EcoTTA: Memory-Efficient Continual Test-time Adaptation via Self-distilled Regularization

Junha Song[1,2*], Jungsoo Lee[1], In So Kweon[2], Sungha Choi[1†]
[1]Qualcomm AI Research[‡], [2]KAIST

Abstract

This paper presents a simple yet effective approach that improves continual test-time adaptation (TTA) in a memory-efficient manner. TTA may primarily be conducted on edge devices with limited memory, so reducing memory is crucial but has been overlooked in previous TTA studies. In addition, long-term adaptation often leads to catastrophic forgetting and error accumulation, which hinders applying TTA in real-world deployments. Our approach consists of two components to address these issues. First, we present lightweight meta networks that can adapt the frozen original networks to the target domain. This novel architecture minimizes memory consumption by decreasing the size of intermediate activations required for backpropagation. Second, our novel self-distilled regularization controls the output of the meta networks not to deviate significantly from the output of the frozen original networks, thereby preserving well-trained knowledge from the source domain. Without additional memory, this regularization prevents error accumulation and catastrophic forgetting, resulting in stable performance even in long-term test-time adaptation. We demonstrate that our simple yet effective strategy outperforms other state-of-the-art methods on various benchmarks for image classification and semantic segmentation tasks. Notably, our proposed method with ResNet-50 and WideResNet-40 takes 86% and 80% less memory than the recent state-of-the-art method, CoTTA.

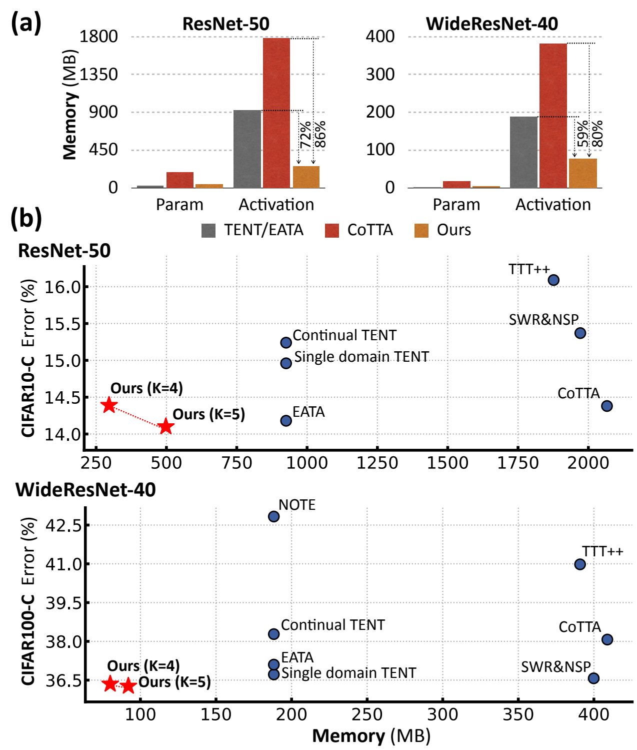


Figure 1. **(a) Memory cost comparison between TTA methods**. The size of activations, not the parameters, is the primary memory bottleneck during training. **(b) CIFAR-C adaptation performance.** We perform the continual online adaptation on CIFAR-C dataset. The x- and y-axis are the average error of all corruptions and the total memory consumption including the parameters and activations, respectively. Our approach, EcoTTA, achieves the best results while consuming the least amount of memory, where K is the model partition factor used in our method.

1. Introduction

Despite recent advances in deep learning [14, 21, 20, 19], deep neural networks often suffer from performance degradation when the source and target domains differ significantly [8, 36, 31]. Among several tasks addressing such domain shifts, test-time adaptation (TTA) has recently received a significant amount of attention due to its practicality and wide applicability especially in on-device settings [53, 35, 26, 15]. This task focuses on adapting the model to unlabeled online data from the target domain without access to the source data.

While existing TTA methods show improved TTA performances, minimizing the sizes of memory resources have been relatively under-explored, which is crucial considering the applicability of TTA in on-device settings. For example, several studies [54, 35, 9] update entire model parameters

*Work done during an internship at Qualcomm AI Research.
†Corresponding author. ‡ Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

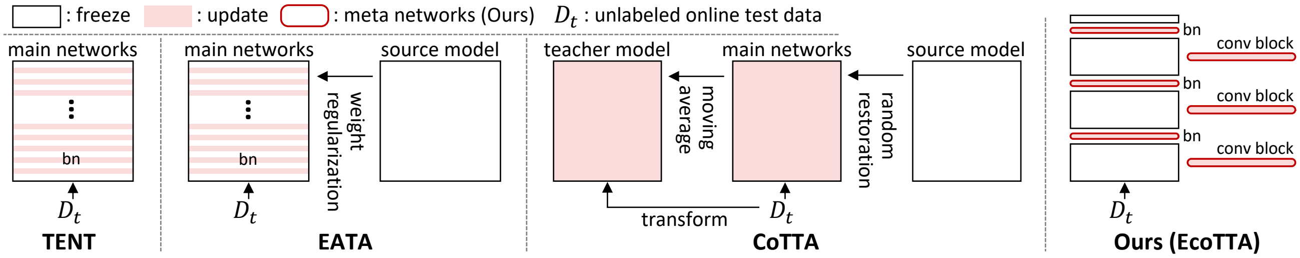


Figure 2. **Architecture for test-time adaptation.** We illustrate TTA methods: TENT [53], EATA [41], CoTTA [54], and Ours (EcoTTA). TENT and EATA update *multiple* batch norm layers, in which large activations have to be stored for gradient calculation. In CoTTA, an entire network is trained with additional strategies for continual adaptation that requires a significant amount of both memory and time. In contrast, our approach requires a minimum size of activations by updating only *a few* layers. Also, stable long-term adaptation is performed by our proposed regularization, named self-distilled regularization.

to achieve large performance improvements, which may be impractical when the available memory sizes are limited. Meanwhile, several TTA approaches update only the batch normalization (BN) parameters [53, 41, 16] to make the optimization efficient and stable However, even updating only BN parameters is not memory efficient enough since the amount of memory required for training models significantly depends on the size of intermediate activations rather than the learnable parameters [4, 13, 57]. Throughout the paper, activations refer to the intermediate features stored during the forward propagation, which are used for gradient calculations during backpropagation. Fig. 1 (a) demonstrates such an issue.

Moreover, a non-trivial number of TTA studies assume a stationary target domain [53, 35, 9, 48], but the target domain may continuously change in the real world (*e.g.,* continuous changes in weather conditions, illuminations, and location [8] in autonomous driving). Therefore, it is necessary to consider long-term TTA in an environment where the target domain constantly varies. However, there exist two challenging issues: 1) catastrophic forgetting [54, 41] and 2) error accumulation. Catastrophic forgetting refers to degraded performance on the source domain due to long-term adaptation to target domains [54, 41]. Such an issue is important since the test samples in the real world may come from diverse domains, including the source and target domains [41]. Also, since target labels are unavailable, TTA relies on noisy unsupervised losses, such as entropy minimization [17], so long-term continual TTA may lead to error accumulation [63, 2].

To address these challenges, we propose memory-**E**fficient **co**ntinual **T**est-**T**ime **A**daptation (EcoTTA), a simple yet effective approach for 1) enhancing memory efficiency and 2) preventing catastrophic forgetting and error accumulation. First, we present a memory-efficient architecture consisting of frozen original networks and our proposed meta networks attached to the original ones. During the test time, we freeze the original networks to discard the intermediate activations that occupy a significant amount of memory. Instead, we only adapt lightweight meta networks to the target domain, composed of only one batch normalization and one convolution block. Surprisingly, updating only the meta networks, not the original ones, can result in significant performance improvement as well as considerable memory savings. Moreover, we propose a self-distilled regularization method to prevent catastrophic forgetting and error accumulation. Our regularization leverages the preserved source knowledge distilled from the frozen original networks to regularize the meta networks. Specifically, we control the output of the meta networks not to deviate from the one extracted by the original networks significantly. Notably, our regularization leads to negligible overhead because it requires no extra memory and is performed in parallel with adaptation loss, such as entropy minimization.

Recent TTA studies require access to the source data *before model deployments* [35, 9, 28, 1, 33, 41]. Similarly, our method uses the source data to warm up the newly attached meta networks for a small number of epochs before model deployment. If the source dataset is publicly available or the owner of the pre-trained model tries to adapt the model to a target domain, access to the source data is feasible [9]. Here, we emphasize that pre-trained original networks are frozen throughout our process, and our method is applicable to any pre-trained model because it is agnostic to the architecture and pre-training method of the original networks.

Our paper presents the following contributions:

- We present novel meta networks that help the frozen original networks adapt to the target domain. This architecture significantly minimize memory consumption up to 86% by reducing the activation sizes of the original networks.
- We propose a self-distilled regularization that controls the output of meta networks by leveraging the output of frozen original networks to preserve the source knowledge and prevent error accumulation.
- We improve both memory efficiency and TTA performance compared to existing state-of-the-art methods on 1) image classification task (*e.g.*, CIFAR10/100-C and ImageNet-C) and 2) semantic segmentation task (*e.g.*, Cityscapes with weather corruption)

2. Related Work

Mitigating domain shift. One of the fundamental issues of DNNs is the performance degradation due to the domain shift between the train (*i.e.* source) and test (*i.e.* target) distributions. Several research fields attempt to address this problem, such as unsupervised domain adaptation [52, 6, 44, 47, 38, 49] and domain generalization [64, 8]. In particular, domain generalization aims to learn invariant representation so as to cover the possible shifts of test data. They simulate the possible shifts using a single or multiple source dataset [64, 62, 32] or force to minimize the dependence on style information [43, 8]. However, it is challenging to handle all potential test shifts using the given source datasets [18]. Thus, instead of enhancing generalization ability during the training time, TTA [53] overcomes the domain shift by directly adapting to the test data.

Test-time adaptation. Test-time adaptation allows the model to adapt to the test data (*i.e.*, target domain) in a source-free and online manner [27, 50, 53]. Existing works improve TTA performance with sophisticated designs of unsupervised loss [39, 60, 35, 9, 37, 48, 5, 15, 1, 3, 12] or enhance the usability of small batch sizes [30, 58, 25, 42, 33] considering streaming test data. They focus on improving the adaptation performance with a stationary target domain (*i.e.*, single domain TTA setup). In such a setting, the model that finished adaptation to a given target domain is reset to the original model pre-trained with the source domain in order to adapt to the next target domain.

Recently, CoTTA [54] has proposed continual TTA setup to address TTA under a continuously changing target domain which also involves a long-term adaptation. This setup frequently suffers from error accumulation [63, 2, 51] and catastrophic forgetting [54, 29, 41]. Specifically, performing a long-term adaptation exposes the model to unsupervised loss from unlabeled test data for a long time, so errors are accumulated significantly. Also, the model focuses on learning new knowledge and forgets about the source knowledge, which becomes problematic when the model needs to correctly classify the test sample as similar to the source distribution. To address such issues, CoTTA [54] randomly restores the updated parameters to the source one, while EATA [41] proposed a weight regularization loss.

Efficient on-device learning. Since the edge device is likely to be memory constrained (*e.g.*, a Raspberry Pi with 512MB and iPhone 13 with 4GB), it is necessary to take account of the memory usage when deploying the models on the device [34]. TinyTL [4], a seminal work in on-device learning, shows that the activation size, not learnable parameters, bottlenecks the training memory. Following this, recent on-device learning studies [4, 56, 57] targeting fine-tuning task attempt to decrease the size of intermediate activations. In contrast, previous TTA studies [53, 41] have overlooked these facts and instead focused on reducing learnable parameters. This paper, therefore, proposes a method that not only reduces the high activation sizes required for TTA, but also improves adaptation performance.

3. Approach

Fig. 3 illustrates our simple yet effective approach which only updates the newly added meta networks on the target domain while regularizing them with the knowledge distilled from the frozen original network. This section describes how such a design promotes memory efficiency and prevents error accumulation and catastrophic forgetting which are frequently observed in long-term adaptation.

3.1. Memory-efficient Architecture

Prerequisite. We first formulate the forward and the backward propagation. Assume that the i^{th} linear layer in the model consists of weight $\mathcal{W}$ and bias b, and the input and output features of this layer are f_i and f_{i+1}, respectively. Given that the forward propagation of $f_{i+1} = f_i\mathcal{W} + b$, the backward propagation from the $i{+}1^{th}$ layer to the i^{th} layer, and the weight gradient are respectively formulated as:

$$\frac{\partial \mathcal{L}}{\partial f_i} = \frac{\partial \mathcal{L}}{\partial f_{i+1}}\mathcal{W}^T, \quad \frac{\partial \mathcal{L}}{\partial \mathcal{W}} = f_i^T \frac{\partial \mathcal{L}}{\partial f_{i+1}}. \tag{1}$$

Eq. (1) means that the learnable layers whose weight $\mathcal{W}$ need to be updated must store intermediate activations f_i to compute the weight gradient. In contrast, the backward propagation in frozen layers can be accomplished without saving the activations, only requiring its weight $\mathcal{W}$. Further descriptions are provided in Appendix A.

TinyTL [4] shows that activations occupy the majority of the memory required for training the model rather than learnable parameters. Due to this fact, updating the entire model (e.g., CoTTA [54]) requires a substantial amount of memory. Also, updating only parameters in batch normalization (BN) layers (e.g., TENT [53] and EATA [41]) is not an effective approach enough since they still save the large intermediate activations for *multiple* BN layers. While previous studies fail to reduce memory by utilizing large activations, this work proposes a simple yet effective way to reduce a significant amount of memory by discarding them.

Before deployment. As illustrated in Fig. 3 (a, b), we first take a pre-trained model using any pre-training method. We divide the encoder of the pre-trained model into K number of parts and attach lightweight meta networks to each part of the original network. The details of how to divide the model into K number of parts are explained in the next section. One group of meta network composes of one batch normalization layer and one convolution block (*i.e.*, Conv-BN-Relu). Before the deployment, we pre-train the meta networks on the source dataset $\mathcal{D}_s$ for a small number of

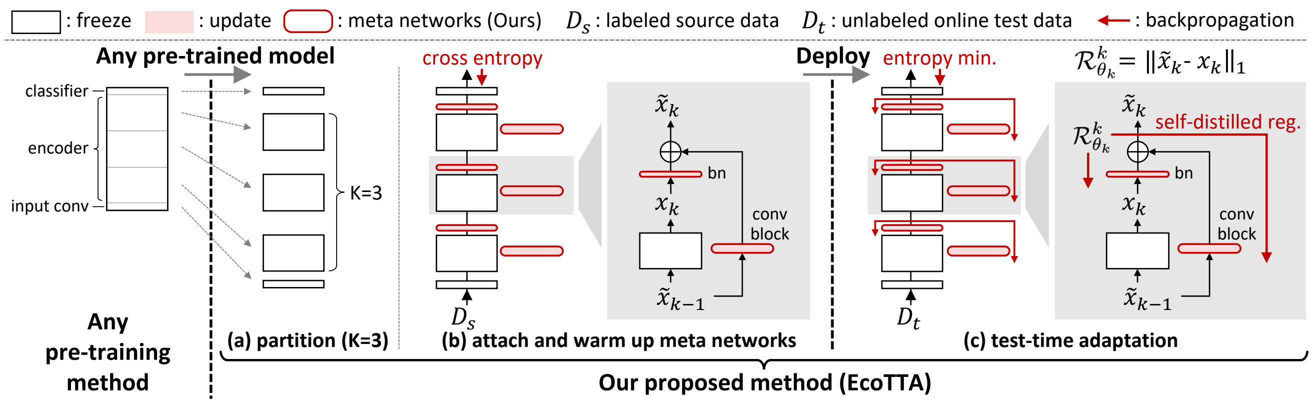


Figure 3. **Overview of our approach. (a)** The encoder of the pre-trained model is divided into K parts (*i.e.,* model partition factor K). **(b)** Before deployment, the meta networks are attached to each part of the original networks and pre-trained with source dataset $\mathcal{D}_s$. **(c)** After the model is deployed, *only* the meta networks are updated with unsupervised loss (*i.e.*, entropy minimization) on target data $\mathcal{D}_t$, while the original networks are frozen. To avoid error accumulation and catastrophic forgetting by the long-term adaptation, we regularize the output $\tilde{x}_k$ of each group of the meta networks leveraging the output x_k of the *frozen* original network, which preserves the source knowledge.

epochs (e.g., 10 epochs for CIFAR dataset) while freezing the original networks. Such a warm-up process is completed before the model deployment, similarly done in several TTA works [9, 28, 33, 41]. Note that we do not require source dataset $\mathcal{D}_s$ during test time.

Pre-trained model partition. Previous TTA studies addressing domain shifts [9, 39] indicate that updating shallow layers is more crucial for improving the adaptation performance than updating the deep layers. Inspired by such a finding, given that the encoder of the pre-trained model is split into model partition factor K (*e.g.*, 4 or 5), we partition the shallow parts of the encoder more (*i.e.*, densely) compared to the deep parts of it. Table 4c shows how performance changes as we vary the model partition factor K.

After deployment. During the test-time adaptation, we only adapt meta networks to target domains while freezing the original networks. Following EATA [41], we use the entropy minimization $H(\hat{y}) = -\sum_c p(\hat{y}) \log p(\hat{y})$ to the samples achieving entropy less than the pre-defined entropy threshold H_0, where $\hat{y}$ is the prediction output of a test image from test dataset $\mathcal{D}_t$ and p$(\cdot)$ is the softmax function. Thus, the main task loss for adaptation is defined as

$$\mathcal{L}^{ent} = \mathbb{I}_{\{H(\hat{y})<H_0\}} \cdot H(\hat{y}), \tag{2}$$

where $\mathbb{I}_{\{\cdot\}}$ is an indicator function. In addition, in order to prevent catastrophic forgetting and error accumulation, we apply our proposed regularization loss $\mathcal{R}^k$, which is described next in detail. Consequently, the overall loss of our method is formulated as,

$$\mathcal{L}^{total}_\theta = \mathcal{L}^{ent}_\theta + \lambda \sum_k^K \mathcal{R}^k_{\theta_k}, \tag{3}$$

where θ and θ_k denotes parameters of all meta networks and those of k-th group of meta networks, respectively, and λ is used to balance the scale of the two loss functions. Note that our architecture requires less memory than previous works [54, 53] since we use frozen original networks and discard its intermediate activations. To be more specific, our architecture uses 82% and 60% less memory on average than CoTTA and TENT/EATA.

3.2. Self-distilled Regularization

The unsupervised loss from unlabeled test data $\mathcal{D}_t$ is likely to provide a false signal (*i.e.*, noise) to the model ($\hat{y} \neq y_t$ where y_t is the ground truth test label). Previous works have verified that long-term adaptation with unsupervised loss causes overfitting due to error accumulation [63, 2] and catastrophic forgetting [54, 29]. To prevent the critical issues, we propose a self-distilled regularization utilizing our architecture. As shown in Fig. 3, we regularize the output $\tilde{x}_k$ of each k-th group of the meta networks not to deviate from the output x_k of the k-th part of frozen original networks. Our regularization loss which computes the mean absolute error (*i.e.*, L1 loss) is formulated as follows:

$$\mathcal{R}^k_{\theta_k} = \|\tilde{x}_k - x_k\|_1 \,. \tag{4}$$

Since the original networks are not updated, the output $x_{k,k\sim K}$ extracted from them can be considered as containing the knowledge learned from the *source* domain. Taking advantage of this fact, we let the output of meta networks $\tilde{x}_k$ be regularized with knowledge distilled from the original networks. By preventing the adapted model to not significantly deviate from the original model, we can prevent 1) catastrophic forgetting by maintaining the source domain knowledge and 2) error accumulation by utilizing the class discriminability of the original model. Remarkably, unlike previous works [54, 41], our regularization does not require saving additional original networks, which accompanies ex-

	WideResNet-40 (AugMix)		WideResNet-28		ResNet-50	
Method	**Avg. err** ↓	**Mem. (MB)**	**Avg. err** ↓	**Mem. (MB)**	**Avg. err** ↓	**Mem. (MB)**
Source	36.7	11	43.5	58	48.8	91
BN Stats Adapt [40]	15.4	11	20.9	58	16.6	91
Single do. TENT [53]	12.7	188	19.2	646	15.0	925
Continual TENT	13.3	188	20.0	646	15.2	925
TTT++ [35]	14.6	391	20.3	1405	16.1	1877
SWR&NSP [9]	<u>12.1</u>	400	17.2	1551	15.4	1971
NOTE [16]	13.4	188	20.2	646	-	-
EATA [41]	13.0	188	18.6	646	<u>14.2</u>	925
CoTTA [54]	14.0	409	17.0	1697	14.4	2066
Ours (K=4)	12.2	80 (80, 58%↓)	<u>16.9</u>	404 (76, 38%↓)	14.4	296 (86, 68%↓)
Ours (K=5)	**12.1**	92 (77, 51%↓)	**16.8**	471 (72, 27%↓)	**14.1**	498 (76, 46%↓)

(a) **CIFAR10-C with severity level 5**

	WideResNet-40 (AugMix)		ResNet-50	
Method	**Avg. err** ↓	**Mem. (MB)**	**Avg. err** ↓	**Mem. (MB)**
Source	69.7	11	73.8	91
BN Stats Adapt [40]	41.1	11	44.5	91
Single do. TENT [53]	36.7	188	40.1	926
Continual TENT	38.3	188	45.9	926
TTT++ [35]	41.0	391	44.2	1876
SWR&NSP [9]	36.6	400	44.1	1970
NOTE [16]	42.8	188	-	-
EATA [41]	37.1	188	39.9	926
CoTTA [54]	38.1	409	40.2	2064
Ours (K=4)	<u>36.4</u>	80 (80, 58%↓)	<u>39.5</u>	296 (86, 68%↓)
Ours (K=5)	**36.3**	92 (77, 51%↓)	**39.3**	498 (76, 46%↓)

(b) **CIFAR100-C with severity level 5**

Table 1. **Comparison of error rate (%) on CIFAR-C.** We report an average error of 15 corruptions on *continual* TTA and a memory requirement including model parameters and activation sizes. The lowest error is in bold, and the second lowest error is underlined. The memory reduction rates compared to CoTTA and TENT are presented sequentially. WideResNet-40 was pre-trained with AugMix [23] that is a data processing to increase the robustness of the model. Source denotes the pre-trained model without adaptation. Single domain (in short, single do.) TENT resets the model when adapting to a new target domain, so the domain labeles are required.

	ResNet-50 (AugMix)	ResNet-50	(MB)
Method	**Avg. err** ↓	**Avg. err** ↓	**Total Mem.** ↓
Source	74.36	82.35	91
BN Stats Adapt [40]	57.87	72.18	91
Continual TENT [53]	56.1 (0.6)	66.2 (1.1)	1486
EATA [41]	54.9 (2.3)	63.8 (2.7)	1486
CoTTA [54]	<u>54.6</u> (3.9)	**62.6** (3.1)	3132
Ours (K=4)	55.2 (3.0)	64.6 (3.2)	**438** (86, 72%↓)
Ours (K=5)	**54.4** (2.7)	<u>63.4</u> (3.0)	<u>747</u> (75, 51%↓)

Table 2. **Comparison of error rate (%) on ImageNet-C with severity level 5.** Standard deviation for ten diverse corruption sequences is denoted by the parentheses values. The total memory refers to the sum of model parameters and activations.

Avg. err (%)		CIFAR10-C		CIFAR100-C	
Method	**Mem. (MB)**	single do.	continual	single do.	continual
BN Stats Adapt [40]	91	16.6	16.6	44.5	44.5
TinyTL† [4]	379	15.8	21.9	<u>40.5</u>	77.4
RepNet† [57]	508	<u>15.2</u>	20.9	41.5	52.1
AuxAdapt† [61]	**207**	16.0	<u>16.7</u>	44.0	<u>45.8</u>
Ours (K=4)	<u>296</u>	**14.4**	**14.4**	**39.5**	**39.2**

Table 3. **Comparison with methods for on-device learning.** The backbone is ResNet-50. † denotes our own re-implemented models. single do. indicates the singe domain TTA setup.

tra memory usage. Moreover, it only needs a negligible amount of computational overhead because it is performed in parallel with the entropy minimization loss $\mathcal{L}^{ent}$.

4. Classification Experiments

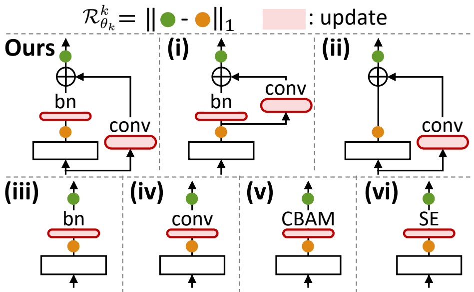
We evaluate our approach to image classification tasks based on the continual test-time adaptation setup with three datasets: CIFAR10-C, CIFAR100-C, and ImageNet-C.

Experimental setup. Following CoTTA [54], we conduct most experiments on the continual TTA task, where we continually adapt the deployed model to each corruption type sequentially without resetting the model. This task is more challenging but more realistic than single domain TTA task [53] in which the adapted model is periodically reset to the original pre-trained model after finishing adaptation to each target, so they require additional domain information. Moreover, we evaluate our approach on the long-term TTA setup, which is detailed in Section 4.2.

Following the previous TTA studies [53, 54], we evaluate models with {CIFAR10, CIFAR10-C}, {CIFAR100, CIFAR100-C}, and {ImageNet, ImageNet-C} where the first and the second dataset in each bracket refers to the source and the target domain, respectively. The target domains include 15 types of corruptions (*e.g.* noise, blur, weather, and digital) with 5 levels of severity, which are widely used in conventional benchmarks [22].

Implementation Details. We evaluate our approach within the frameworks officially provided by previous state-of-the-art methods [54, 41]. For fair comparisons, we use the same pre-trained model, which are WideResNet-28 and WideResNet-40 [59] models from the RobustBench [11], and ResNet-50 [21] model from TTT++ [35, 9]. Before the deployment, we pre-train the meta networks on the source dataset using a cross-entropy loss with SGD optimizer with the learning rate of 5e-2. Since the meta networks contain only a few layers, we pre-train them with a small number of epochs: 10 and 3 epochs for CIFAR and ImageNet, respectively. After deployment, similar to EATA [41], we use the same SGD optimizer with the learning rate of 5e-3. In Eq. (2), the entropy threshold H_0 is set to $0.4 \times \ln C$ where C denotes the number of task classes. The batch size is 64 and 32 for CIFAR and ImageNet, respectively. We set the importance of the regularization λ in Eq. (3) to 0.5 to balance it with the entropy minimization loss. Additional implementation details can be found in Appendix C.

Evaluation Metric. For all the experiments, we report error rates calculated during testing and the memory consump-



(a) **Visualization of networks variants**

Avr. err	CIFAR10-C	CIFAR10-C	CIFAR100-C
Arch	**WRN-28**	**WRN-40**	**WRN-40**
(i)	<u>18.1</u>	<u>12.6</u>	<u>37.2</u>
(ii) Ours w\o BN	18.7	13.7	38.2
(iii) Ours w\o Conv	20.7	14.9	40.1
(iv) Conv	60.6	73.3	77.2
(v) CBAM [55]	21.4	15.1	40.9
(vi) SE [24]	22.3	16.2	40.5
Ours	**16.8**	**12.1**	**36.3**

(b) **Meta network design (K=5)**

Model	#Block	Avg. err
WRN-28 (12) CIFAR10-C	3,3,3,3	<u>17.3</u>
	4,4,2,2	17.9
	2,2,4,4	**16.9**
WRN-40 (18) CIFAR10-C	4,4,5,5	<u>12.8</u>
	6,6,3,3	13.7
	3,3,6,6	**12.2**
WRN-40 (18) CIFAR100-C	4,4,5,5	<u>36.9</u>
	6,6,3,3	38.5
	3,3,6,6	**36.4**

(c) **# of blocks of each partition (K=4)**

Table 4. **Architecture ablation experiments. (a,b)** We compare continual TTA performance on several memory-efficient designs. WRN refers to WideResNet [59] backbone. **(c)** We report the performance based on different designs of partitioning the model. The value next to the backbone's name denotes the total number of residual blocks of a model.

tion, including the model parameter and the activation storage. We demonstrate the memory efficiency of our work by using the official code provided by TinyTL [4].

4.1. Comparisons

Comparisons with TTA methods. We compare our approach to competing TTA methods on extensive benchmarks and various pre-trained models. The results of CIFAR10/100-C are detailed in Table 1. The model partition factor K are set to 4 and 5. Our approach outperforms existing TTA methods with the lowest memory usage in all pre-trained models. Specifically, in WideResNet-40, our method achieves superior performance while requiring 80% and 58% less memory than CoTTA [54] and EATA [41], respectively, which are also designed for continual TTA. Approaches targeting single domain TTA [53, 35, 9] show poor performance due to error accumulation and catastrophic forgetting, as observed in CoTTA. The error rates for each corruption type are provided in Appendix F.

Table 2 shows the experiment for ImageNet-C. Two ResNet-50 backbones from RobustBench [11] are leveraged. Following CoTTA, evaluations are conducted on ten diverse corruption-type sequences. We achieve comparable performance to CoTTA while utilizing 86% and 75% less memory with K=4 and 5, respectively. In addition, we observe that our approach shows superior performance when adopting the model pre-trained with strong data augmentation methods (*e.g.,* Augmix [23]).

Comparisons with on-device learning methods. We compare our approach with methods for memory-efficient on-device learning. TinyTL [4] and RepNet [57] focus on *supervised* on-device learning (*i.e.*, requiring labeled target data). However, since TTA assumes that we do not have access to the target labels, utilizing such methods to TTA directly is infeasible. Therefore, we experimented by replacing supervised loss (*i.e.*, cross-entropy) with unsupervised loss (*i.e.*, entropy minimization) in TinyTL and RepNet. As shown in Table 3, they suffer from performance degradation in continual TTA, showing inferior performance compared

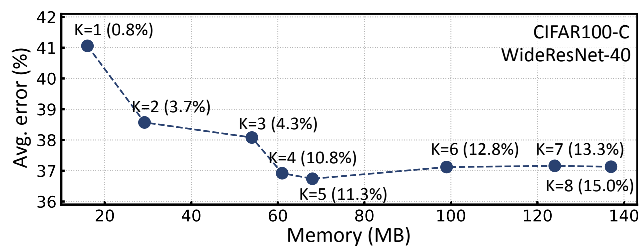


Figure 4. **Ablation study of K.** We uniformly divide the encoder of the pre-trained model into the model partition factor K. The x-axis indicates the memory size including both model parameter size and activation size while the y-axis indicates the average error rate. The values in parentheses show the rate of increase for the model parameters compared to the original model.

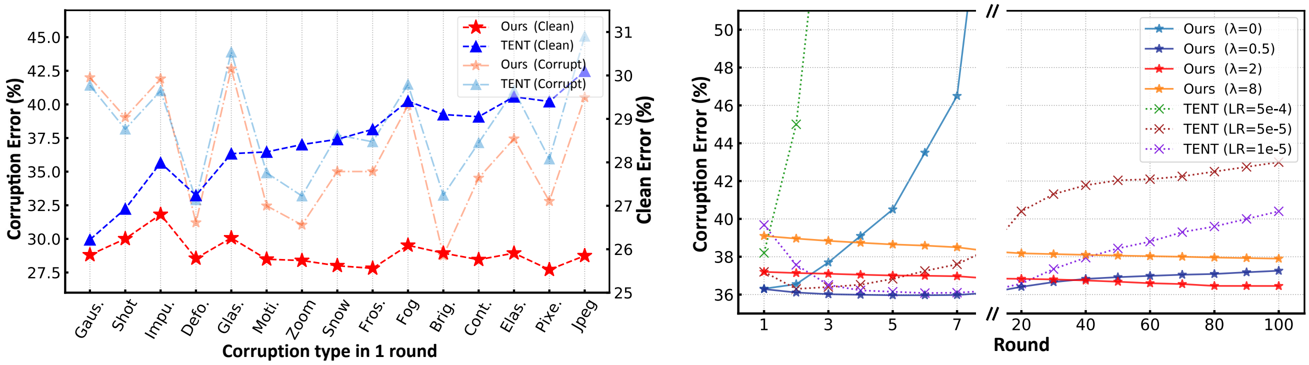
to our proposed approach even in the single domain TTA.

Similar to ours, AuxAdapt [61] adds and updates a small network (*i.e.*, ResNet-18) while freezing the pre-trained model. Unlike our approach, they only modify a prediction output, not intermediate features. While AuxAdapt requires the least memory usage, it fails to improve TTA performance in single domain TTA. Nevertheless, since the original model is frozen, it suffers less from catastrophic forgetting and error accumulation than TinyTL [4] and RepNet [57] in the continual TTA. Through the results, we confirm that our proposed method brings both memory efficiency and a significant performance improvement in both TTA setups.

4.2. Empirical Study

Architecture design. An important design of our meta networks is injecting a single BN layer before the original networks and utilizing a residual connection with one conv block. Table 4b studies the effectiveness of the proposed design by comparing it with six different variants. From the results, we observe that using only either conv block (ii) or BN (iii) aggravates the performance: error rate increases by 1.4% and 3.8% on CIFAR-100-C with WideResNet-40.

In design (i), we enforce both BN parameters and Conv layers in the meta networks to take the output of the original networks as inputs. Such a design brings performance



(a) **Catastrophic forgetting effect** (b) **Error accumulation effect**

Figure 5. **Regularization ablation experiments**. We conduct experiments with WideResNet-40 on CIFAR100-C. **(a)** We utilize a test set of the CIFAR-100 dataset to measure clean error after adapting to each corruption. Maintaining clean errors at a stable level indicates that our approach helps the model robust to catastrophic forgetting. **(b)** We simulate a long-term adaptation scenario by repeating 100 rounds of 15 corruption sequences. In the absence of regularization, error accumulation can lead to overfitting (*i.e.*, the case of the error increases exponentially). However, our approach does not suffer from such an error accumulation. We set K to 5 in the above experiments.

	Batch size	16	8	4	2	1
Non training	Source	69.7	69.7	69.7	69.7	69.7
	BN Stats Adapt [40]	41.1	50.2	59.9	81.0	99.1
	AdaptBN [46]	39.1	41.2	45.2	49.0	54.0
Training	Con. TENT [53]	40.9	47.8	58.6	82.2	99.0
	Con. TENT+AdaptBN	38.2	40.2	43.2	47.7	52.2
	Ours (K=5)	40.0	45.8	63.4	80.8	99.0
	Ours (K=5)+AdaptBN	**36.9**	**39.3**	**42.2**	**46.5**	**51.8**

Table 5. **Experiments with small batch sizes.** We evaluate all baselines with WideResNet-40 on CIFAR100-C. Con. TENT is the abbreviation for continual TENT.

drop. We speculate that it is because the original network, which is not adapted to the target domain, lacks the ability to extract sufficiently meaningful features from the target image. Also, we observed a significant performance degradation after removing the residual connection in design (iv). In addition, since attention mechanisms [55, 24] generally have improved classification accuracy, we study how attention mechanisms can further boost TTA performance of our approach in design (v, vi). The results show that it is difficult for the attention module to train ideally in TTA setup using unsupervised learning, unlike when applying it to supervised learning. An ablation study on each element of meta networks can be found in Appendix D.

Number of blocks in each partition. ResNet [21] consists of multiple residual blocks (*e.g.*, BasicBlock and Bottleneck in Pytorch [45]). For instance, WideResNet-28 has 12 residual blocks. By varying the number of blocks for each part of the original networks, we analyze TTA performance in Table 4c. We observe that splitting the shallow parts of the encoder densely (*e.g.*, 2,2,4,4 blocks, from the shallow to the deep parts sequentially) brings more performance gain than splitting the deep layers densely (*e.g.*, 4,4,2,2 blocks). We suggest that it is because we modify the lower-level feature more as we split shallow layers densely. Our observation is aligned with the finding of previous TTA works [9, 39], which show that updating the shallow layers more than the deep layers improves TTA performance.

Number of model partition K. Fig. 4 shows both memory requirement and adaptation performance according to the model partition factor K. With a small K (*e.g.*, 1 or 2), the intermediate outputs are barely modified, making it difficult to achieve a reasonable level of performance. We achieve the best TTA performance with K of 4 or 5 as adjusting a greater numver of intermediate features. In the meanwhile, we observe that the average error rate is saturated and remains consistent when K is set to large values (*e.g.* 6,7 or 8) even with the increased amount of activations and learnable parameters. Therefore, we set K to 4 and 5.

Catastrophic forgetting. We conduct experiments to confirm the catastrophic forgetting effect (Fig. 5a). Once finishing adaptation to each corruption, we evaluate the model on clean target data (*i.e.*, test-set of CIFAR dataset) without updating the model. For TENT with no regularization, the error rates for the clean target data (*i.e.*, clean error (%)) increase gradually, which can be seen as the phenomenon of catastrophic forgetting. In contrast, our approach consistently maintains the error rates for the clean target data, proving that our regularization loss effectively prevents catastrophic forgetting. These results indicate that our method can be reliably utilized in various domains, including the source and target domains.

Error accumulation in long-term adaptation. To evaluate the error accumulation effect, we repeat all the corruption sequences for 100 rounds. The results are described in Fig. 5b. For TENT, a gradual increase in error rates is observed in later rounds, even with small learning rates. For example, TENT [53] with the learning rate of 1e-5 achieves the error rate of 39.7%, and reached its lowest error rate of 36.5% after 8 rounds. However, it shows increased error rate of 38.6% after 100 rounds due to overfitting. It suggests

Time		t ⟶																
Round		1				4				7				10				All
Method	**Mem. (MB)**	**Brig.**	**Fog**	**Fros.**	**Snow**	**Brig.**	**Fog**	**Fros.**	**Snow**	**Brig.**	**Fog**	**Fros.**	**Snow**	**Brig.**	**Fog**	**Fros.**	**Snow**	**Mean**
Source	280	60.4	54.3	30.0	4.1	60.4	54.3	30.0	4.1	60.4	54.3	30.0	4.1	60.4	54.3	30.0	4.1	37.2
BN Stats Adapt [40]	280	69.1	61.0	44.8	39.1	69.1	61.0	44.8	39.1	69.1	61.0	44.8	39.1	69.1	61.0	44.8	39.1	53.6
Continual TENT [53]	2721	70.1	62.1	46.1	40.2	62.2	53.7	44.4	37.9	50.0	41.5	31.6	26.6	39.2	32.6	25.3	22.4	42.9
Ours (K=4)	**918** (66%↓)	**70.2**	**62.4**	**46.3**	**41.9**	**70.0**	**62.8**	**46.5**	**42.2**	**70.0**	**62.8**	**46.5**	**42.1**	**70.1**	**62.8**	**46.6**	**42.2**	**55.3**

Table 6. **Semantic segmentation results in continual test-time adaptation tasks.** We conduct experiments on Cityscapes [10] with four weather corruptions [22] applied. The four conditions are repeated ten times to simulate continual domain shifts. All results are evaluated based on DeepLabV3Plus-ResNet-50. Among above methods, only single domain TENT requires domain labels.

that without regularization, TTA methods eventually face overfitting in long-term adaptation [63, 2, 29]. Our method in the absence of regularization ($\lambda = 0$) also causes overfitting. On the other hand, when self-distilled regularization is involved ($\lambda > 0$), the performance remains consistent even in the long-term adaptation.

Small batch size. We examine the scalability of our approach with a TTA method designed for small batches size, named adapting BN statistics (*i.e.*, AdaptBN [46, 60]). When the number of batches is too small, the estimated statistics can be unreliable [46]. Thus, they calibrate the source and target statistics for the normalization of BN layers so as to alleviate the domain shift and preserve the discriminative structures. As shown in Table 5, training models with small batch sizes (e.g., 2 or 1) generally increase the error rates. However, such an issue can be addressed by appying AdaptBN to our method. To be more sepcific, we achieve an absolute improvement of 17.9% and 2.2% from Source and AdaptBN, respectively, in the batch size of 1.

Number of the source samples for meta networks. Like previous TTA works [9, 35, 28, 33] including EATA [41], our approach requires access to the source data for pretraining our proposed meta networks before model deployment. In order to cope with the situation where we can only make use of a subset of the source dataset, we study the TTA performance of our method according to the number of accessible source samples. The results are specified in Table 7 where we use WideResNet-40. We observe that our method outperforms the baseline model even with small number of training samples (*e.g.*, 10% or 20%) while showing comparable performance with excessively small numbers (*e.g.* 5%). Note that we still reduce the memory usage of about 51% compared to EATA.

Target domain	**EATA [41]** (188MB)	**Ours (K=5)** (92MB)	**# of source samples** 10k (20%)	5k (10%)	2.5k (5%)
CIFAR10-C	13.0	**12.1**	12.4	12.9	13.1
CIFAR100-C	37.1	**36.3**	36.4	36.6	37.2

Table 7. **Ablation of # of source samples to warm up the meta networks.** Before deployment, we pre-trained the meta networks using only a subset of the source dataset (*e.g.*, 20%, 10%, and 5%). The memory usage (MB) of each method is also presented.

5. Segmentation Experiments

We investigate our approach in semantic segmentation. First, we create Cityscapes-C by applying the weather corruptions (brightness, fog, frost, and snow [22]) to the validation set of Cityscapes [10]. Then, to simulate continual distribution shifts, we repeat the four types of Cityscapes-C ten times. In this scenario, we conduct continual TTA using the publicly-available ResNet-50-based DeepLabV3+ [7], which is pre-trained on Cityscapes for domain generalization task [8] in semantic segmentation. For TTA, we use the batch size of 2. More details are specified in Appendix C.

Results. We report the results based on mean intersection over union (mIoU) in Table 6. It demonstrates that our approach helps to both minimize memory consumption and performs long-term adaptation stably for semantic segmentation. Unlike continual TENT, our method avoids catastrophic forgetting and error accumulation, allowing us to achieve the highest mIoU score while using 66% less memory usage in a continual TTA setup. Additional experiment results can be found in Appendix B.

6. Conclusion

This paper proposed a simple yet effective approach that improves continual TTA performance and saves a significant amount of memory, which can be applied to edge devices with limited memory. First, we presented a memory-efficient architecture that consists of original networks and meta networks. This architecture requires much less memory size than the previous TTA methods by decreasing the intermediate activations used for gradient calculations. Second, in order to preserve the source knowledge and prevent error accumulation during long-term adaptation with noisy unsupervised loss, we proposed self-distilled regularization that controls the output of meta networks not to deviate significantly from the output of the original networks. With extensive experiments on diverse datasets and backbone networks, we verified the memory efficiency and TTA performance of our approach. In this regard, we hope that our efforts will facilitate a variety of studies that make test-time adaptation for edge devices feasible in practice.

Acknowledgments. We would like to thank Kyuwoong Hwang, Simyung Chang, and Byeonggeun Kim for their valuable feedback. We are also grateful for the helpful discussions from Qualcomm AI Research teams.

References

[1] Kazuki Adachi, Shin'ya Yamaguchi, and Atsutoshi Kumagai. Covariance-aware feature alignment with pre-computed source statistics for test-time adaptation. *arXiv preprint arXiv:2204.13263*, 2022. 2, 3

[2] Eric Arazo, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. Pseudo-labeling and confirmation bias in deep semi-supervised learning. In *IJCNN*, 2020. 2, 3, 4, 8

[3] Kambiz Azarian, Debasmit Das, Hyojin Park, and Fatih Porikli. Test-time adaptation vs. training-time generalization: A case study in human instance segmentation using keypoints estimation. In *WACV Workshops*, 2023. 3

[4] Han Cai, Chuang Gan, Ligeng Zhu, and Song Han. Tinytl: Reduce memory, not parameters for efficient on-device learning. In *NeurIPS*, 2020. 2, 3, 5, 6

[5] Dian Chen, Dequan Wang, Trevor Darrell, and Sayna Ebrahimi. Contrastive test-time adaptation. In *CVPR*, 2022. 3

[6] Lin Chen, Huaian Chen, Zhixiang Wei, Xin Jin, Xiao Tan, Yi Jin, and Enhong Chen. Reusing the task-specific classifier as a discriminator: Discriminator-free adversarial domain adaptation. In *CVPR*, 2022. 3

[7] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018. 8

[8] Sungha Choi, Sanghun Jung, Huiwon Yun, Joanne T Kim, Seungryong Kim, and Jaegul Choo. Robustnet: Improving domain generalization in urban-scene segmentation via instance selective whitening. In *CVPR*, 2021. 1, 2, 3, 8

[9] Sungha Choi, Seunghan Yang, Seokeon Choi, and Sungrack Yun. Improving test-time adaptation via shift-agnostic weight regularization and nearest source prototypes. In *ECCV*, 2022. 1, 2, 3, 4, 5, 6, 7, 8

[10] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 8

[11] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. In *NeurIPS Datasets and Benchmarks Track*, 2021. 5, 6

[12] Debasmit Das, Shubhankar Borse, Hyojin Park, Kambiz Azarian, Hong Cai, Risheek Garrepalli, and Fatih Porikli. Transadapt: A transformative framework for online test time adaptive semantic segmentation. In *ICASSP*, 2023. 3

[13] Sauptik Dhar, Junyao Guo, Jiayi Liu, Samarth Tripathi, Unmesh Kurup, and Mohak Shah. A survey of on-device machine learning: An algorithms and learning theory perspective. *ACM Transactions on Internet of Things*, 2021. 2

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 1

[15] Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei A Efros. Test-time training with masked autoencoders. In *NeurIPS*, 2022. 1, 3

[16] Taesik Gong, Jongheon Jeong, Taewon Kim, Yewon Kim, Jinwoo Shin, and Sung-Ju Lee. Robust continual test-time adaptation: Instance-aware bn and prediction-balanced memory. In *NeurIPS*, 2023. 2, 5

[17] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *NeurIPS*, 2004. 2

[18] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In *ICLR*, 2021. 3

[19] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022. 1

[20] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 1

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 5, 7

[22] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*, 2019. 5, 8

[23] Dan Hendrycks, Norman Mu, Ekin D. Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. AugMix: A simple data processing method to improve robustness and uncertainty. In *ICLR*, 2020. 5, 6

[24] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018. 6, 7

[25] Xuefeng Hu, Gokhan Uzunbas, Sirius Chen, Rui Wang, Ashish Shah, Ram Nevatia, and Ser-Nam Lim. Mixnorm: Test-time adaptation through online normalization estimation. *arXiv preprint arXiv:2110.11478*, 2021. 3

[26] Yusuke Iwasawa and Yutaka Matsuo. Test-time classifier adjustment module for model-agnostic domain generalization. In *NeurIPS*, 2021. 1

[27] Vidit Jain and Erik Learned-Miller. Online domain adaptation of a pre-trained cascade of classifiers. In *CVPR*, 2011. 3

[28] Sanghun Jung, Jungsoo Lee, Nanhee Kim, and Jaegul Choo. Cafa: Class-aware feature alignment for test-time adaptation. *arXiv preprint arXiv:2206.00205*, 2022. 2, 4, 8

[29] Tommie Kerssies, Joaquin Vanschoren, and Mert Kılıçkaya. Evaluating continual test-time adaptation for contextual and semantic domain shifts. *arXiv preprint arXiv:2208.08767*, 2022. 3, 4, 8

[30] Ansh Khurana, Sujoy Paul, Piyush Rai, Soma Biswas, and Gaurav Aggarwal. Sita: Single image test-time adaptation. *arXiv preprint arXiv:2112.02355*, 2021. 3

[31] Daiqing Li, Junlin Yang, Karsten Kreis, Antonio Torralba, and Sanja Fidler. Semantic segmentation with generative models: Semi-supervised learning and strong out-of-domain generalization. In *CVPR*, 2021. 1

[32] Xiaotong Li, Yongxing Dai, Yixiao Ge, Jun Liu, Ying Shan, and Ling-Yu Duan. Uncertainty modeling for out-of-distribution generalization. In *ICLR*, 2022. 3

[33] Hyesu Lim, Byeonggeun Kim, Jaegul Choo, and Sungha Choi. TTN: A domain-shift aware batch normalization in test-time adaptation. In *ICLR*, 2023. 2, 3, 4, 8

[34] Ji Lin, Wei-Ming Chen, Yujun Lin, Chuang Gan, Song Han, et al. Mcunet: Tiny deep learning on iot devices. In *NeurIPS*, 2020. 3

[35] Yuejiang Liu, Parth Kothari, Bastien van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. Ttt++: When does self-supervised test-time training fail or thrive? In *NeurIPS*, 2021. 1, 2, 3, 5, 6, 8

[36] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Unsupervised domain adaptation with residual transfer networks. In *NeurIPS*, 2016. 1

[37] Robert A Marsden, Mario Döbler, and Bin Yang. Gradual test-time adaptation by self-training and style transfer. *arXiv preprint arXiv:2208.07736*, 2022. 3

[38] Ke Mei, Chuang Zhu, Jiaqi Zou, and Shanghang Zhang. Instance adaptive self-training for unsupervised domain adaptation. In *ECCV*, 2020. 3

[39] Chaithanya Kumar Mummadi, Robin Hutmacher, Kilian Rambach, Evgeny Levinkov, Thomas Brox, and Jan Hendrik Metzen. Test-time adaptation to distribution shift by confidence maximization and input transformation. *arXiv preprint arXiv:2106.14999*, 2021. 3, 4, 7

[40] Zachary Nado, Shreyas Padhy, D Sculley, Alexander D'Amour, Balaji Lakshminarayanan, and Jasper Snoek. Evaluating prediction-time batch normalization for robustness under covariate shift. *arXiv preprint arXiv:2006.10963*, 2020. 5, 7, 8

[41] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *ICML*, 2022. 2, 3, 4, 5, 6, 8

[42] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards stable test-time adaptation in dynamic wild world. In *ICLR*, 2023. 3

[43] Xingang Pan, Ping Luo, Jianping Shi, and Xiaoou Tang. Two at once: Enhancing learning and generalization capacities via ibn-net. In *ECCV*, 2018. 3

[44] Kwanyong Park, Sanghyun Woo, Inkyu Shin, and In So Kweon. Discover, hallucinate, and adapt: Open compound domain adaptation for semantic segmentation. In *NeurIPS*, 2020. 3

[45] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, and et al. Lin. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 7

[46] Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. In *NeurIPS*, 2020. 7, 8

[47] Inkyu Shin, Dong-Jin Kim, Jae Won Cho, Sanghyun Woo, Kwanyong Park, and In So Kweon. Labor: Labeling only if required for domain adaptive semantic segmentation. In *ICCV*, 2021. 3

[48] Inkyu Shin, Yi-Hsuan Tsai, Bingbing Zhuang, Samuel Schulter, Buyu Liu, Sparsh Garg, In So Kweon, and Kuk-Jin Yoon. Mm-tta: Multi-modal test-time adaptation for 3d semantic segmentation. In *CVPR*, 2022. 2, 3

[49] Inkyu Shin, Sanghyun Woo, Fei Pan, and InSo Kweon. Two-phase pseudo label densification for self-training based domain adaptation. In *ECCV*, 2020. 3

[50] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *ICML*, 2020. 3

[51] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NeurIPS*, 2017. 3

[52] Tuan-Hung Vu, Himalaya Jain, Maxime Bucher, Mathieu Cord, and Patrick Pérez. Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation. In *CVPR*, 2019. 3

[53] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *ICLR*, 2021. 1, 2, 3, 4, 5, 6, 7, 8

[54] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *CVPR*, 2022. 1, 2, 3, 4, 5, 6

[55] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *ECCV*, 2018. 6, 7

[56] Li Yang, Adnan Siraj Rakin, and Deliang Fan. Da3: Dynamic additive attention adaption for memory-efficient on-device multi-domain learning. In *CVPR Workshops*, 2022. 3

[57] Li Yang, Adnan Siraj Rakin, and Deliang Fan. Rep-net: Efficient on-device learning via feature reprogramming. In *CVPR*, 2022. 2, 3, 5, 6

[58] Fuming You, Jingjing Li, and Zhou Zhao. Test-time batch statistics calibration for covariate shift. *arXiv preprint arXiv:2110.04065*, 2021. 3

[59] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016. 5, 6

[60] Marvin Zhang, Sergey Levine, and Chelsea Finn. Memo: Test time robustness via adaptation and augmentation. In *NeurIPS*, 2021. 3, 8

[61] Yizhe Zhang, Shubhankar Borse, Hong Cai, and Fatih Porikli. Auxadapt: Stable and efficient test-time adaptation for temporally consistent video semantic segmentation. In *WACV*, 2022. 5, 6

[62] Yabin Zhang, Minghan Li, Ruihuang Li, Kui Jia, and Lei Zhang. Exact feature distribution matching for arbitrary style transfer and domain generalization. In *CVPR*, 2022. 3

[63] Zixing Zhang, Fabien Ringeval, Bin Dong, Eduardo Coutinho, Erik Marchi, and Björn Schüller. Enhanced semi-supervised learning for multimodal emotion recognition. In *ICASSP*, 2016. 2, 3, 4, 8

[64] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain generalization with mixstyle. In *ICLR*, 2021. 3