STAMP: Outlier-Aware Test-Time Adaptation with Stable Memory Replay

Yongcan Yu[1,2], Lijun Sheng[1,3], Ran He[1,2], and Jian Liang[1,2⋆]

[1] NLPR & MAIS, Institute of Automation, Chinese Academy of Sciences
[2] School of Artificial Intelligence, University of Chinese Academy of Sciences
[3] University of Science and Technology of China
{yuyongcan0223, liangjian92}@gmail.com

Abstract. Test-time adaptation (TTA) aims to address the distribution shift between the training and test data with only unlabeled data at test time. Existing TTA methods often focus on improving recognition performance specifically for test data associated with classes in the training set. However, during the open-world inference process, there are inevitably test data instances from unknown classes, commonly referred to as outliers. This paper pays attention to the problem that conducts both sample recognition and outlier rejection during inference while outliers exist. To address this problem, we propose a new approach called STAble Memory rePlay (STAMP), which performs optimization over a stable memory bank instead of the risky mini-batch. In particular, the memory bank is dynamically updated by selecting low-entropy and label-consistent samples in a class-balanced manner. In addition, we develop a self-weighted entropy minimization strategy that assigns higher weight to low-entropy samples. Extensive results demonstrate that STAMP outperforms existing TTA methods in terms of both recognition and outlier detection performance. The code is released at https://github.com/yuyongcan/STAMP.

Keywords: online test-time adaptation · outlier detection · memory bank · self-weighted entropy minimization

1 Introduction

Deep neural networks (DNNs) [46, 48] have achieved remarkable performance across various tasks. Nevertheless, the impressive performance is based on the assumption that training data and test data come from the same distribution. However, in real-world scenarios, the presence of distribution shifts is ubiquitous [13, 30], leading to a degradation in the prediction performance [7]. Taking autonomous driving [60] as an example, changing weather conditions causes systems trained on laboratory data to make incorrect decisions. To mitigate the impact of distribution shift, test-time adaptation (TTA) [29, 58] has emerged as

⋆ Corresponding author.

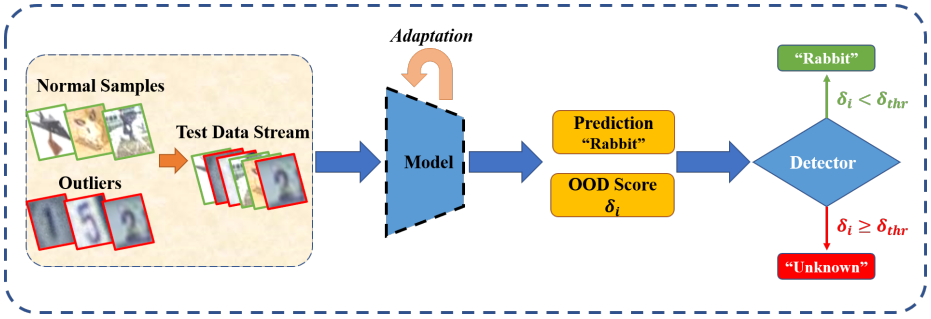


Fig. 1: The illustration of outlier-aware test-time adaptation. The test data stream consists of both normal samples and outliers. When the test data arrives, the algorithm adapts the model and provides two outputs, a prediction and an OOD score. OOD score indicates the likelihood that a sample is an outlier. The detector in the deployment scenario chooses to keep the prediction or reject samples according to the OOD score.

a promising approach, adapting models with unlabeled samples during inference. Nowadays, TTA has achieved remarkable results in various tasks, such as image classification [51, 62], semantic segmentation [3, 49], object detection [1].

Previous TTA works [5, 38, 59] always assume that test data and training data share the same class space. Nevertheless, in realistic open-world scenarios, test data streams will inevitably contain samples from new semantic classes, also known as outliers. The introduction of outliers brings two additional risks to adaptation tasks: (i) Classifying outliers into known categories poses safety risks, especially in contexts like autonomous driving, where systems should request human intervention to handle unknown objects to avoid traffic accidents. (ii) Self-supervised loss (*e.g.*, entropy minimization, pseudo-labeling) calculated with outliers may misguide the optimization process, decreasing the model's recognition ability on known classes.

To better tackle the above risks, we pay attention to a more realistic TTA scenario termed outlier-aware test-time adaptation. Outlier-aware test-time adaptation aims to conduct recognition for normal samples and reject to make predictions for outliers. The overview of outlier-aware test-time adaptation is illustrated in Figure 1. When test data mixed with outliers arrives, the algorithm adapts the model and generates the prediction and an out-of-distribution (OOD) score for each sample. Then we decide to keep the prediction or reject the sample based on its OOD score. The sample with a higher OOD score is more likely to be considered an outlier.

To address the outlier-aware test-time adaptation, we additionally design a novel algorithm called STAble Memory rePlay (STAMP). STAMP consists of three components: reliable class-balanced memory, self-weighted entropy minimization, and stable optimization strategy. Reliable class-balanced memory stores reliable samples for optimization by filtering samples based on their pre-

diction's consistency and entropy. To maintain the class-balanced prediction of the model, it dynamically discards samples with the class that participates most frequently in optimization when the memory is full. Furthermore, we adopt self-weighted entropy minimization to assign greater weights to low-entropy samples in memory and adjust the weights for each sample adaptively. To ensure the stability of the optimization process, we dynamically adjust the optimization step size to quickly adapt to the target domain at the early stage and mitigate error accumulation later. Moreover, we apply the sharpness-aware minimization technique [14] to further reduce the impact of noisy gradients.

To demonstrate the effectiveness of STAMP, we compare it with eight state-of-the-art TTA algorithms. We select three commonly used datasets in TTA, including CIFAR10-C, CIFAR100-C [10, 20], and ImageNet-C [9, 20], alongside some outlier datasets (i.e., Noise, SVHN [37], LSUN [57], TinyImageNet [25], Textures [6], Places365 [64]). Experimental results show that our method achieves superior performance on recognition and outlier detection under the outlier-aware TTA setting. The experiment on conventional protocol indicates that STAMP also performs well in the scenario without outliers. In summary, we list our contributions as follows:

- We introduce outlier-aware test-time adaptation, a TTA scenario considering both normal sample recognition and outlier rejection during inference.
- We propose a memory that dynamically stores reliable samples for replay. To prevent noisy samples from entering the memory, we propose a filtering strategy based on the consistency and entropy of the model's prediction.
- We design self-weighted entropy minimization, which guides the model to adaptively learn to assign higher weights to valuable samples.
- We propose a proper metric that considers both generalization and outlier detection performance. Extensive experiments compared with eight TTA algorithms on several benchmarks demonstrate the effectiveness of our method.

2 Related Work

2.1 Test-Time Adaptation

TTA intends to adapt the model pre-trained on the source domain to the target domain while the distribution shift exists. Compared to domain generalization [47, 52] and domain adaptation [15, 53], TTA needs to access neither the source data nor the target labels, making it has a wide range of applications [11, 12, 45]. TTA methods can be mainly divided into two solutions: batch normalization (BN) calibration and self-training. In BN calibration, Nado et al. [36] utilize the BN statistics of the test batch (μ_t, σ_t) to conduct normalization in the BN layer, while Schneider [43] propose a more comprehensive approach considering statistics from both the source domain (μ_s, σ_s) and the target domain to calibrate the domain shifts. To conquer the correlation sampling problem, Yuan et al. [59] adopt robust batch normalization, which updates the statistic (μ_t, σ_t) in a moving average manner with the test data stream.

Currently, the primary paradigm in TTA is self-training [2, 61], where Wang et al. [51] adopt entropy minimization as a self-supervised loss for optimization and Chen et al. [5] introduce contrastive loss to TTA. However, unsupervised loss may contain noise information. Niu et al. [38] first propose filtering out high-entropy samples, a mechanism adopted by subsequent works [17, 39]. Subsequently, more attention has been paid to addressing more realistic adaptation scenarios such as continually changing target domains [4, 54], label shift [41, 65], and temporally correlated shift [16, 63]. In our study, we focus on a more realistic scenario where test data contains samples with new semantic classes. Some existing works [17, 26, 28] discuss the TTA algorithm in scenarios containing outliers. However, Lee et al. [26] and Gong et al. [17] solely focus on the generalization ability to domain shift, neglecting the issue of rejecting the outliers. A recent work [28] discusses outlier detection performance based on TTA tasks, but it only provides binary predictions to indicate whether the test data is outlier. In contrast, outlier-aware test-time adaptation is more flexible, providing an OOD score, and different systems can conduct sample rejection based on security requirements and the score.

2.2 Out-of-Distribution Detection

OOD detection aims to identify instances that significantly deviate from the distribution of the training data [31]. Their primary focus is usually on detecting samples whose labels were not seen during the training phase. This area has been extensively researched, with one common approach being the utilization of confidence scores such as max softmax probability [21], energy score [33, 34], and others [27] derived from the model's output to detect OOD samples. Further improvements have been made by Liang et al. [32] through techniques like temperature scaling and input preprocessing to enhance the effectiveness of confidence scores. Another avenue for OOD sample detection lies in training-based methods [10, 50, 55]. These techniques involve training a branch using both in-distribution data and OOD data to enable the model to distinguish OOD data during training. However, these methods alter the training process of the source model, which can not be directly applied to adaptation tasks based on a well-trained model. Moreover, OOD detection traditionally assumes that both the training and test data originate from the same domain, a condition not met in TTA scenarios. Under the realistic scenario, there may be a domain shift between test data and training data, which brings greater challenges to detection.

3 Methodology

In this section, we first introduce the outlier-aware test-time adaptation problem in Section 3.1. Then we describe our proposed method, including how to build a memory bank and conduct the optimization in Section 3.2.

3.1 Outlier-Aware Test-Time Adaptation

Let $\mathcal{D}_S$ be the source domain dataset with label space $\mathcal{C}_S$ follows a distribution $P_S(x, y)$. Similarly, let $\mathcal{D}_T = \{(x_i, y_i)\}_{i=1}^{N_T}$ be the target domain dataset with label space $\mathcal{C}_T$ follows a distribution $P_T(x, y)$. Given a source model $f_{\theta_s}(x)$ well trained on $\mathcal{D}_S$ with parameter θ_s. TTA aims to adapt the source model to the target domain while $P_S(x) \neq P_T(x)$.

In traditional closed-set TTA, it is commonly assumed that the $\mathcal{D}_S$ and $\mathcal{D}_T$ share the same label space, i.e., $\mathcal{C}_S = \mathcal{C}_T$. However, in outlier-aware test-time adaptation, we consider $\mathcal{C}_S \subseteq \mathcal{C}_T$ while some outliers with new semantic classes will be encountered in $\mathcal{D}_T$. The illustration of outlier-aware test-time adaption is shown in Figure 1. In addition to adapting the model and providing the prediction, we also generate an OOD score δ_i for each sample x_i. Then we adopt an outlier detector $g(\cdot)$ as follow:

$$g(x_i) = \begin{cases} 1, & \text{if } \delta_i \geq \delta_{thr} \\ 0, & \text{if } \delta_i < \delta_{thr}, \end{cases} \tag{1}$$

where δ_{thr} is a threshold that can be adjusted according to the requirements of different application scenarios. The samples with $g(x_i) = 1$ will be considered outliers and rejected for recognition. And we output the prediction for samples with $g(x_i) = 0$. In our work, we take the entropy of the model prediction as our OOD score.

3.2 Stable Memory Replay

In this section, we introduce stable memory replay (STAMP). STAMP first uses a filtering mechanism to filter out unreliable samples in the coming mini-batch and stores reliable samples in memory. Then, STAMP optimizes the model using in-memory samples by applying self-weighted entropy minimization. In order to further improve the stability of optimization, STAMP uses test-time step size decay and sharpness-aware minimization to guide optimization. The illustration of STAMP is shown in Figure 2.

Reliable Class-Balanced Memory. Since excluding unreliable samples from optimization has shown great effectiveness [17, 38], we propose reliable class-balanced memory to store reliable samples for optimization. To have a more robust estimation of the sample's reliability, we get its prediction in an augmentation-averaged way. Moreover, we prevent unreliable samples from getting into the memory bank $\mathcal{M}$ by employing consistency filtering and entropy filtering. To preserve the model's balanced class prior distribution, we dynamically discard samples with the highest class frequency from the memory bank.

To ensure robust predictions, we adopt the augmentation-averaged method, which addresses the issue of deep neural networks being overconfident with out-of-distribution samples. This overconfidence often blurs the line between in-distribution and out-of-distribution samples. However, research has demonstrated that employing test-time augmentation [44, 61] enhances model calibration [40]. Thus, when presented with a test sample x, we derive its prediction

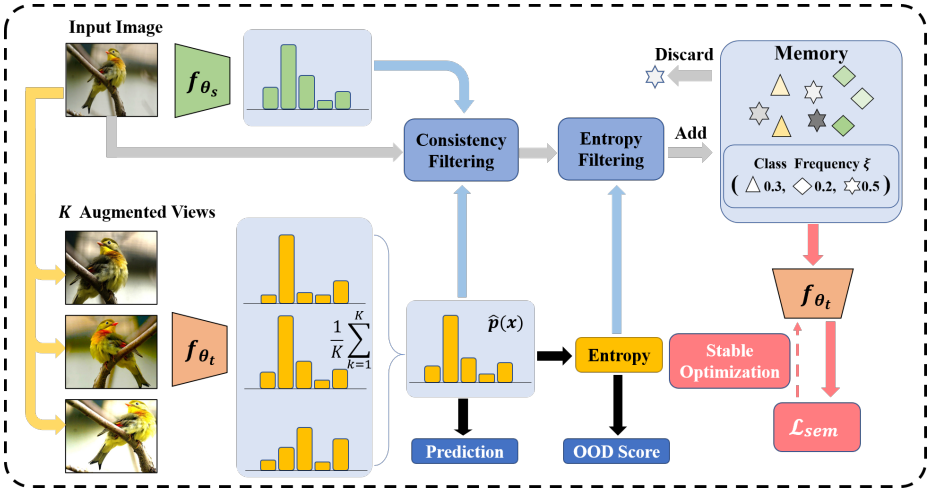


Fig. 2: The overview of STAMP. STAMP retains reliable samples for optimizing the network by maintaining a memory bank. When test samples arrive, STAMP generates an augmentation-averaged prediction. Two filtering mechanisms are designed to filter out inconsistent and high-entropy samples and add the remaining to the memory. A frequency vector is maintained to select samples to discard when the memory is full. STAMP leverages the reliable samples in the memory to optimize the model with the SAM optimizer alongside a decaying step strategy.

through the following process:

$$\hat{p}(x) = \frac{1}{K}\sum_{k=1}^{K} f_{\theta_t}(\tau_k(x)), \tag{2}$$

where $\tau(\cdot)$ is the operation of augmentation, K is the total number of augmented views. By averaging predictions over augmented samples, the model can better eliminate the uncertainty inherent in its predictions, improving calibration and obtaining more reliable confidence estimates. In the supplementary material, we discuss the impact of the augmentation-averaged estimation.

After acquiring the prediction $\hat{p}$, we implement filtering strategies to prevent unreliable samples from being added to memory for two reasons. Firstly, optimization using unreliable samples can result in error accumulation over time [38]. Secondly, we prevent outliers from entering the memory and reduce the entropy of normal samples in the memory. This action facilitates the model's ability to distinguish between normal samples and outliers. Since reliable samples that are beneficial to optimization tend to have the same pseudo-label as the source model, we employ consistency filtering to remove samples whose predictions are inconsistent with those of the source model:

$$m_{con}(x) = \mathbb{I}\Big(\arg\max_c \, [\hat{p}(x)]_c = \arg\max_c \, [f_{\theta_s}(x)]_c\Big), \tag{3}$$

Algorithm 1 Reliable Class-Balanced Memory

Input: memory bank $\mathcal{M}$, a batch of data $\{x_i\}_{i=1}^{B}$, entropy threshold $\mathcal{H}_{thr}$, class frequency $\{\xi_c\}_{c=1}^{|\mathbf{C}_S|}$.
for $i = 1$ **to** B **do**
  Calculate $\hat{p}(x_i)$ by Equation (2) and $\hat{y}_i = \arg\max_c [\hat{p}(x_i)]_c$.
  Calculate $m_{con}(x_i)$, $m_{ent}(x_i)$ of x_i by Equation (3), (4).
  if $m_{con}(x_i) = 1$ and $m_{ent}(x_i) = 1$ **then**
    if $\mathcal{M}$ is full **then**
      Discard one sample with highest class frequency ξ from $\mathcal{M}$.
    end if
    add $(x_i, \hat{y}_i)$ into $\mathcal{M}$.
  end if
end for
Update ξ with Equation (5).

where $\mathbb{I}(\cdot)$ is the indicator function. Moreover, including high entropy samples for optimization can harm the stability of the model [38,39], while samples with low entropy always possess greater value and reliability. Therefore, we use entropy filtering to filter out samples with high entropy from entering the memory.

$$m_{ent}(x) = \mathbb{I}\Big(\mathcal{H}(\hat{p}(x)) < \mathcal{H}_{thr}\Big), \tag{4}$$

where $\mathcal{H}(q) = -\sum_c q_c \log q_c$ denotes the entropy of the prediction vector q. We filter out samples with $\mathcal{H}(\hat{p}) \geq \mathcal{H}_{thr}$, where $\mathcal{H}_{thr}$ is a threshold. Considering both filtering strategies, we add sample $(x, \hat{y})$ into memory bank $\mathcal{M}$ if the sample satisfies consistency filtering and entropy filtering simultaneously, *i.e.*, $m_{con}(x) = 1$ and $m_{ent}(x) = 1$.

Since the number of samples stored in memory is limited, we design a class-balanced strategy to choose a sample to discard when the memory is full. Since we adopt entropy minimization as our optimization objective, an issue arises when the model excessively optimizes samples from a single category over a short period. It leads to a bias towards predicting the dominant category. To counteract this bias, we dynamically discard samples from the memory that belong to the category with the highest update frequency. After each optimization of the model, we update the class frequencies in a moving average manner:

$$\xi_c^t = (1 - \beta)\xi_c^{t-1} + \beta\nu_c^t, \tag{5}$$

where ν_c^t represents the number of samples with class label c in memory and β is moving average coefficient hyperparameter. While the memory is full, we discard a sample from the class with the highest ξ. We summarize the work process of reliable class-balanced memory in Algorithm 1.

Self-Weighted Entropy Minimization. After updating the memory, we replay the reliable samples stored in the memory $\mathcal{M}$ to optimize the model. Following [51], we employ entropy minimization as our loss function. This self-supervised loss encourages the model to exhibit higher confidence in its predic-

Algorithm 2 The Pipeline of STAMP

Initialize: Memory bank $\mathcal{M} = \emptyset$, zero-initialized class frequency $\{\xi_c\}_{c=1}^{|\mathcal{C}_S|}$.
Input: Unlabeled test samples $\mathcal{D}_T = \{x_i\}_{i=1}^{N_T}$, a trained model $f_{\theta_s}(\cdot)$.
for each online mini-batch X_b **do**
  Calculate $\hat{p}(x)$ by Equation (2) and $\mathcal{H}(\hat{p}(x))$ for all $x \in X_b$.
  Obtain the prediction $\hat{y} = \arg\max_c [\hat{p}(x)]_c$ for all $x \in X_b$.
  Update memory bank $\mathcal{M}$ by Algorithm 1.
  Compute $\mathcal{L}_{sem}$ with $\mathcal{M}$ by Equation (6).
  Update θ_t with SAM in Equation (7) and decay step size in Equation (8).
end for
Output: The prediction and OOD score $\{\hat{y}_i, \mathcal{H}(\hat{p}(x_i))\}_{i=1}^{N_T}$.

tions. In order to give more valuable samples a higher importance in optimization, we weight samples in $\mathcal{M}$ in a self-weighted manner. Given samples $\{x_i\}_{i=1}^{|\mathcal{M}|}$, we calculate the following loss function:

$$\mathcal{L}_{sem}(\mathcal{M}, \theta_t) = \sum_{x_i \in \mathcal{M}} \frac{w_i}{\sum_j w_j} \cdot \mathcal{H}(f_{\theta_t}(x_i)) \tag{6}$$

where $w_i = e^{-\mathcal{H}(f_{\theta_t}(x_i))}$ denotes the weight of sample x_i. **It's worth noting that w_i is also involved with the gradient back-propagation.** The reason for this adaptive weighting is two-fold: (i) it gives greater weights to samples with lower entropy, which are considered more reliable. (ii) The model needs to learn the weight of each sample and adaptively give important samples higher weights, instead of solely minimizing entropy for all samples.

Stable Optimization Strategy. To optimize the model in a stable way, we employ sharpness-aware minimization (SAM) [14] and step size decay in parameter updating. SAM is a technique that aims to improve generalization performance in deep learning models by minimizing a loss function while simultaneously controlling the sharpness of the loss landscape. Previous work [14] demonstrates that SAM exhibits resilience against noisy labels, which is also a prevalent challenge in TTA. Therefore, we consider SAM as a good choice of the optimizer in the adaptation scenario [17, 39]. Mathematically, this involves minimizing the following objective function:

$$\min_{\theta_t} \mathcal{L}_{sem}^{SA}(\mathcal{M}, \theta_t), \text{ where } \mathcal{L}_{sem}^{SA}(\mathcal{M}, \theta_t) \triangleq \max_{||\epsilon||_2 \leq \rho} \mathcal{L}_{sem}(\mathcal{M}, \theta_t + \epsilon), \tag{7}$$

where ρ represents the radius of the neighborhood. SAM minimizes the maximum loss within the parameters' vicinity.

Recent research [26, 42] has revealed that under long-term adaptation, TTA algorithms may suffer from substantial error accumulation and consequent performance degradation. This issue becomes even more severe when test data contains outliers. To mitigate this problem, we introduce a test-time step size (*a.k.a.* learning rate) decay strategy. To achieve this, we update the model's parameters

Table 1: Results (%) on **CIFAR10-C** and **CIFAR100-C** datasets under outlier-aware TTA setting across **four outlier datasets**. We report the accuracy (ACC), AUC score, and H-score. All the results are averaged over 15 different corruptions. The results in bold denote the best performance.

		Noise			SVHN-C			LSUN-C			TinyImageNet-C			Avg.		
		ACC	AUC	H-score	ACC	AUC	H-score	ACC	AUC	H-score	ACC	AUC	H-score	ACC	AUC	H-score
CIFAR10-C	Source	57.3	70.4	62.3	57.3	67.4	61.1	57.3	62.8	59.6	57.3	64.5	59.4	57.3	66.3	60.6
	BN Stats [36]	72.9	68.6	70.6	78.7	75.3	76.9	79.4	79.4	79.4	79.0	72.9	75.8	77.5	74.0	75.7
	Tent [51]	77.4	48.7	59.7	80.8	54.9	65.1	81.2	62.3	70.4	81.1	65.6	72.4	80.1	57.9	66.9
	EATA [38]	72.9	68.5	70.6	78.8	75.3	76.9	79.4	79.4	79.4	78.9	73.1	75.9	77.5	74.1	75.7
	SAR [39]	72.9	68.5	70.6	78.7	75.3	76.9	79.4	79.4	79.4	79.0	72.9	75.8	77.5	74.0	75.7
	CoTTA [54]	77.3	62.4	67.3	81.6	78.6	80.1	82.2	84.2	83.2	81.9	**75.3**	**78.4**	80.8	75.1	77.2
	RoTTA [59]	77.6	74.3	75.6	78.4	76.0	77.2	78.8	79.5	79.1	78.6	73.3	75.8	78.3	75.8	76.9
	SoTTA [17]	77.8	51.7	61.6	79.3	72.8	75.9	79.8	77.9	78.8	79.6	72.6	75.9	79.1	68.8	73.1
	OWTTT [28]	62.3	64.4	58.5	66.1	75.3	69.6	63.1	78.9	68.5	56.3	58.8	56.2	62.0	69.3	63.2
	STAMP	**77.9**	**80.7**	**78.8**	**82.3**	**79.0**	**80.5**	**83.5**	**84.9**	**84.2**	**82.6**	75.1	**78.6**	**81.6**	**79.9**	**80.5**
CIFAR100-C	Source	35.8	43.1	38.0	35.8	49.4	40.1	35.8	58.2	43.2	35.8	57.1	42.7	35.8	52.0	41.0
	BN Stats [36]	45.8	80.9	58.4	52.7	72.5	60.9	53.7	73.8	62.0	53.2	68.6	59.7	51.3	74.0	60.3
	Tent [51]	47.9	55.8	51.2	54.4	70.4	61.2	55.4	72.4	62.7	55.0	68.6	60.9	53.2	66.8	59.0
	EATA [38]	55.2	86.1	67.1	58.1	75.6	65.6	58.8	77.2	66.7	58.6	70.7	64.0	57.7	77.4	65.9
	SAR [39]	57.5	88.6	68.9	59.2	65.2	61.9	60.5	73.5	66.3	60.8	72.1	65.9	59.5	74.9	65.8
	CoTTA [54]	47.0	83.4	59.9	53.7	73.2	61.8	54.3	76.9	63.6	54.5	68.1	60.4	52.4	75.4	61.4
	RoTTA [59]	47.9	54.0	49.4	47.3	67.0	55.3	48.3	69.5	56.7	47.8	65.5	55.0	47.8	64.0	54.1
	SoTTA [17]	54.4	53.3	52.8	53.6	70.3	60.7	54.4	70.8	61.4	53.9	68.4	60.1	54.1	65.7	58.8
	OWTTT [28]	47.1	70.3	56.2	53.9	74.3	62.3	54.5	73.5	62.5	54.2	68.5	60.4	52.4	71.6	60.4
	STAMP	**57.9**	**97.5**	**72.6**	**63.7**	**80.0**	**70.9**	**63.7**	**80.7**	**71.2**	**63.9**	**74.4**	**68.7**	**62.3**	**83.1**	**70.8**

θ_t using cosine annealing decay step size:

$$\theta_{t+1} = \theta_t - \alpha_t \nabla \mathcal{L}_{sem}^{SA}(\mathcal{M}, \theta_t), \text{ where } \alpha_t = \frac{1}{2}\alpha_0(1 + cos(\frac{t}{T}\pi)), \tag{8}$$

where α_t is the step size at timestamp t and T is a hyper-parameter to control the step size decay. While step size decay is commonly utilized in model training [56], we leverage it during testing for a distinct purpose. In the early stages of TTA, the model should adapt swiftly due to the significant disparity between the source and target domains. However, as adaptation progresses and the model acquires comprehensive distributional information about the target domain, it should reduce parameter updates to prevent error accumulation and enhance stability. We summarize the pipeline of STAMP in Algorithm 2.

4 Experiments

4.1 Setup

Datasets. Following previous TTA methods [16, 18, 54], we adopt three corruption datasets as normal datasets (*i.e.*, in-distribution datasets). CIFAR10-C and CIFAR100-C [20, 24] stand out as the most widely used benchmarks in TTA tasks, comprising 10/100 categories with 15 types of corruptions respectively. Each type of corruption has 10,000 images. Similarly, ImageNet-C [9, 20] serves as a significant benchmark, consisting of 1,000 categories with the same 15 types

Table 2: Results (%) on **ImageNet-C** under outlier-aware TTA setting across **two outlier datasets**. All the results are averaged over 15 different corruptions.

	Places365-C			Textures-C		
Method	ACC	AUC	H-score	ACC	AUC	H-score
Source	18.2	61.6	26.1	18.2	54.6	25.8
BN Stats [36]	31.1	67.7	41.1	31.6	61.2	40.7
Tent [51]	34.9	51.8	39.5	39	48.6	42.0
EATA [38]	**46.4**	72.6	56.0	46.4	62.2	52.8
SAR [39]	44.9	73.3	55.0	45.6	67.0	54.0
CoTTA [54]	33.8	66.9	43.5	34.2	60.7	42.8
RoTTA [59]	36.6	68.6	46.5	37.0	65.3	46.5
SoTTA [17]	41.7	67.8	50.7	41.8	60.3	48.8
OWTTT [28]	9.1	54.0	13.9	9.4	59.4	14.6
STAMP	**46.4**	**76.2**	**57.2**	**46.5**	**70.0**	**55.6**

of corruption. For out-of-distribution datasets, we choose Noise, SVHN [37], LSUN [57], and TinyImageNet [25] for the CIFAR10/100-C datasets, and Textures [6] and Places365 [64] for ImageNet-C. To maintain consistency in domain shift between outliers and normal datasets, we apply the same corruptions [4] on outlier datasets. (*e.g.*, SVHN-C refers to the corruptions version of SVHN dataset.) Throughout all experiments, we maintain a fixed ratio of 8:2 between normal samples and outliers. Further details of the outlier datasets are provided in the supplementary material.

Baselines. To have a comprehensive comparison of existing TTA methods, we select eight TTA algorithms as our baselines. **Source** refers to the source pre-trained model. **BN Stats** [36] adjusts batch normalization (BN) layers [22] with the statistics of test data. **Tent** [51] optimizes the BN affine parameters in the model by entropy minimization. **CoTTA** [54] calculates weighted-average and augmentation-averaged pseudo-label to distill knowledge from the teacher model to the student model, and stochastically restores parameters to prevent the model from catastrophic forgetting. **EATA** [38] adopts the entropy and diversity weight with elastic weight consolidation [23] for anti-forgetting. **SAR** [39] excludes high-entropy samples from sharpness-aware entropy minimization and reset the model when it overfits. **RoTTA** [59] uses robust batch normalization and category-balanced sampling with timeliness and uncertainty to deal with the correlated label problem. **SoTTA** [17] applies high-confidence uniform-class sampling and entropy-sharpness minimization to alleviate the impact of noisy samples. **OWTTT** [28] proposes prototype expansion with prototype clustering and distribution alignment.

Evaluation Metrics. We evaluate the above baselines for both generalization and outlier detection. For generalization, we adopt classification accuracy (ACC) as a metric. For outlier detection, we consider the area under the ROC curve (AUC) [8], a threshold-free metric to measure the binary classification problem. The ROC curve depicts the relationship between the true positive rate (TPR) and the false positive rate (FPR). Models with better detection ability

[4] https://github.com/hendrycks/robustness

Table 3: Results (%) on **CIFAR10/100-C** datasets under outlier-aware TTA setting. We divide datasets into the normal part and the outlier part. In detail, we choose 8/80 classes as known samples and other 2/20 classes as outliers for CIFAR10/100-C. All the results are averaged over 15 different corruptions.

	CIFAR10-C (8:2)			CIFAR100-C (80:20)		
Method	ACC	AUC	H-score	ACC	AUC	H-score
Source	60.6	61.5	60.0	37.1	60.6	44.2
BN Stats [36]	79.4	66.9	72.6	55.5	66.2	60.2
Tent [51]	80.1	65.9	72.3	60.4	65.6	62.8
EATA [38]	81.5	67.9	74.1	61.3	67.4	64.2
SAR [39]	82.3	66.0	73.2	63.4	69.1	66.1
CoTTA [54]	81.8	65.4	72.5	57.1	66.3	61.3
RoTTA [59]	79.4	62.4	69.8	50.6	63.7	56.2
SoTTA [17]	82.7	62.7	71.3	61.4	68.2	64.6
OWTTT [28]	65.9	63.5	64.4	56.2	66.0	60.6
STAMP	**85.0**	**68.7**	**76.0**	**66.0**	**70.7**	**68.2**

will have a higher AUC score. To consider both generalization and detection performance simultaneously, we also report H-score:

$$H\text{-}score = \frac{2 \cdot ACC \cdot AUC}{ACC + AUC} \tag{9}$$

Implementation Details. We adopt the corruption with severity 5 in all experiments. We consider ResNet-18 [19] as backbones for CIFAR10-C, CIFAR100-C and ResNet-50 [19] for ImageNet-C. For CIFAR10-C and CIFAR100-C, we train the source model using standard cross-entropy loss with the initial learning rate of 0.1 and multi-step learning rate scheduling. For ImageNet-C, we use the released checkpoint from TorchVision library [35]. For a fair comparison, we set the batch size for all methods as 64 and the capacity of our memory bank is 64. Concerning the hyperparameters, we search the hyperparameters on a validation set and set the learning rate to 0.1/0.05/0.01, $\mathcal{H}_{thr}$ to $0.25/0.9/0.8 \times \ln|\mathcal{C}_S|$, T to 150/150/750 for CIFAR10-C/CIFAR100-C/ImageNet-C. Moreover, we set the other hyperparameter the same in experiments with $\beta = 0.1, \rho = 0.05, K = 16$. For parameters' updating, we select the affine parameters in the BN layer for optimization and freeze the top layer's parameter following [39]. More details of hyper-parameters searching and setting of baselines and STAMP can be found in the supplementary material.

4.2 Results

Results under outlier-aware TTA setting. We first evaluate TTA methods on the datasets with outliers. The results for CIFAR datasets are illustrated in Table 1. We observe that a simple baseline method, Tent [51], has a competitive generalization performance on distribution shift (achieving an accuracy of 80.1% on CIFAR10-C). However, it exhibits substantial degradation in the AUC score compared to Source, for instance, dropping from 70.4% to 48.7% on CIFAR10-C with Noise outlier dataset. This decline could be attributed to Tent's indiscriminate minimization of entropy across all samples, leading to a gradual loss of its

Table 4: Classification accuracies (%) on **CIFAR10-C** and **CIFAR100-C** under closed-set TTA setting.

	Method	*Gaussian*	*shot*	*impulse*	*defocus*	*glass*	*rotion*	*zoom*	*snow*	*frost*	*fog*	*brightness*	*contrast*	*elastic*	*pixelate*	*jpeg*	Avg.
CIFAR10-C	Source	28.7	35.2	24.2	57.3	49.0	66.4	64.8	76.7	62.9	73.4	90.3	31.4	78.8	46.5	74.7	57.3
	BN Stats [36]	70.2	72.0	63.6	87.6	66.5	85.9	86.9	82.1	80.5	84.1	90.8	85.4	77.3	78.6	74.3	79.0
	Tent [51]	76.1	78.4	70.5	87.8	70.1	86.8	87.5	84.8	82.0	85.1	91.1	86.6	79.4	82.7	78.5	81.8
	CoTTA [54]	76.1	77.6	73.9	87.9	71.9	86.9	87.5	82.8	82.0	85.2	90.8	85.7	79.3	80.4	78.3	81.8
	EATA [38]	76.5	77.9	71.7	89.1	70.4	87.4	**89.0**	85.2	84.2	86.0	91.5	**88.4**	79.9	83.9	78.5	82.6
	SAR [39]	70.7	72.1	66.8	87.6	68.2	85.9	86.9	82.1	80.5	84.1	90.8	85.4	77.3	78.6	74.3	79.4
	RoTTA [59]	69.6	71.0	62.9	87.5	67.0	85.9	86.9	82.4	79.6	84.7	91.2	73.5	78.3	77.9	74.9	78.2
	SoTTA [17]	75.8	79.2	71.5	**89.4**	70.6	**87.8**	**89.0**	85.6	84.0	87.2	**92.4**	87.4	79.9	84.5	79.0	82.9
	OWTTT [28]	70.1	72.0	63.0	86.9	66.5	85.7	86.7	82.7	80.9	84.5	91.2	83.0	77.9	76.8	74.9	78.9
	STAMP	**80.9**	**82.9**	**77.2**	87.6	**74.9**	86.6	87.7	**85.9**	**85.9**	**88.1**	90.4	87.2	**80.3**	**86.7**	**82.6**	**84.3**
CIFAR100-C	Source	12.4	14.5	7.2	36.0	44.7	45.1	45.2	49.5	41.6	36.9	63.3	13.2	57.5	23.8	46.6	35.8
	BN Stats [36]	41.1	41.3	38.9	63.1	51.5	60.8	63.8	51.2	53.5	53.2	64.4	59.0	58.4	58.0	47.6	53.7
	Tent [51]	52.3	52.1	47.7	66.9	56.1	64.3	65.3	58.3	58.7	60.0	67.8	62.1	61.8	63.0	54.5	59.4
	CoTTA [54]	47.0	47.8	45.1	59.6	54.0	59.5	61.3	53.3	55.0	52.9	62.6	50.3	58.1	61.8	53.1	54.8
	EATA [38]	50.7	53.5	48.1	67.0	55.6	64.8	67.0	59.1	59.2	60.4	67.7	63.9	61.8	63.3	54.6	59.8
	SAR [39]	**55.1**	55.0	51.2	68.4	58.2	66.0	67.4	60.3	60.8	61.9	69.8	65.5	63.6	66.2	56.8	61.7
	RoTTA [59]	35.9	36.6	33.8	60.6	47.1	57.7	60.8	48.0	42.2	50.8	59.2	32.1	53.8	52.3	44.4	47.7
	SoTTA [17]	51.6	53.8	47.4	66.9	56.9	65.3	68.1	58.9	60.1	60.1	69.4	63.1	62.3	62.8	54.8	60.1
	OWTTT [28]	41.6	42.8	38.8	63.4	52.6	61.6	64.8	53.3	54.8	54.5	65.8	58.4	60.1	57.6	49.3	54.6
	STAMP	**57.2**	**58.5**	**52.8**	**69.9**	**61.4**	**68.1**	**70.1**	**63.3**	**63.9**	**64.8**	**72.2**	**69.9**	**66.5**	**69.2**	**59.0**	**64.4**

ability to discern outliers. In addition, our experimental findings demonstrate that with judicious parameter selection, TTA methods can enhance not only the generalization capability against distribution shift but also the detection ability for outliers. Compared to existing TTA methods, STAMP achieves the most favorable performance on both classification and outlier detection. On CIFAR10-C, STAMP achieves the highest ACC, and +4.1% huge improvements in AUC compared to the previous methods. Regarding CIFAR100-C, STAMP exhibits +2.8% and +5.7% improvements compared to the most effective TTA methods on ACC and AUC, respectively. We also report the results on the ImageNet-C in Table 2. STAMP still achieves the highest performance. Specifically, we outperform existing TTA methods with +1.2%, +1.6% on H-score with Places365-C and Textures-C outlier datasets, respectively.

Additionally, we evaluate the TTA algorithm under a more challenging scenario involving homogeneous outlier datasets in Table 3, which makes it more difficult to identify outliers. STAMP also exhibits the most promising performance on CIFAR10-C and CIFAR100-C under this scenario.

Concerning the above results, we achieve optimal results in most cases, affirming the robustness and effectiveness of STAMP.

Results under closed-set TTA setting. Additionally, we assess our method in a scenario without outliers as most TTA works do. We present the classification accuracy on the clean dataset in Table 4. Remarkably, our method attains the highest accuracy on both CIFAR10-C and CIFAR100-C datasets, achieving an average improvement of +2.9% over the second-best method (SoTTA). This shows that STAMP still has strong applicability even in the traditional situation without outliers.

Table 5: Ablation studies on reliable class-balanced memory (RBM), self-weighted entropy minimization (S.W.), sharpness-aware minimization (S.A.), and decay step size (D.S.). Experiments are conducted on **CIFAR100-C** with the outlier dataset **SVHN-C**. The first row in the table reports the results of entropy minimization.

S.A.	D.S.	RBM	S.W.	ACC	AUC	H-score
-	-	-	-	54.4	70.4	61.2
-	✓	-	-	58.9	65.3	61.8
✓	-	-	-	56.4	72.8	63.5
✓	✓	-	-	59.2	70.8	64.4
✓	✓	-	✓	59.3	77.7	67.1
✓	✓	✓	-	63.0	75.0	68.4
✓	✓	✓	✓	63.7	80.0	70.9

To summarize, all the above results prove that our method achieves the best results under both outlier-aware TTA and conventional closed-set settings.

4.3 Ablation Study

Effect of each component. We delve into the role of each component within our method as follows. First, we provide the results of different weight strategies of entropy minimization in Figure 3. We compare our adaptive weight with static weight and EATA weight [38]. EATA calculates the weight by $exp[\mathcal{H}_{thr} - \mathcal{H}(p(x))]$ and static weight calculates weight in the same way of Equation (6). Different from adaptive weight, both EATA and static weight don't involve any gradient back-propagation. As depicted in Figure 3, adaptive weight shows the best performance among the strategies.

Furthermore, we conduct an ablation study on each component of our method, as summarized in Table 5. We observe that the effect of combining test-time step size decay and SAM is better than using either of them alone. Step size decay can

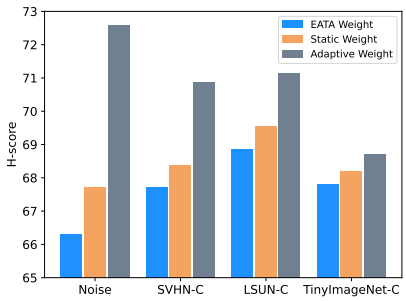


Fig. 3: Results for the different weight strategies on dataset CIFAR100-C with SVHN-C outlier dataset. Static weight represents that the w_i in Equation (6) without gradient back-propagation. EATA weight represents the weight for reliable samples used in EATA [38].

Table 6: H-score (%) with the different ratios of outliers to total samples on **CIFAR100-C** with different outlier datasets.

	SVHN-C	LSUN-C	TinyImageNet-C
5%	68.4	70.2	68.3
10%	70.1	70.9	68.8
20%	70.9	71.2	68.7
33%	70.3	70.1	68.1
50%	69.3	68.8	67.4

improve the ACC but result in AUC decline. However, combining SAM and step size decay mitigates the decline in AUC. In addition, after the introduction of self-weighted entropy minimization, the AUC of the model further increased to 77.7%. Notably, by removing reliable class-balanced memory from our method, the ACC decreases from 63.7% to 59.3%. It reveals that reliable class-balanced memory significantly contributes to enhancing the generalization ability to distribution shift. Moreover, by removing the self-weighted entropy minimization in STAMP, its AUC drops from 80.0% to 75.0%. Self-weighted entropy minimization plays an important role in outlier detection. Besides decreasing the sample entropy, self-weighted entropy minimization incorporates entropy into the optimization weight, allowing outliers that deviate from the majority of samples to receive lower weights, consequently increasing their entropy. This mechanism enhances outlier detection performance.

Effect of outlier proportion. In the realistic scenario, the proportion of outliers can vary considerably. We investigate how this variability affects the performance of our method. Specifically, we manipulate the proportion of outliers, ranging from 5% to 50%, and present the corresponding H-scores in Table 6. The results show that the fluctuation of our method is within 3%. The findings suggest that STAMP exhibits robustness across different scenarios with varying proportions of outliers, as it does not experience significant degradation.

5 Conclusion

In this study, we research the challenge of handling outliers in test-time adaptation. To tackle this issue effectively, we introduce a scenario called outlier-aware test-time adaptation, which refers to conducting both recognition and outlier rejection simultaneously. Additionally, we propose STAMP, using augmentation-averaged estimation and a filtering mechanism to store reliable samples in memory. Moreover, we dynamically discard samples with the highest class frequency from memory to maintain a class-balanced prediction. Then we replay the samples stored in the memory with self-weighted entropy minimization. Furthermore, we leverage sharpness-aware minimization and learning rate decay to bolster the stability of our algorithm. Through extensive experiments across multiple benchmarks, we demonstrate that our method surpasses existing TTA methods, underscoring its effectiveness and versatility in open-world applications.

Acknowledgements

We thank Yuhe Ding and Zhengbo Wang for their valuable feedback on our work. This work was funded by the Beijing Municipal Science and Technology Project (No. Z231100010323005), the Beijing Nova Program (No. Z211100002121108), the Young Elite Scientists Sponsorship Program by CAST (2023QNRC001), and the National Natural Science Foundation of China under (No. 62276256).

References

1. An, S., Park, S., Kim, G., Baek, J., Lee, B., Kim, S.: Context enhanced transformer for single image object detection. arXiv preprint arXiv:2312.14492 (2023) 2
2. Boudiaf, M., Mueller, R., Ben Ayed, I., Bertinetto, L.: Parameter-free online test-time adaptation. In: Proc. CVPR (2022) 4
3. Cao, H., Xu, Y., Yang, J., Yin, P., Yuan, S., Xie, L.: Multi-modal continual test-time adaptation for 3d semantic segmentation. In: Proc. ICCV (2023) 2
4. Chakrabarty, G., Sreenivas, M., Biswas, S.: Santa: Source anchoring network and target alignment for continual test time adaptation. Transactions on Machine Learning Research (2023) 4
5. Chen, D., Wang, D., Darrell, T., Ebrahimi, S.: Contrastive test-time adaptation. In: Proc. CVPR (2022) 2, 4
6. Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., Vedaldi, A.: Describing textures in the wild. In: Proc. CVPR (2014) 3, 10
7. Croce, F., Andriushchenko, M., Sehwag, V., Debenedetti, E., Flammarion, N., Chiang, M., Mittal, P., Hein, M.: Robustbench: a standardized adversarial robustness benchmark. In: Proc. NeurIPS (2021) 1
8. Davis, J., Goadrich, M.: The relationship between precision-recall and roc curves. In: Proc. ICML (2006) 10
9. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: Proc. CVPR (2009) 3, 9
10. DeVries, T., Taylor, G.W.: Learning confidence for out-of-distribution detection in neural networks. arXiv preprint arXiv:1802.04865 (2018) 3, 4
11. Ding, Y., Liang, J., Jiang, B., Zheng, A., He, R.: Maps: A noise-robust progressive learning approach for source-free domain adaptive keypoint detection. IEEE Transactions on Circuits and Systems for Video Technology (2023) 3
12. Ding, Y., Sheng, L., Liang, J., Zheng, A., He, R.: Proxymix: Proxy-based mixup training with label refinery for source-free domain adaptation. Neural Networks **167**, 92–103 (2023) 3
13. Farahani, A., Voghoei, S., Rasheed, K., Arabnia, H.R.: A brief review of domain adaptation. Advances in Data Science and Information Engineering: Proceedings from ICDATA 2020 and IKE 2020 pp. 877–894 (2021) 1
14. Foret, P., Kleiner, A., Mobahi, H., Neyshabur, B.: Sharpness-aware minimization for efficiently improving generalization. In: Proc. ICLR (2021) 3, 8
15. Gebru, T., Hoffman, J., Fei-Fei, L.: Fine-grained recognition in the wild: A multi-task domain adaptation approach. In: Proc. ICCV (2017) 3
16. Gong, T., Jeong, J., Kim, T., Kim, Y., Shin, J., Lee, S.J.: Note: Robust continual test-time adaptation against temporal correlation. In: Proc. NeurIPS (2022) 4, 9
17. Gong, T., Kim, Y., Lee, T., Chottananurak, S., Lee, S.J.: Sotta: Robust test-time adaptation on noisy data streams. In: Proc. NeurIPS (2023) 4, 5, 8, 9, 10, 11, 12

18. Goyal, S., Sun, M., Raghunathan, A., Kolter, J.Z.: Test time adaptation via conjugate pseudo-labels. In: Proc. NeurIPS (2022) 9
19. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proc. CVPR (2016) 11
20. Hendrycks, D., Dietterich, T.: Benchmarking neural network robustness to common corruptions and perturbations. In: Proc. ICLR (2018) 3, 9
21. Hendrycks, D., Gimpel, K.: A baseline for detecting misclassified and out-of-distribution examples in neural networks. In: Proc. ICLR (2016) 4
22. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proc. ICML (2015) 10
23. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al.: Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences **114**(13), 3521–3526 (2017) 10
24. Krizhevsky, A.: Learning multiple layers of features from tiny images. Master's thesis, University of Tront (2009) 9
25. Le, Y., Yang, X.: Tiny imagenet visual recognition challenge. CS 231N **7**(7), 3 (2015) 3, 10
26. Lee, J., Das, D., Choo, J., Choi, S.: Towards open-set test-time adaptation utilizing the wisdom of crowds in entropy minimization. In: Proc. CVPR (2023) 4, 8
27. Lee, K., Lee, K., Lee, H., Shin, J.: A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In: Proc. NeurIPS (2018) 4
28. Li, Y., Xu, X., Su, Y., Jia, K.: On the robustness of open-world test-time training: Self-training with dynamic prototype expansion. In: Proc. CVPR (2023) 4, 9, 10, 11, 12
29. Liang, J., He, R., Tan, T.: A comprehensive survey on test-time adaptation under distribution shifts. International Journal of Computer Vision (2024) 1
30. Liang, J., Hu, D., Feng, J.: Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In: Proc. ICML (2020) 1
31. Liang, J., Sheng, L., Wang, Z., He, R., Tan, T.: Realistic unsupervised clip fine-tuning with universal entropy optimization. In: Proc. ICML (2024) 4
32. Liang, S., Li, Y., Srikant, R.: Enhancing the reliability of out-of-distribution image detection in neural networks. In: Proc. ICLR (2018) 4
33. Lin, Z., Roy, S.D., Li, Y.: Mood: Multi-level out-of-distribution detection. In: Proc. CVPR (2021) 4
34. Liu, W., Wang, X., Owens, J., Li, Y.: Energy-based out-of-distribution detection. In: Proc. NeurIPS (2020) 4
35. Maintainers, T., Contributors: Torchvision: Pytorch's computer vision library (2016) 11
36. Nado, Z., Padhy, S., Sculley, D., D'Amour, A., Lakshminarayanan, B., Snoek, J.: Evaluating prediction-time batch normalization for robustness under covariate shift. In: Proc. ICML Workshops (2020) 3, 9, 10, 11, 12
37. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning (2011) 3, 10
38. Niu, S., Wu, J., Zhang, Y., Chen, Y., Zheng, S., Zhao, P., Tan, M.: Efficient test-time model adaptation without forgetting. In: Proc. ICML (2022) 2, 4, 5, 6, 7, 9, 10, 11, 12, 13
39. Niu, S., Wu, J., Zhang, Y., Wen, Z., Chen, Y., Zhao, P., Tan, M.: Towards stable test-time adaptation in dynamic wild world. In: Proc. ICLR (2022) 4, 7, 8, 9, 10, 11, 12

40. Nixon, J., Dusenberry, M.W., Zhang, L., Jerfel, G., Tran, D.: Measuring calibration in deep learning. In: Proc. CVPR Workshops (2019) 5
41. Park, S., Yang, S., Choo, J., Yun, S.: Label shift adapter for test-time adaptation under covariate and label shifts. In: Proc. CVPR (2023) 4
42. Press, O., Schneider, S., Kümmerer, M., Bethge, M.: Rdumb: A simple approach that questions our progress in continual test-time adaptation. In: Proc. NeurIPS (2023) 8
43. Schneider, S., Rusak, E., Eck, L., Bringmann, O., Brendel, W., Bethge, M.: Improving robustness against common corruptions by covariate shift adaptation. In: Proc. NeurIPS (2020) 3
44. Shanmugam, D., Blalock, D., Balakrishnan, G., Guttag, J.: Better aggregation in test-time augmentation. In: Proc. CVPR (2021) 5
45. Sheng, L., Liang, J., He, R., Wang, Z., Tan, T.: Adaptguard: Defending against universal attacks for model adaptation. In: Proc. ICCV (2023) 3
46. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: Proc. ICML (2019) 1
47. Tong, P., Su, W., Li, H., Ding, J., Haoxiang, Z., Chen, S.X.: Distribution free domain generalization. In: Proc. ICML (2023) 3
48. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Proc. NeurIPS (2017) 1
49. Volpi, R., De Jorge, P., Larlus, D., Csurka, G.: On the road to online adaptation for semantic image segmentation. In: Proc. CVPR (2022) 2
50. Vyas, A., Jammalamadaka, N., Zhu, X., Das, D., Kaul, B., Willke, T.L.: Out-of-distribution detection using an ensemble of self supervised leave-out classifiers. In: Proc. ECCV (2018) 4
51. Wang, D., Shelhamer, E., Liu, S., Olshausen, B., Darrell, T.: Tent: Fully test-time adaptation by entropy minimization. In: Proc. ICML (2020) 2, 4, 7, 9, 10, 11, 12
52. Wang, J., Lan, C., Liu, C., Ouyang, Y., Qin, T., Lu, W., Chen, Y., Zeng, W., Yu, P.S.: Generalizing to unseen domains: A survey on domain generalization. IEEE Transactions on Knowledge and Data Engineering **35**(8), 8052–8072 (2023) 3
53. Wang, M., Deng, W.: Deep visual domain adaptation: A survey. Neurocomputing **312**, 135–153 (2018) 3
54. Wang, Q., Fink, O., Van Gool, L., Dai, D.: Continual test-time domain adaptation. In: Proc. CVPR (2022) 4, 9, 10, 11, 12
55. Wang, Y., Li, B., Che, T., Zhou, K., Liu, Z., Li, D.: Energy-based open-world uncertainty modeling for confidence calibration. In: Proc. CVPR (2021) 4
56. You, K., Long, M., Wang, J., Jordan, M.I.: How does learning rate decay help modern neural networks? arXiv preprint arXiv:1908.01878 (2019) 9
57. Yu, F., Seff, A., Zhang, Y., Song, S., Funkhouser, T., Xiao, J.: Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. arXiv preprint arXiv:1506.03365 (2015) 3, 10
58. Yu, Y., Sheng, L., He, R., Liang, J.: Benchmarking test-time adaptation against distribution shifts in image classification. arXiv preprint arXiv:2307.03133 (2023) 1
59. Yuan, L., Xie, B., Li, S.: Robust test-time adaptation in dynamic scenarios. In: Proc. CVPR (2023) 2, 3, 9, 10, 11, 12
60. Yurtsever, E., Lambert, J., Carballo, A., Takeda, K.: A survey of autonomous driving: Common practices and emerging technologies. IEEE Access **8**, 58443–58469 (2020) 1
61. Zhang, M., Levine, S., Finn, C.: Memo: Test time robustness via adaptation and augmentation. In: Proc. NeurIPS (2022) 4, 5

62. Zhang, Y., Wang, X., Jin, K., Yuan, K., Zhang, Z., Wang, L., Jin, R., Tan, T.: Adanpc: Exploring non-parametric classifier for test-time adaptation. In: Proc. ICML (2023) 2
63. Zhao, H., Liu, Y., Alahi, A., Lin, T.: On pitfalls of test-time adaptation. In: Proc. ICML (2023) 4
64. Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., Torralba, A.: Places: A 10 million image database for scene recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **40**(6), 1452–1464 (2017) 3, 10
65. Zhou, Z., Guo, L.Z., Jia, L.H., Zhang, D., Li, Y.F.: Ods: test-time adaptation in the presence of open-world data shift. In: Proc. ICML (2023) 4