation for a single model unsuitable.

3 PROBLEM FORMULATION

The problem of continual test-time adaptation is defined by a pair of random variables (X, Y) over instances $x \in \mathcal{X} \subseteq \mathbb{R}^d$ and corresponding labels $y \in \mathcal{Y}$, where (X, Y) follows an unknown joint distribution $\mathbb{P}(X, Y)$. At training, we have access to a single source domain \mathcal{S} with labelled dataset $\mathcal{D}_{\mathcal{S}} = \{\mathbf{x}_s^i, \mathbf{y}_s^i\}_{i=1}^{n_s}$, where $\mathcal{D}_{\mathcal{S}} \sim \mathbb{P}_{\mathcal{S}}(X, Y)$. At test time, we are presented with a sequence of target domains $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_N\}$ where for each \mathcal{T}_i we only have access to an unlabelled dataset $\mathcal{D}_{\mathcal{T}_i} = \{\mathbf{x}_{\mathcal{T}_i}^k\}_{k=1}^{n_{\mathcal{T}_i}}$ and $\mathbf{x}_{\mathcal{T}_i}$ arrive in an online manner with small batches while $\mathcal{D}_{\mathcal{S}}$ is not available:

$$\mathbf{x}_{\mathcal{T}_i} \stackrel{i.i.d.}{\sim} \mathcal{D}_{\mathcal{T}_i} \text{ where } \mathcal{D}_{\mathcal{T}_1} \rightarrow \mathcal{D}_{\mathcal{T}_2}, \dots, \rightarrow \mathcal{D}_{\mathcal{T}_N} \sim \mathbb{P}_{\mathcal{T}}.$$

The objective is to learn a predictor $f_{\psi} : \mathcal{X} \rightarrow \mathcal{Y}$ to predict labels $\{\mathbf{y}_{\mathcal{T}_i}^k\}_{k=1}^{n_{\mathcal{T}_i}}$ for each \mathcal{T}_i , where ψ are learnable model parameters. Typically, f_{ψ} is decomposed into a feature extractor $h_{\phi} : \mathcal{X} \rightarrow \mathcal{Z} \subset \mathbb{R}^P$ and a classifier $g_{\theta} : \mathcal{Z} \rightarrow \mathcal{Y}$, i.e., $f_{\psi} = g_{\theta} \circ h_{\phi}$, where $\psi = (\phi, \theta)$.

Evaluation Metrics

The first evaluation metric is the standard mean online error, which measures the average classification error across all batches during adaptation, denoted as $\mathcal{E}(\psi)$. In addition, motivated by continual learning [43], we propose to evaluate the model in terms of average accuracy and forward transfer. Let $\mathcal{R}_\psi(\mathcal{T}_i|\mathcal{T}_j)$ denote the test accuracy on domain \mathcal{T}_i after observing \mathcal{T}_j . We measure the average accuracy $\mathcal{A}(\psi)$ on all domains at the end of adaptation:

$$\mathcal{A}(\psi) = \frac{1}{N} \sum_{t=1}^N \mathcal{R}_\psi(\mathcal{T}_t|\mathcal{T}_{1:N}). \quad (1)$$

Second, we evaluate forward transfer by measuring the accuracy difference between a model that was updated through the sequence of past domains and a model that was only updated by the last domain:

$$\mathcal{F}(\psi) = \frac{1}{N-1} \sum_{t=2}^N \mathcal{R}_\psi(\mathcal{T}_t|\mathcal{T}_{1:t}) - \mathcal{R}_\psi(\mathcal{T}_t|\mathcal{T}_t). \quad (2)$$

The higher the better for two metrics. A high $\mathcal{A}(\psi)$ denotes that the model does not deviate from the true prediction mechanism; a positive $\mathcal{F}(\psi)$ indicates the model’s ability to leverage knowledge from previous domains.

4 METHOD

We introduce a cascading paradigm, as illustrated in Fig. 1 (b), tailored for continual test-time adaptation. During training, we update the model on the source domain through a main and a self-supervised learning (SSL) task. At test time, the feature extractor and classifier are concurrently updated through the SSL task. This simultaneous update diminishes discrepancies between them, paving the way for sustained model adaptation to target domains. While a straightforward approach to achieve the cascading paradigm might involve pre-training the model on both the main and SSL tasks using multi-task learning, recent research by [29] indicates potential pitfalls: in multi-task learning, unrestricted updates from the SSL task can inadvertently conflict with the main task, potentially undermining rather than enhancing test accuracy. Furthermore, multi-task learning does not inherently ensure rapid adaptation to target distributions. To address these challenges, we propose a meta-learning framework during model pre-training. This framework enforces gradient alignment between the two tasks, effectively reducing task interference and improving test accuracy.

4.1 Cascading Paradigm

Domain Adaptation [4] is widely used to address distributional shifts without supervision. However, it requires the co-existence of both the source and target domains. Test-time training [40] was proposed to address this issue by leveraging a SSL task for model adaptation. This method employs a **parallel paradigm** (Fig. 1 (a)): a main classifier θ_m for the supervised learning task, an auxiliary classifier θ_a for the SSL task, and the two classifiers share the same feature extractor ϕ . All modules are updated in pre-training by multi-task learning, while only the feature extractor is updated at test-time through the SSL task. However, continually updating the model in such manner would gradually increase the discrepancy

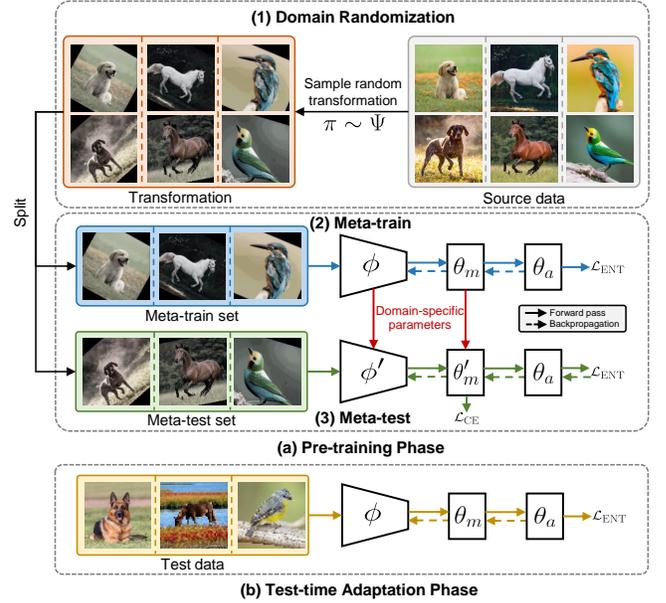


Figure 2: Overview of pre-training and adaptation phases. $\pi \sim \Psi$ denotes a transformation randomly sampled from a predefined pool of transformations. \mathcal{L}_{ENT} : entropy loss, \mathcal{L}_{CE} : cross-entropy loss.

between the feature extractor and main classifier, accumulating prediction errors and deviating from the true prediction mechanism.

To address this issue, we propose a **cascading paradigm** (Fig. 1 (b)) to synchronously modulate the feature extractor and classifier for continual test-time adaptation. Specifically, we reorganize the architecture in a sequential manner, starting with the feature extractor ϕ , followed by the main classifier θ_m and finally the auxiliary classifier θ_a in that order. At test time, the self-supervised loss calculated on the output of θ_a is utilized to synchronously update both ϕ and θ_m .

4.2 Model Pre-training

To optimize the proposed cascading paradigm, a straightforward way is to pre-train the model on both the main and SSL task through multi-task learning. However, recent work by [29] shows that in multi-task learning, the unconstrained model update from the SSL task may interfere with the main task, deteriorating the test accuracy rather than improving it. Moreover, multi-task learning cannot guarantee fast adaptation to target distributions. To address these issues, we propose a meta-learning framework for model pre-training to enforce the gradient alignment between the two tasks, mitigating task interference and enabling fast adaptation with only a single gradient step. Following [32, 45], we employ the entropy loss as the self-supervised loss. Our pre-training phase, which includes meta-train and meta-test steps, is structured in a manner that aligns with gradient-based meta-learning approaches like MAML [9].

Domain Randomization. First, we create simulated non-stationary target domains that the model would encounter during test time to support the meta-learning scheme. We employ *domain randomization* [44?] to generate domain augmentations from the single

source \mathcal{S} through a set of transformations Ψ . Given Ψ and the source samples $\{(\mathbf{x}_s, \mathbf{y}_s)\} \sim \mathbb{P}_{\mathcal{S}}(X, Y)$, we generate augmentations by sampling a specific transformation $\pi \sim \Psi$, and then applying it to the given data, obtaining $\{(\pi(\mathbf{x}_s), \mathbf{y}_s)\} \sim p(\mathcal{S}^+)$. For each iteration, we sample a mini-batch $\tau \sim \mathbb{P}_{\mathcal{S}^+}(X, Y) \tau \sim \mathbb{P}_{\mathcal{S}^+}(X, Y)$ representing a random domain. We then divide τ into meta-train data $\mathcal{D}_{\tau}^{\text{trn}}$ and meta-validation data $\mathcal{D}_{\tau}^{\text{val}}$ for the meta-learning pre-training phase. Domain randomization varies across different modalities. The implementation details are provided in Sec. 6 and Appendix A.

Meta-train. In the inner loop, to mimic the model adaptation at test-time, we update ϕ and θ_m using the gradients of entropy calculated on the output of θ_a . Specifically, let $\psi = \{\phi, \theta_m\}$, we update ψ via one step stochastic gradient descent on $\mathcal{D}_{\tau}^{\text{trn}}$ while fixing θ_a :

$$\psi' \leftarrow \psi - \alpha \nabla_{\psi} \mathbb{E}_{\tau \sim \mathbb{P}_{\mathcal{S}^+}} [\mathcal{L}_{\text{ENT}}(\psi, \theta_a; \mathcal{D}_{\tau}^{\text{trn}})], \quad (3)$$

where α is the learning rate of the inner loop and \mathcal{L}_{ENT} is the entropy loss. At test time, to avoid overadapting to the data stream and mitigate the risk of catastrophic forgetting, we only update BN parameters of ϕ . Inspired by the work of [25], [38], and [45], we re-estimate the statistical moments and update the affine parameters. **Meta-test.** In the outer loop, we update ψ and θ_a using the gradient of the meta-loss $\mathcal{L}_{\text{Meta}}$ on $\mathcal{D}_{\tau}^{\text{val}}$:

$$\begin{aligned} \{\psi, \theta_a\} &\leftarrow \{\psi, \theta_a\} - \beta \nabla_{\{\psi, \theta_a\}} \mathcal{L}_{\text{Meta}}(\psi', \theta_a; \mathcal{D}_{\tau}^{\text{val}}), \\ \mathcal{L}_{\text{Meta}} &= \mathbb{E}_{\tau \sim \mathbb{P}_{\mathcal{S}^+}} \left[\mathcal{L}_{\text{CE}}(\psi'; \mathcal{D}_{\tau}^{\text{val}}) + \lambda \mathcal{L}_{\text{ENT}}(\psi', \theta_a; \mathcal{D}_{\tau}^{\text{val}}) \right], \end{aligned} \quad (4)$$

where β is the learning rate of the outer loop, \mathcal{L}_{CE} denotes cross-entropy loss, and λ is the balancing coefficient. Intuitively, the meta-learning framework distills knowledge from unlabelled samples and leverages it to facilitate supervised classification. Moreover, the meta-learned initialization enables the model to fast adapt to target domains using limited unlabeled samples. With a few steps of gradient update from the meta-learned representation, the model can produce accurate predictions on the incoming data stream. This capability further allows the model to be reliably adapted long-term without deviating too far from the meta-learned initialization, empirically illustrated in Sec. 6.1. The pre-training procedure is summarized in Algorithm 1.

It is worth noting that although Tent and CoTTA do not require special treatment to model pre-training, they require statistically sufficient data for model adaptation. We empirically show that our approach outperforms CoTTA by $\sim 9\%$ in accuracy with the batch size of 16 (see Sec. 6.4).

5 THEORETICAL ANALYSIS

We first show that the meta-learning framework encourages gradient alignment between the main and self-supervised learning (SSL) tasks. Next, we show this alignment further upper-bounds the generalization error by adopting the \mathcal{H} -divergence [4].

Theorem 1. (Gradient alignment between the main and SSL tasks). Let $\mathcal{L}_{\text{Main}}(\psi)$ be the loss function for the main task and $\mathcal{L}_{\text{SSL}}(\psi, \theta_a)$ be the loss function for the SSL task. Suppose the model parameters are updated using Eq. (4). Then, the gradient of the meta-objective

Algorithm 1: Pre-training for Cascading Paradigm.

Input: Source domain \mathcal{S} , transformations Ψ

Output: Learned ψ and θ_a

```

1 while not converged do
2   Sample data  $\{(\mathbf{x}_s, \mathbf{y}_s)\} \sim \mathbb{P}_{\mathcal{S}}(X, Y)$ ;
3   Sample a transformation  $\pi \sim \Psi$ ;
4   Generate augmentations  $\{(\pi(\mathbf{x}_s), \mathbf{y}_s)\}$ ;
5   Split  $\{(\pi(\mathbf{x}_s), \mathbf{y}_s)\}$  into  $\{\mathcal{D}_{\tau}^{\text{trn}}, \mathcal{D}_{\tau}^{\text{val}}\}$ ;
6   Meta-train: Get  $\psi'$  using  $\mathcal{D}_{\tau}^{\text{trn}}$  via Eq. (3);
7   Meta-test: Update  $\{\psi, \theta_a\}$  on  $\mathcal{D}_{\tau}^{\text{val}}$  via Eq. (4);
8 end

```

$\mathcal{L}_{\text{Meta}}$ with respect to (ψ, θ_a) can be decomposed as:

$$(I - \beta \nabla_{(\psi, \theta_a)}^2 \mathcal{L}_{\text{SSL}}(\psi', \theta_a))^{\top} (\nabla_{\psi} \mathcal{L}_{\text{Main}}(\psi') + \lambda \nabla_{(\psi, \theta_a)} \mathcal{L}_{\text{SSL}}(\psi', \theta_a)),$$

where I is the identity matrix and $\nabla_{(\psi, \theta_a)}^2 \mathcal{L}_{\text{SSL}}(\psi', \theta_a)$ is the Hessian matrix of \mathcal{L}_{SSL} with respect to (ψ, θ_a) . If the Hessian matrix is positive semi-definite and the learning rate β is sufficiently small (small β ensures that the first-order approximation is valid), the gradient of the SSL task does not interfere with the gradient of the main task, but rather provides useful information for adaptation.

Theorem 2. (Generalization Upper Bound. Adapted from [28]). Assuming $d_{\mathcal{H}\Delta\mathcal{H}}(\mathbb{P}_{\mathcal{T}_i}, \mathbb{P}_{\mathcal{T}_j}) \leq \alpha |t_i - t_j|$ holds with constant α for $t_i, t_j \geq 0$, then for any ψ , with probability at least $1 - \delta$ over the sequence of N target domains \mathcal{T} :

$$\begin{aligned} \mathbb{E}_t \mathbb{E}_{\mathcal{P}_{\mathcal{T}}} \mathcal{L}(f_{\psi}(\mathbf{x}), \mathbf{y}) &\leq \mathbb{E}_{\mathcal{P}_{\mathcal{S}}} \mathcal{L}(f_{\psi}(\mathbf{x}), \mathbf{y}) + \frac{1}{N} \sum_{i=1}^N [d_{\mathcal{H}\Delta\mathcal{H}}(\mathbb{P}_{\mathcal{S}}, \mathbb{P}_{\mathcal{T}_i})] \\ &\quad + \mathbb{E}_t \lambda_t + O\left(\frac{\alpha}{\delta N}\right), \end{aligned}$$

where $\lambda_t = \min_{\psi} [\mathbb{E}_{\mathcal{P}_{\mathcal{S}}} \mathcal{L}(f_{\psi}(\mathbf{x}), \mathbf{y}) + \mathbb{E}_{\mathcal{P}_{\mathcal{T}}} \mathcal{L}(f_{\psi}(\mathbf{x}), \mathbf{y})]$ measures the adaptability from source to target. Our method reduces generalization risks on target domains ($\mathbb{E}_{\mathcal{P}_{\mathcal{T}}}$ in λ_t) by aligning the self-supervised learning (SSL) and main tasks.

6 EXPERIMENTS

We evaluate our approach on three modalities using five benchmark datasets: *CIFAR-10-C*, *CIFAR-100-C* and *Tiny-ImageNet-C* [13] for image classification, *Amazon Reviews* [6] for text classification, and *Google Commands* [47] for speech recognition. Section 6.4 includes ablation studies to investigate key components of the proposed cascading paradigm. We include source code, implementation details, and more experimental results in the supplementary.

Baselines. We compare the proposed cascading paradigm to the following baselines: (1) Empirical Risk Minimization (ERM) [42]: is trained on the source domain and directly evaluated on target domains without any update. (2) Adaptive Batch Normalization (AdaBN) [24]: re-estimate normalization statistics on each incoming batch in target domains. (3) Test-time training (TTT) [40]: use a self-supervised learning task, *i.e.*, predicting image rotation, to update the feature extractor at test time. (4) Tent [45]: update both normalization statistics and affine parameters by entropy minimization, modified to fit the continual setting by not resetting during the adaptation process. (5) CoTTA [46]: continually adapt a source

Table 1: Results (%) on *CIFAR-10/100-C* and *Tiny-ImageNet-C* with the highest corruption severity on the instantaneously changing setup. Models are pre-trained on the original *CIFAR-10/100* and *Tiny-ImageNet* and continually adapted to a sequence of corruptions with a batch size of 32 for *CIFAR* and 64 for *Tiny-ImageNet*. Our method significantly outperforms other baselines in online error $\mathcal{E}(\psi)$, average accuracy $\mathcal{A}(\psi)$, and forward transfer $\mathcal{F}(\psi)$.

Dataset	Method	$t \rightarrow$														$\mathcal{E}(\psi) \downarrow$	$\mathcal{A}(\psi) \uparrow$	$\mathcal{F}(\psi) \uparrow$	
		gauss	shot	impul	defoc	glass	motn	zoom	snow	frost	fog	brit	contr	elast	pixel				jpeg
CIFAR-10-C	ERM	73.06	67.73	73.38	23.71	64.87	34.53	27.08	33.41	47.54	19.49	10.97	23.14	39.14	73.25	36.70	43.20	56.80	-
	AdaBN	40.20	37.65	40.63	22.46	52.14	26.03	23.01	27.10	30.20	23.86	12.82	21.13	36.37	33.56	35.31	30.83	69.17	-
	TTT	40.61	29.31	37.04	32.68	41.92	34.13	21.29	28.10	22.47	24.25	16.22	24.45	30.01	25.97	24.20	28.84	64.38	-4.42
	Tent	27.83	23.08	29.89	20.60	42.48	28.79	26.81	34.92	38.45	38.88	31.71	41.08	49.51	50.70	50.34	35.67	47.69	-31.11
	CoTTA	25.63	22.46	27.09	19.75	35.35	23.10	21.33	25.16	24.91	29.15	19.15	33.40	33.22	29.04	28.87	26.51	66.18	-7.49
	Ours	28.64	25.36	32.38	15.52	40.55	19.55	16.77	20.65	20.78	19.81	11.34	16.86	28.36	24.16	24.70	22.99	77.36	0.58
CIFAR-100-C	ERM	94.24	91.36	91.52	55.00	87.42	62.58	56.06	67.12	76.52	57.58	42.42	66.97	65.15	90.91	67.73	71.51	28.49	-
	AdaBN	78.33	75.91	74.55	52.27	76.21	58.18	55.00	64.70	61.67	58.64	43.64	60.61	64.09	66.06	70.30	64.01	36.12	-
	TTT	72.97	68.18	72.86	72.93	77.54	74.06	57.36	73.42	62.37	60.33	48.48	71.81	61.77	60.55	62.00	66.44	26.49	-12.64
	Tent	76.52	71.21	68.18	48.79	71.52	52.88	45.15	55.61	53.33	52.42	41.21	54.55	55.00	52.27	59.39	57.20	38.84	-2.30
	CoTTA	68.79	70.61	69.39	57.88	70.00	61.36	55.61	61.36	56.67	62.27	45.00	75.61	64.09	60.00	60.30	62.60	35.37	1.52
	Ours	66.42	64.78	65.65	44.33	67.20	47.89	46.01	54.11	54.35	53.75	39.85	49.87	57.46	54.66	60.87	55.11	45.23	0.19
Tiny ImageNet-C	ERM	86.06	83.94	95.00	88.64	92.27	73.79	77.27	65.45	61.21	75.45	57.42	95.45	81.97	64.55	48.94	76.49	23.51	-
	AdaBN	85.84	83.43	89.35	82.70	91.00	74.67	75.98	75.23	74.43	78.94	70.90	94.44	77.76	72.01	73.80	80.03	19.97	-
	TTT	95.29	78.44	79.32	82.74	88.30	81.13	67.35	75.47	65.57	69.71	60.50	96.02	67.32	59.40	57.01	74.90	20.25	-11.16
	Tent	85.22	82.79	88.23	84.40	91.61	80.31	82.26	83.49	84.17	86.33	83.14	96.48	89.76	88.28	89.90	86.42	7.72	-12.58
	CoTTA	83.49	80.37	87.86	79.22	89.18	69.65	71.10	70.83	69.49	74.49	64.67	93.07	73.35	65.33	68.00	76.01	24.60	1.29
	Ours	74.62	70.12	80.72	77.56	83.63	54.13	54.04	59.22	53.69	63.77	49.34	91.72	63.29	53.84	49.56	65.28	27.39	-12.27

pre-trained model to non-stationary target data by reducing error accumulation and alleviating forgetting.

For fair comparison, we use our pre-trained model as the backbone for all baselines except ERM and TTT, since TTT has its own pre-training strategy. In contrast to previous work that utilizes pre-trained models such as WideResNet-28 [50], we train all models from scratch and refrain from using the data augmentations that coincide with the corruptions/domains at test time to avoid test data leakage, accurately measuring the adapting capability of the algorithms.

Metrics. 1) Online error. We immediately record the online prediction of each batch after the model was adapted to it. We calculate the online prediction error, $\mathcal{E}(\psi)$, by averaging the errors of all target domains. **2) Continual learning metrics.** We further evaluate the cascading paradigm using the metrics proposed in Section 3: average accuracy $\mathcal{A}(\psi)$ and forward transfer $\mathcal{F}(\psi)$.

6.1 Image Classification

For image classification, We validate our method on *CIFAR-10/100-C* and *Tiny-ImageNet-C*.

Dataset. *CIFAR-10/100-C* and *Tiny-ImageNet-C* [13] is a robustness benchmark consisting of fifteen corruptions types with five levels of severity applied to the test set of *CIFAR-10/100* [19] and *Tiny-ImageNet* [21]. The corruptions consist of four main categories: noise, blur, weather, and digital. We show the model performance on the highest severity.

Setup. Following [40], we use 15 corruptions as target domains. The model is pre-trained on the original *CIFAR-10/100* and *Tiny-ImageNet* datasets and continually adapted to a sequence of image corruptions in *CIFAR-10/100-C* and *Tiny-ImageNet-C*. We use ResNet-26 [12] for the first two datasets and ResNet-34 for the third dataset. We append a lightweight 2-layer fully connected network as the auxiliary classifier. During pre-training, the initial learning

rate is 0.001 with a linear decay and the number of epochs is 75. We use AugMix [14] for domain randomization. At test time, the SGD optimizer with Nesterov momentum [41] with an online learning rate of 0.001 updates the model. As we wish to simulate the online setting, we use a small batch size of 32 for *CIFAR-10/100-C* and 64 for the more challenging benchmark *Tiny-ImageNet-C*.

Table 2: Results (%) on *CIFAR-10-C* on the gradually changing setup. Models are pre-trained on *CIFAR-10* and continually adapted to a sequence of 135 gradually changing domains with a batch size of 32. Our method enables long-term adaptation to target domains while others fail catastrophically.

Metrics	$\mathcal{E}(\psi) \downarrow$	$\mathcal{A}(\psi) \uparrow$	$\mathcal{F}(\psi) \uparrow$
TTT	28.39	60.29	-13.94
Tent	64.26	11.39	-72.83
CoTTA	41.08	26.93	-54.83
Ours	17.51	81.19	-2.4

Results. 1) Instantaneously changing setup. Tab. 1 shows the results of *CIFAR-10-C*, *CIFAR-100-C*, and *Tiny-ImageNet-C* with the highest corruption severity on the standard domain sequence. Our method outperforms other baselines in most corruptions and yields the lowest average error of 22.99%, 55.11%, and 65.28% on the three benchmarks, respectively. For forward transfer, our model achieves a $\mathcal{F}(\psi)$ of 0.58% and 0.19% on *CIFAR-10/100-C*, showcasing its capability to adapt to current domains by leveraging past knowledge. However, on the more challenging benchmark *Tiny-ImageNet-C*, it suffers a relatively high negative $\mathcal{F}(\psi)$ of -12.27 . Other baselines exhibit lower performance on online error with negative forward transfer, except for CoTTA on *CIFAR-100-C*, and *Tiny-ImageNet-C* with a forward transfer of 1.52%, and 1.29%. The results underscore

our method’s efficiency in continual adaptation to various image corruptions.

2) Gradually changing setup. Next, we evaluate our model on the gradually changing setup, which is more relevant to the real-world scenario. In the standard sequence of corruptions, target domains change abruptly, especially in the highest severity. Instead, following [46], we construct a sequence of target domains such that the evolving pattern of distributional drift is smoother. Specifically, for each corruption t we continuously change its severity level from 1 to 5 and then back to 1 and then change the corruption type, making up 135 target domains:

$$\underbrace{\dots 2 \rightarrow 1}_{t-1 \text{ and before}} \rightarrow \underbrace{1 \rightarrow 2 \rightarrow \dots 5 \rightarrow 4 \rightarrow \dots 1}_{\text{corruption } t} \rightarrow \underbrace{1 \rightarrow 2 \dots}_{t+1 \text{ and after}}$$

This setup evaluates the model’s capability of long-term adaptation. Per Tab. 2, Tent and CoTTA, with the need of statistically sufficient data, *i.e.*, a large batch size, perform rather poorly. They both suffer from error accumulation over a lengthy sequence of domains and eventually deviate from the true prediction mechanism. Since CoTTA updates all model parameters, continually adapting to small batches causes the model to overfit to the current batch, preventing it from efficiently adapt to the subsequent ones. In contrast, our model maintains stability over an extended domain sequence, achieving a substantial 81.19% average accuracy with minimal forward transfer losses. Our approach alleviates batch dependency [52] by successfully adapting over 135 diverse domains.

6.2 Text Classification

In this section, we conduct experiments on *Amazon Reviews* by [6] for text classification.

Dataset. *Amazon Reviews* is a widely adopted benchmark in the context of domain adaptation for sentiment classification. It is a collection of product reviews from amazon.com in four product domains: books, dvds, electronics, and kitchen appliances. Reviews are assigned with binary labels - 0 (or “negative”) if the rating of the product is up to 3 stars, and 1 (or “positive”) if the rating is 4 or 5 stars. We use unigrams and bigrams as features resulting in 5000 dimensional representations [10]. Following [48], we leverage the four operations for text augmentation: synonym replacement, random insertion, random swap and random deletion. See Appx. A.2 for more details on augmentations.

Setup. We pre-train the model on one source domain “books”, and continually adapt the model to “dvds”, “electronics”, and “kitchen”. Similar to [10], the extracted features are fed into two FC layers with the size of 50. A softmax layer with the size of two is used to classify the sentiment of reviews into “positive” or “negative”. We append a BN layer to each hidden layer and append a FC layer with the size of 2 as the auxiliary classifier. We use Adam [17] to optimize the model, and the initial learning rate $\eta_0 = 10^{-4}$. Similar to the image classification task, the batch size is set to 32.

Results. Tab. 3 shows the results of text classification on *Amazon Reviews*. Results show that our method outperforms others across all test domains by a large margin. Specially, our method achieves the lowest online prediction error of 21.73% and the highest average accuracy of 78.73% across all past domains at the end of the adaptation. Notably, our model achieves 4.82% in forward transfer,

Table 3: Results (%) on *Amazon Reviews*. Models are pre-trained on “books” and continually adapted to “dvds” → “electronics” → “kitchen” with a batch size of 32.

Method	$t \rightarrow$			$\mathcal{E}(\psi) \downarrow$	$\mathcal{A}(\psi) \uparrow$	$\mathcal{F}(\psi) \uparrow$
	dvds	electronics	kitchen			
ERM	19.78	26.50	20.06	22.11	77.89	-
AdaBN	20.47	25.03	20.99	22.17	77.83	-
Tent	20.87	25.26	21.85	22.66	77.26	-1.08
Ours	19.19	25.00	21.01	21.73	78.73	4.82

indicating the model can leverage knowledge from past domains to accelerate the adaptation to the current domain. Results demonstrates the effectiveness of our method in continually adapting to different text domains.

6.3 Speech Recognition

In this section, we perform experiments on *Google Commands* by [47] for speech recognition.

Table 4: Results (%) on *Google Commands*. Models are pre-trained on the clean audios and continually adapted to “Amp.” → “Pit.” → “Noise” → “Stretch” → “Shift”.

Method	$t \rightarrow$					$\mathcal{E}(\psi) \downarrow$	$\mathcal{A}(\psi) \uparrow$	$\mathcal{F}(\psi) \uparrow$
	Amp.	Pit.	Noise	Stretch	Shift			
ERM	36.5	26.7	27.4	25.8	31.7	29.62	70.38	-
AdaBN	36.8	26.1	26.9	22.5	28.3	28.12	71.88	-
Tent	36.1	23.6	24.8	20.2	28.5	26.64	72.25	-23.15
Ours	35.4	21.2	21.9	20.6	27.3	25.28	74.66	-6.04

Dataset. *Google Commands* has 65000 utterances (one second long) from thousands of people. There are 30 different command words in total. There are 56196, 7477, and 6835 examples for training, validation, and test. To simulate domain shift in real-world scenarios, we apply five common corruptions in both time and frequency domains. This creates five test sets that are “harder” than training sets, namely amplitude change (Amp.), pitch change (Pit.), background noise (Noise), stretch (Stretch), and time shift (Shift). The range of “amplitude change” is (0.7,1.1). The maximum scales of “pitch change”, “background noise”, and “stretch” are 0.2, 0.45, and 0.2, respectively. The maximum shift of “time shift” is 8. See Appx. A.3 for more details on data preprocessing and augmentations. We use the default hyper-parameters from audiomentations¹.

Setup. We pre-train the model on the clean train set, and continually adapt it to “Amp.”, “Pit.”, “Noise”, “Stretch”, and “Shift”. We encode each audio into a Mel-spectrogram with the size of $1 \times 32 \times 32$ and feed them to LeNet [22] as one-channel input.

Results. Tab. 4 shows the results of speech recognition on *Google Commands*. As seen, our method outperforms other baselines on all target domains except “stretch”, indicating its strong adaptation capability on corruptions in both time and frequency domains. In detail, our method outperforms the second best by 0.7% on “amplitude change”, 2.4% on “pitch change”, 2.9% on “background noise”, and

¹<https://github.com/iver56/audiomentations>

Table 5: Online classification error (%) on *CIFAR-10-C* across different batch sizes on the instantaneously changing setting. Our method shows consistent performance across different batch sizes, and outperforms Tent and CoTTA by at least 9% with a batch size of 16. We report the mean and standard deviation across all batch sizes.

Batch Size	128	64	32	16	Mean \pm Std
Tent	22.53	25.99	35.67	50.28	33.61 \pm 12.4
CoTTA	21.75	22.96	26.51	33.58	26.20 \pm 5.3
TTT	28.15	27.02	28.84	27.15	27.79 \pm 0.7
Ours	22.26	22.32	22.99	24.63	23.05 \pm 1.1

1.0% on “time shift”, respectively. Our method suffers significantly less deterioration in forward transfer compared to Tent.

6.4 Ablation Study

We have shown that our method yields significant improvements across different modalities including image, text, and speech. In this section, we perform ablation study to investigate the key components of the proposed cascading paradigm and provide *main observations* as follows:

Our model is insensitive to batch sizes. In online adaptation, we expect the batch size to be relatively small and varied; an online model should perform well on such conditions. We show the classification error on *CIFAR-10-C* across different batch sizes in Tab. 5. A smaller standard deviation indicates that a model is less sensitive to the batch size. CoTTA’s performance drops by 11.83% when the batch size reduces from 128 to 16 due to overfitting on early batches with insufficient statistical input. Tent, with limited data, shows 50.28% average error for batch size 16. In contrast, our method exhibits the lowest error, remaining consistent across batch sizes with a 0.9% standard deviation, showcasing its insensitivity and online adaptability. It is also worth noting that both TTT and our method are insensitive to different batch sizes, possibly due to the usage of the auxiliary classifier.

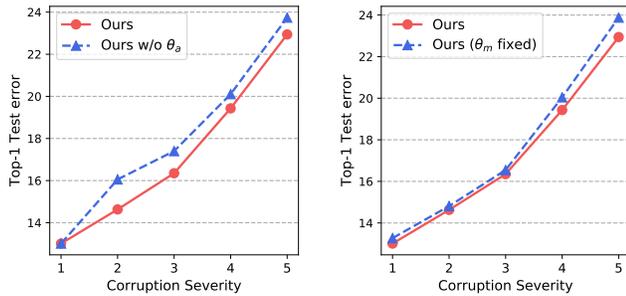


Figure 3: Validation of the cascading paradigm on *CIFAR-10-C*. Left: Classification error of models w/ and w/o the auxiliary classifier θ_a . Right: Classification error of models updating and fixing the main classifier θ_m at test time.

Auxiliary classifier effectively synchronizes modules. The cascading paradigm allows us to synchronously modulate the feature extractor ϕ and main classifier θ_m through self-supervised

Table 6: Ablation study on the effectiveness of θ_a . Online classification error (%) on *Tiny-ImageNet-C* on 5 levels of corruption severity.

Methods	1	2	3	4	5
w/o θ_a	46.07	50.46	56.47	63.82	68.09
Ours	44.20	48.36	54.14	61.45	65.28

learning at test time. To validate its effectiveness, we conduct two ablation studies on *CIFAR-10-C* dataset: (1) We discard θ_a and compute the entropy of the output of θ_m . In Fig. 3 (left), we observe that θ_a consistently improves the classification performance across all levels of corruption severity, possibly due to that θ_a converts the logits into another space with less intervention to the cross-entropy. (2) We only modulate BN-related parameters and keep θ_m fixed at test time. In Fig. 3 (right), we can see that our model outperforms the variant with θ_m fixed during test, validating the effectiveness of jointly modulating both ϕ and θ_m . We further conduct the ablation study on *Tiny-ImageNet-C*, a more challenging dataset, to confirm the effectiveness of θ_a , as depicted in Tab. 6.

Table 7: Ablation study on the effect of meta-learning on *CIFAR-10-C* across different batch sizes.

Method	Meta	128	64	32	16
TTT	✗	28.15	27.02	28.84	27.15
TTT	✓	24.04	24.37	27.93	28.20
Ours	✗	27.93	47.18	66.53	75.88
Ours	✓	22.26	22.32	22.99	24.63

Meta-learning improves model robustness. The meta-learning approach aligns gradients between self-supervised and supervised losses and enables effective initialization. This allows the model to rapidly adapt to unlabeled target domains with limited data. To validate its effectiveness, we create a model variant using conventional multi-task learning (MTL). In this variant, we update the model via gradients from the combined entropy and cross-entropy loss:

$$\{\psi, \theta_a\} \leftarrow \{\psi, \theta_a\} - \alpha \nabla_{\{\psi, \theta_a\}} \mathcal{L}_{\text{MTL}}(\psi, \theta_a; \mathcal{D}_\tau),$$

$$\mathcal{L}_{\text{MTL}} = \mathbb{E}_{\tau \sim \mathbb{P}_{S^+}} [\mathcal{L}_{\text{CE}}(\psi; \mathcal{D}_\tau) + \lambda \mathcal{L}_{\text{ENT}}(\psi, \theta_a; \mathcal{D}_\tau)].$$

Furthermore, we apply meta-learning on TTT and achieve a significant improvement in performance across different batch sizes, as shown in Tab. 7. Similar to Eq. 3 and Eq. 4 described in Sec. 4.2, we adapt the inner and outer loop update to TTT respectively as:

$$\phi' \leftarrow \phi - \alpha \nabla_{\phi} \mathbb{E}_{\tau \sim \mathbb{P}_{S^+}} [\mathcal{L}_{\text{R}}(\phi, \theta_a; \mathcal{D}_\tau^{\text{tm}})], \quad (5)$$

and

$$\{\psi, \theta_a\} \leftarrow \{\psi, \theta_a\} - \beta \nabla_{\{\psi, \theta_a\}} \mathcal{L}_{\text{Meta}}(\phi', \theta_m, \theta_a; \mathcal{D}_\tau^{\text{val}}), \quad (6)$$

$$\mathcal{L}_{\text{Meta}} = \mathbb{E}_{\tau} [\mathcal{L}_{\text{CE}}(\phi', \theta_m; \mathcal{D}_\tau^{\text{val}}) + \lambda \mathcal{L}_{\text{R}}(\phi', \theta_a; \mathcal{D}_\tau^{\text{val}})],$$

where \mathcal{L}_{R} is the image rotation prediction loss. Noticeably, it boosts TTT’s accuracy with the batch size of 128 by 4.11%. Another critical observation is that without meta-learning our cascading paradigm

exhibits a poor performance for small batch sizes and has to rely on sufficient data statistics to yield a decent accuracy. Results demonstrate the effectiveness of meta-learning in adapting to few samples.

6.5 Uncertainty Quantification

In our approach, we use entropy as the self-supervised loss, which is used to quantify the uncertainty. To investigate the relationship between uncertainty and classification error, we analyze the evolution of entropy and error. Results are shown in Fig. 4. In Fig. 4 (left), we note that both entropy and error decrease when the model adapts to more samples of the target domain. In Fig. 4 (right), we can see that both entropy and error increase with the level of severity. Entropy shows consistent trends with the classification error in both cases. The results demonstrate that entropy can reflect the model’s uncertainty with respect to target domains as well as measure the distributional distance between the source and target domain.

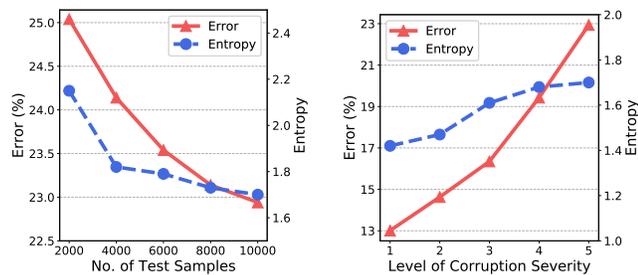


Figure 4: Uncertainty quantification on *CIFAR-10-C*. Entropy and error after adapting to different numbers of test samples (left) and different levels of corruption severity (right). Entropy shows consistent trends with the classification error.

7 CONCLUSION

We proposed a cascading paradigm for continual test-time adaptation. In contrast to the parallel paradigm which only enables feature updates, the cascade paradigm synchronously modulates the feature extractor and classifier at test-time, mitigating the mismatch between them and enabling long-term model adaptation. To minimize the interference between the main and self-supervised tasks, we propose a meta-learning framework for model pre-training to align the gradients between the supervised and self-supervised losses. Moreover, the meta-learning framework facilitates fast adaptation to target distributions using limited unlabelled data. Extensive experiments as well as ablation studies demonstrate the superiority of our approach on a range of tasks including image classification, text classification, and speech recognition.

ACKNOWLEDGMENTS

This work is supported by the National Science Foundation through the Faculty Early Career Development Program (NSF CAREER) Award NSF-IIS-2340074 and the Department of Defense under the Defense Established Program to Stimulate Competitive Research (DoD DEPSCoR) Award.

A IMPLEMENTATION DETAILS

We provide the details of network architectures used for the three modalities in our experiments. Following Fig. 1, we will describe the design in terms of the feature extractor ϕ , the main classifier θ_m and the auxiliary classifier θ_a for easy interpretation. All experiments are conducted on a single GeForce RTX 2080 Ti GPU with 12G memory.

A.1 Image

A.1.1 *CIFAR-10/100-C*. We use ResNet-26 [12] as the backbone. The input dimension is $3 \times 32 \times 32$.

ϕ : The first layer of the model contains a 3×3 convolution (Conv) layer with 16 output channels. The input is then passed through three ResNet layers, each having four basic blocks. Each basic block consists of two Conv 3×3 layers with the same number of channels. The number of channels for each ResNet layer is {16, 32, 64}. Each Conv is followed by batch normalization (BN) and a rectifier linear unit (ReLU). After all the Conv layers, the network produces a feature map of size $64 \times 8 \times 8$, reduced to $64 \times 1 \times 1$ after pooling.

θ_m : The feature map is flattened and fed into a fully connected (FC) layer of size 10/100 to generate the logits before applying Softmax for the final prediction.

θ_a : We then append a two-layer FC network with size {64, 10/100} that take the logits from θ_m as the input.

A.1.2 *Tiny-ImageNet-C*. We use ResNet-34 [12] as the backbone. The input is resized to $3 \times 224 \times 224$.

ϕ : The module follows the official PyTorch [33] implementation.

θ_m : The feature map is flattened and fed into an FC layer of size 200 to generate the logits before applying the softmax activation function for the final prediction.

θ_a : We then append a two-layer FC network with size {1024, 200} that take the logits from θ_m as the input.

We use the Stochastic Gradient Descent optimizer with Nesterov momentum [41] and set the batch size to 32 for *CIFAR-10/100-C* and 64 for *Tiny-ImageNet-C*. During pre-training, the initial learning rate is 0.001 with a linear decay and the number of epochs is 75. During adaptation, the learning rate is set to 0.001.

A.2 Text

We employ four data augmentations: (1) Synonym replacement: randomly choose words from the sentence that are not stop words and replace each of them with one of their synonyms chosen at random; (2) Random insertion: find a random synonym of a random word in the sentence that is not a stop word and insert that synonym into a random position in the sentence; (3) Random swap: randomly choose a pair of words in the sentence and swap their positions; (4) Random deletion: randomly remove each word in the sentence with a certain probability. For all augmentations, we use the hyper-parameters from [48]. We use a simple FC network. Raw text samples are first converted into the mSDA [6] representation.

ϕ : The mSDA vector is then fed into two FC layers with the hidden dimensions of 1024 and 512.

θ_m : The output feature is then passed into an FC layer of size 2 and a softmax layer for binary prediction.

θ_a : Another FC layer of size 2 is appended to θ_m .

Table 8: Results (%) on CIFAR-10-C with corruption level ranging from 1 to 4 following the instantaneously changing setup. Models are pre-trained on the original CIFAR-10 and continually adapted to a sequence of corruptions with a batch size of 32. Our method significantly outperforms other baselines in online error $\mathcal{E}(\psi)$, average accuracy $\mathcal{A}(\psi)$, and forward transfer $\mathcal{F}(\psi)$.

Level	Method	$t \rightarrow$															$\mathcal{E}(\psi) \downarrow$	$\mathcal{A}(\psi) \uparrow$	$\mathcal{F}(\psi) \uparrow$
		gauss	shot	impul	defoc	glass	motn	zoom	snow	frost	fog	brit	contr	elast	pixel	jpeg			
4	ERM	68.23	57.38	54.03	13.99	66.91	27.14	20.42	29.06	35.58	11.44	9.86	12.11	27.59	57.57	32.91	34.95	65.05	-
	AdaBN	37.13	32.78	32.83	15.53	51.74	22.21	19.08	27.53	25.37	15.61	11.96	15.95	26.26	24.18	32.14	26.02	73.98	-
	TTT	36.94	24.67	30.79	20.42	40.74	25.44	17.95	27.31	18.49	14.97	12.64	15.35	22.18	19.65	21.58	23.27	73.09	0.13
	Tent	27.58	20.17	23.50	14.18	37.28	19.16	15.77	23.20	19.59	14.39	12.77	14.18	21.87	17.61	21.61	20.19	79.31	-1.55
	CoTTA	26.47	23.80	26.69	18.61	43.18	25.55	24.70	28.05	26.82	25.11	17.67	31.18	36.98	31.83	35.06	28.11	63.32	-12.35
	Ours	25.64	21.43	25.54	12.08	40.49	16.27	14.41	21.38	17.36	13.08	10.01	13.07	20.05	17.97	22.62	19.43	80.65	-0.06
3	ERM	62.71	49.81	30.51	11.30	55.19	27.97	17.07	26.83	33.17	10.01	9.62	10.73	16.60	33.55	28.99	28.27	71.73	-
	AdaBN	34.24	29.07	24.69	13.14	39.59	22.09	16.97	25.50	23.81	13.80	11.94	14.39	18.48	18.28	29.93	22.39	77.61	-
	TTT	35.07	22.45	24.02	14.71	30.25	25.08	16.67	23.46	18.01	12.64	11.19	12.86	15.42	15.00	19.59	19.82	78.23	1.99
	Tent	25.25	17.92	18.23	11.83	27.75	18.52	14.49	19.64	18.37	12.86	11.84	12.89	15.55	14.90	20.27	17.35	82.41	-1.11
	CoTTA	24.75	21.55	21.01	15.33	31.85	24.39	22.25	24.67	25.55	20.31	16.36	25.01	29.41	25.81	31.79	24.00	68.89	-10.45
	Ours	23.47	18.85	17.80	9.97	28.74	16.51	12.86	18.84	17.06	11.18	9.76	11.48	14.24	13.81	20.76	16.35	83.64	-0.26
2	ERM	46.90	28.94	21.62	9.54	56.80	19.92	14.27	31.57	21.03	9.54	9.21	10.00	14.29	24.50	25.88	22.93	77.07	-
	AdaBN	27.66	21.47	20.29	11.95	38.96	17.88	15.02	25.55	18.75	12.61	11.76	13.24	17.00	16.30	26.68	19.67	80.33	-
	TTT	29.04	16.88	21.81	11.52	31.61	20.59	14.72	23.47	15.46	11.06	10.29	11.25	14.65	13.73	18.51	17.64	81.74	2.68
	Tent	21.22	14.21	15.33	10.49	27.98	15.44	12.79	19.53	14.98	11.51	10.95	11.69	14.86	13.50	17.80	15.49	85.76	-0.42
	CoTTA	20.58	16.33	17.67	13.29	31.33	20.23	19.69	23.14	21.12	17.41	15.46	21.03	24.92	23.51	29.21	20.99	72.15	-9.64
	Ours	19.48	14.43	15.14	9.65	28.36	13.73	11.37	18.13	13.87	10.24	9.35	10.84	13.13	12.79	18.98	14.63	85.42	-0.42
1	ERM	27.70	19.21	15.06	8.80	59.08	13.64	13.44	17.35	14.61	8.90	8.83	9.00	14.37	13.32	17.98	17.42	82.58	-
	AdaBN	20.42	17.93	15.20	11.69	38.83	14.90	14.51	17.44	16.05	11.83	11.65	12.01	17.22	14.37	20.79	16.99	83.01	-
	TTT	23.69	14.64	17.01	9.90	33.45	15.58	14.21	16.24	11.77	9.98	9.57	9.81	13.31	11.72	15.18	15.07	85.45	3.61
	Tent	16.10	12.54	12.37	9.70	26.92	13.08	12.32	14.19	12.21	10.31	10.32	10.44	14.71	11.51	14.38	13.41	86.72	0.09
	CoTTA	15.95	14.74	14.41	12.07	30.40	16.58	17.82	17.47	17.72	15.41	14.63	16.67	24.28	20.19	24.07	18.16	76.26	-7.72
	Ours	14.97	12.73	12.12	9.45	28.98	11.51	11.26	13.22	11.70	9.67	9.46	9.88	13.79	11.39	14.99	13.01	86.87	-0.75

We append a BN layer to each hidden layer. We use Adam [17] to optimize the model for 1000 iterations, and the initial learning rate $\eta_0 = 10^{-4}$. We adopt the inverse-decay strategy of DANN [51], where the learning rate changes by $\eta_p = \frac{\eta_0}{(1+\omega p)^\phi}$, $\omega = 10$, $\phi = 0.75$, and p is the progress ranging from 0 to 1. Similar to image classification, the batch size is 32, simulating the online setting.

A.3 Speech

In *Google Commands* [47], the range of “amplitude change” is (0.7,1.1). The maximum scales of “pitch change”, “background noise”, and “stretch” are 0.2, 0.45, and 0.2, respectively. The maximum shift of “time shift” is 8. For audio augmentation, we leverage the following three operations: (1) Gain: multiply audio by a random amplitude factor to lower or raise volume; (2) High-pass filtering: apply parameterized filter steepness to input audio; (3) Impulse response: convolve audio with a randomly selected impulse response. The mel-spectrogram features of dimension $1 \times 32 \times 32$ are fed into LeNet [22] as the 1-channel image.

ϕ : The original image is fed into two 5×5 Conv layers with the channels of 6 and 16, outputting a feature of dimension $16 \times 5 \times 5$.

θ_m : The feature then go through two FC layers with the size of 120 and 84, respectively. The output is a 30-dimensional vector before applying softmax to predict the spoken word.

θ_a : An FC layer of size 30 follows θ_m .

We append a BN layer to each Conv layer. Models are trained using SGD with a learning rate of 0.1 linearly reduced to 0.001 for 50 epochs.

Table 9: Average time taken to adapt to each corruption on CIFAR-10/100-C and Tiny-ImageNet-C.

Methods	CIFAR-10-C	CIFAR-100-C	Tiny-ImageNet-C
CoTTA	24 sec	36 sec	175 sec
Ours	7 sec	7 sec	19 sec

B ADDITIONAL RESULTS

We report the results on *CIFAR-10-C* in the instantaneously changing setup on severity levels 4 to 1 (Tab. 8). We use the same baselines as in Sec. 6 of the main paper: ERM [42], AdaBN [24], TTT [40], Tent [45], and CoTTA [46]. Overall, our method consistently outperforms others in terms of $\mathcal{E}(\psi)$ and $\mathcal{A}(\psi)$ across the four corruption severity levels. Notably, CoTTA displays a substantial loss in both $\mathcal{F}(\psi)$ possibly due to two reasons. First, the error accumulation due to insufficient statistics (*i.e.*, small batch size) deteriorates the model performance over time, leading to the incapability to transfer past domains’ knowledge to the current. Second, by updating all the parameters, CoTTA suffers from catastrophic forgetting since the model overfits to the data stream. Experiments show that only updating a portion of the learnable parameters helps retain past knowledge, thus avoiding forgetting.

C INFERENCE TIME

Different from CoTTA which updates all the parameters of the feature extractor, our method only updates BN parameters. Therefore, our methods are an order of magnitude faster than CoTTA (Tab. 9).

REFERENCES

- [1] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. 2017. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3366–3375.
- [2] Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. 2018. Metareg: Towards domain generalization using meta-regularization. In *NeurIPS*. 998–1008.
- [3] Alexander Bartler, Andreas Bühler, Felix Wiewel, Mario Döbler, and Binh Yang. 2021. MT3: Meta Test-Time Training for Self-Supervised Test-Time Adaption. In *International Conference on Artificial Intelligence and Statistics*. <https://api.semanticscholar.org/CorpusID:232417890>
- [4] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. 2006. Analysis of representations for domain adaptation. *Advances in neural information processing systems* 19 (2006).
- [5] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. 2018. Efficient Lifelong Learning with A-GEM. In *International Conference on Learning Representations*.
- [6] Minmin Chen, Zhixiang Xu, Kilian Q Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. In *ICML*. 1627–1634.
- [7] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. 2021. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence* 44, 7 (2021), 3366–3385.
- [8] Qi Dou, Daniel Coelho de Castro, Konstantinos Kamnitsas, and Ben Glocker. 2019. Domain generalization via model-agnostic learning of semantic features. In *NeurIPS*.
- [9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*.
- [10] Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised Domain Adaptation by Backpropagation. In *ICML*. 1180–1189.
- [11] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.
- [12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*. 770–778.
- [13] Dan Hendrycks and Thomas Dietterich. 2019. Benchmarking neural network robustness to common corruptions and perturbations. *ICLR* (2019).
- [14] Dan Hendrycks, Norman Mu, Ekin D. Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. 2020. AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty. *ICLR* (2020).
- [15] Judy Hoffman, Trevor Darrell, and Kate Saenko. 2014. Continuous Manifold Based Adaptation for Evolving Visual Domains. *2014 IEEE Conference on Computer Vision and Pattern Recognition* (2014), 867–874. <https://api.semanticscholar.org/CorpusID:10105727>
- [16] Zhipeng Huang, Zhizheng Zhang, Cuiling Lan, Wenjun Zeng, Peng Chu, Quanzeng You, Jiang Wang, Zicheng Liu, and Zheng-jun Zha. 2022. Lifelong unsupervised domain adaptive person re-identification with coordinated anti-forgetting and adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14288–14297.
- [17] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. In *arXiv:1412.6980 [cs.LG]*.
- [18] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences* 114, 13 (2017), 3521–3526.
- [19] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).
- [20] Qicheng Lao, Xiangxi Jiang, Mohammad Havaei, and Yoshua Bengio. 2021. A Two-Stream Continual Learning System With Variational Domain-Agnostic Feature Replay. *IEEE Transactions on Neural Networks and Learning Systems* 33 (2021), 4466–4478. <https://api.semanticscholar.org/CorpusID:232113812>
- [21] Y. Le and X. Yang. 2015. Tiny ImageNet Visual Recognition Challenge.
- [22] Yann Lecun et al. 1998. Gradient-Based Learning Applied to Document Recognition. *Proc. IEEE* 86, 11 (1998).
- [23] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M. Hospedales. 2017. Learning to Generalize: Meta-Learning for Domain Generalization. In *AAAI Conference on Artificial Intelligence*. <https://api.semanticscholar.org/CorpusID:1883787>
- [24] Yanghao Li, Naiyan Wang, Jianping Shi, Xiaodi Hou, and Jiaying Liu. 2018. Adaptive Batch Normalization for practical domain adaptation. *Pattern Recognit.* 80 (2018), 109–117.
- [25] Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. 2016. Revisiting Batch Normalization For Practical Domain Adaptation. *ArXiv abs/1603.04779* (2016). <https://api.semanticscholar.org/CorpusID:5069968>
- [26] Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence* 40, 12 (2017), 2935–2947.
- [27] Jian Liang, Dapeng Hu, and Jiashi Feng. 2020. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International Conference on Machine Learning*. PMLR, 6028–6039.
- [28] Hong Liu, Mingsheng Long, Jianmin Wang, and Yu Wang. 2020. Learning to Adapt to Evolving Domains. In *Neural Information Processing Systems*. <https://api.semanticscholar.org/CorpusID:227275334>
- [29] Yuejiang Liu, Parth Kothari, Bastien van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. 2021. Ttt++: When does self-supervised test-time training fail or thrive? *NeurIPS* (2021).
- [30] Arun Mallya and Svetlana Lazebnik. 2018. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 7765–7773.
- [31] Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*. Vol. 24. Elsevier, 109–165.
- [32] Shuaicheng Niu, Jiayang Wu, Yifan Zhang, Yaofu Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. 2022. Efficient test-time model adaptation without forgetting. In *International conference on machine learning*. PMLR, 16888–16905.
- [33] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems* 32. Curran Associates, Inc., 8024–8035. <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>
- [34] Mihir Prabhudesai, Anirudh Goyal, Sujoy Paul, Sjoerd Van Steenkiste, Mehdi SM Sajjadi, Gaurav Aggarwal, Thomas Kipf, Deepak Pathak, and Katerina Fragkiadaki. 2023. Test-time adaptation with slot-centric models. In *International Conference on Machine Learning*. PMLR, 28151–28166.
- [35] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2001–2010.
- [36] Mohammad Rostami. 2021. Lifelong domain adaptation via consolidated internal distribution. *Advances in Neural Information Processing Systems* 34 (2021), 11172–11183.
- [37] Jürgen Schmidhuber. 1987. *Evolutionary principles in self-referential learning*. Ph. D. Dissertation. Technische Universität München.
- [38] Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. 2020. Improving robustness against common corruptions by covariate shift adaptation. *ArXiv abs/2006.16971* (2020). <https://api.semanticscholar.org/CorpusID:220266097>
- [39] Peng Su, Shixiang Tang, Peng Gao, Di Qiu, Ni Zhao, and Xiaogang Wang. 2020. Gradient Regularized Contrastive Learning for Continual Domain Adaptation. In *AAAI Conference on Artificial Intelligence*. <https://api.semanticscholar.org/CorpusID:220793718>
- [40] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. 2020. Test-Time Training with Self-Supervision for Generalization under Distribution Shifts. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*, Hal Daumé III and Aarti Singh (Eds.). PMLR, 9229–9248. <https://proceedings.mlr.press/v119/sun20b.html>
- [41] Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. 2013. On the importance of initialization and momentum in deep learning. In *ICML*.
- [42] Vladimir Vapnik. 1998. Statistical learning theory.
- [43] Tom Véniat, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2021. Efficient Continual Learning with Modular Networks and Task-Driven Priors. *ArXiv abs/2012.12631* (2021).
- [44] Riccardo Volpi, Diane Larlus, and Grégory Rogez. 2021. Continual adaptation of visual representations via domain randomization and meta-learning. In *CVPR*.
- [45] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno A. Olshausen, and Trevor Darrell. 2021. Tent: Fully Test-Time Adaptation by Entropy Minimization. In *International Conference on Learning Representations*. <https://api.semanticscholar.org/CorpusID:232278031>
- [46] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. 2022. Continual test-time domain adaptation. In *CVPR*.
- [47] Pete Warden. 2018. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209* (2018).
- [48] Jason Wei and Kai Zou. 2019. EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 6382–6388.
- [49] Markus Wulfmeier, Alex Bewley, and Ingmar Posner. 2018. Incremental adversarial domain adaptation for continually changing environments. In *2018 IEEE International conference on robotics and automation (ICRA)*. IEEE, 4489–4495.
- [50] Sergey Zagoruyko and Nikos Komodakis. 2016. Wide Residual Networks. *ArXiv abs/1605.07146* (2016). <https://api.semanticscholar.org/CorpusID:15276198>

- [51] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2017. Understanding deep learning requires rethinking generalization. In *ICLR*.
- [52] Hao Zhao, Yuejiang Liu, Alexandre Alahi, and Tao Lin. 2023. On Pitfalls of Test-Time Adaptation. arXiv:2306.03536 [cs.LG]