Improved Self-Training for Test-Time Adaptation

Jing Ma
Huazhong University of Science and Technology
jingma0011@gmail.com

Abstract

Test-time adaptation (TTA) is a technique to improve the performance of a pre-trained source model on a target distribution without using any labeled data. However, existing self-trained TTA methods often face the challenges of unreliable pseudo-labels and unstable model optimization. In this paper, we propose an Improved Self-Training (IST) approach, which addresses these challenges by enhancing the pseudo-label quality and stabilizing the adaptation process. Specifically, we use a simple augmentation strategy to generate multiple views of each test sample, and construct a graph structure to correct the pseudo-labels based on the similarity of the latent features. Moreover, we adopt a parameter moving average scheme to smooth the model updates and prevent catastrophic forgetting. Instead of using a model with fixed label space, we explore the adaptability of the foundation model CLIP to various downstream tasks at test time. Extensive experiments on various benchmarks show that IST can achieve significant and consistent improvements over the existing TTA methods in classification, detection, and segmentation tasks.

1. Introduction

Deep Neural Networks have performed superiorly in a broad scope of tasks, and exhibit surprising zero-shot generalization capabilities. However, in practical applications, behavior decline may occur owing to the distribution shifts of unknown domains, which leads to laborious efforts in data collection and model retraining. To address this issue, Test-Time Adaptation (TTA), aims to adapt the model to the target domain at test time, has emerged as a promising paradigm of increasing attention and research [19, 29, 30].

Early methods of TTA [22, 27, 30, 32] explore adapting the activation statistics and updating the affine transformation parameters of batch normalization layers to mitigate the distribution shifts from clean to corrupted images. These approaches are easily scalable to deep neural networks with different architectures, but their performance is often confined to specific ranges of distribution shifts due to the limited adaptive capacity of batch normalization [2]. A series of works [4, 8, 20, 29] leverage self-supervised learning for test-time training (TTT) to update the model parameters before making predictions, which is conditional on having correlated gradients between the loss functions of the auxiliary and main tasks. However, an inappropriate pretext task may be detrimental to the learned model [20]. Moreover, although TTT is not restricted by the type of main task, it typically requires modifying the training process to accommodate a self-supervised task, which increases the cost of applying TTT to pre-trained models [36].

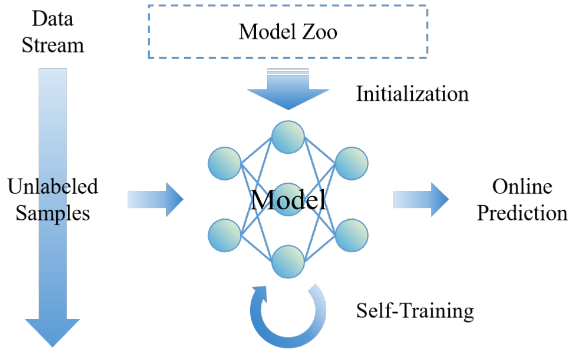


Figure 1. An illustration of test-time adaptation. The approach employs source models from a model zoo, where the source data is unknown. Source models adapt to target domains using one batch of test samples at a time. Self-training is a key component for updating models and providing more accurate online predictions.

Another line of research seeks to utilize the predictions of the model itself as pseudo-labels and re-train the model, which is known as the self-training strategy. Owing to the poor calibration of neural networks [10], pseudo-labels generated by the source model may be inaccurate and noisy for target data. Liang, *et al.* [19] propose to attain class-wise prototypes by clustering and further obtain cleaner pseudo-labels. Recent works [3, 18, 28, 31, 35] suggest performing different data augmentations on a test sample to improve the model's prediction. On the other hand, some works [3, 24, 31, 32] focus on stabilizing the adaptation procedure at test time by adopting active selection, re-weighting strategy, or knowledge distillation. Generally, self-training-based approaches make no strong assumptions about distribution shifts and types of task, thus they are more ubiquitous for real-world applications. However, existing methods still suffer from the following limitations: 1) Test-time augmentation may degrade performance for voluminous and aggregated categories, 2) Pseudo-label correction has not fully exploited the model's intermediate representations of the target domain. 3) Error accumulation and catastrophic forgetting may occur during the adaptation process.

In this paper, we aim to address the above issues by analyzing the key components of self-training, *i.e.*, pseudo-label generation, correction, and model optimization. We challenge the task of fully test-time adaptation [30], which supposes that any information about source data is unavailable, including original samples, intermediate features, and all kinds of statistics. To this end, we propose an **I**mproved **S**elf-**T**raining (IST) method. In contrast to applying complex transformations, IST generates pseudo-labels with a simple test-time augmentation strategy to mitigate the poor calibration caused by unused functions. Moreover, an algorithm is proposed to correct the pseudo-labels by leveraging graph structures constructed with target samples in the latent space. Furthermore, we compare common optimization polices and propose parameter moving average to stabilize the adaptation process and alleviate catastrophic forgetting. Finally, we scale up IST to basic computer vision tasks and conduct extensive experiments on various benchmarks. The results demonstrate that IST achieves credible and consistent performance gains over existing TTA methods. [1]

In summary, our main contributions are as follows:

- We propose a plug-and-play strategy for generating and correcting pseudo-labels, which is effective and efficient for test-time augmentation without additional retraining.
- We propose an improved self-training method for fully test-time adaptation, which is scalable to a variety of pre-trained models and application tasks.
- We exhibit the effectiveness and robustness of our method on various benchmarks, including image classification, object detection, and semantic segmentation.

2. Related Work

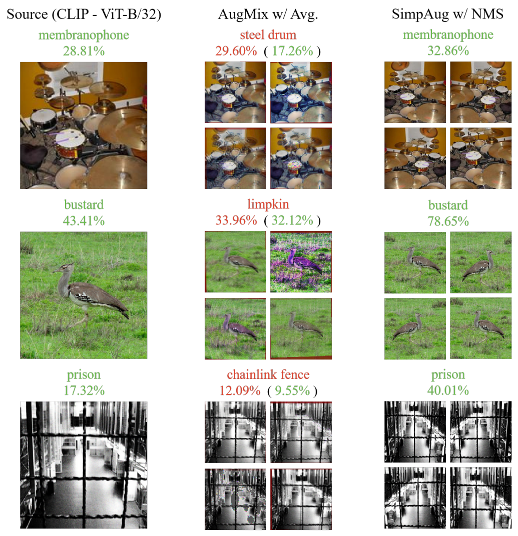
Test-Time Adaptation (TTA) is a challenging task, that aims to adapt the model to arbitrary distributions and achieve improved performance at test time. BN Adapt [27] replaces the activation statistics of batch normalization layers with the estimated statistics of the corrupted images to improve the robustness of the model. Further, Tent [30] discards the statistics from the source data and updates the affine transformation parameters by entropy minimization. Sun, *et al.* [29] propose introducing the self-supervised pretext task to adapt the main task by updating the model parameters before making predictions. Following this way, TTT [29] and TTT-MAE [8] employ rotation prediction and image generation as auxiliary tasks, respectively. TTT++ [20] and ITTA [4] rely on more general contrastive learning tasks. To effectively utilize a model without source data to solve domain adaptation problems, SHOT [19] exploits information maximization and self-supervised pseudo-labeling to align the target representation to the source hypothesis. MEMO [35] and SFOD [18] augment test samples to obtain robust pseudo labels and self-train the model with entropy descent. AdaContrast [3] denoises pseudo-labels via soft voting among nearest neighbors and applies weak-strong augmentation consistency as a regularizer. For non-stationary and continually changing environments, RoTTA [32] and CoTTA [31] are proposed to effectively stabilize the adaptation procedure and avoid catastrophic forgetting.

Existing approaches tend to address the distribution shift caused by image corruption, but infrequently consider unconstrained conditions in nature, such as label shift and correlation shift. Hence, we propose to employ the Contrastive Language-Image Pre-trained model (CLIP) [25] as the source model for test-time adaptation. In this way, we have no need to pre-train models on different benchmarks, thus avoiding additional assumptions of the label distribution. Meanwhile, CLIP can provide a suitable starting point for the adaptation process due to its zero-shot capability and generalized feature properties, which have been proven to have a significant impact on post-adaptation [36]. From an application perspective, exploring how to improve the performance of CLIP in realistic environments at test time has important practical value for foundation models.

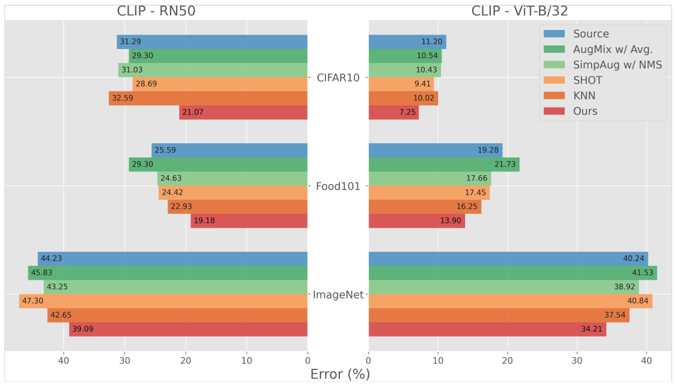
3. Method

We address the problem of adapting a pre-trained model to new domains at test time. Given a source model f_θ with parameters θ_S trained on the source domain $\mathcal{D}_S$, the goal is to adapt it to the target domain $\mathcal{D}_T$ with only unlabeled data $\{x_i\}_{i=1}^n$. We explicitly divide the model into encoder g_φ and predictor h_ϕ, and denote $f = h \circ g$. By default, we define the final linear layer as the predictor and the rest as the encoder. A data point $x_i \in \mathcal{X}$ is projected into a latent space $z_i \in \mathcal{Z}$ by the encoder, and then mapped to a label space $y_i \in \mathcal{Y}$ by the predictor. Our proposed method applies a **Simp**le **Aug**mentation (**SimpAug**) to the original image and produces augmented points $\widetilde{x} \sim \mathcal{U}(x)$. Expanded data sets $\{x\} \cap \{\widetilde{x}\}$, consisting of original and augmented samples, are fed into the encoder to generate intermediate features $\{z\}$. Subsequently, we perform a **P**seudo-**L**abel **C**orrection **A**lgorithm (**PLCA**) to reinforce the prediction of unlabeled data by utilizing the graph structure in the latent space. To stabilize the self-training process at test time, we analyze the primary optimization algorithms and adopt **P**arameter **M**oving **A**verage (**PMA**) for iterative updating. The pipeline of our method can be referred to Fig. 3.

[1]The code is available at https://github.com/JingInAI/IST4TTA



(a)



(b)

Usage Time (seconds)	SHOT	KNN	Ours
CLIP - RN50	7.501	1.528	3.911
CLIP - ViT-B/32	3.239	1.239	3.601

(c)

Figure 2. (a) Visualization of images in ImageNet [6] with two kinds of test-time augmentation polices. Predictions are provided by CLIP (ViT-B/32), with correct and incorrect classes marked in green and red, respectively. (b) Error rate of pseudo-labeling counted on the test/validation set for each benchmark. (c) Computational speed of three pseudo-label correction algorithms on a fixed-size 10K sample set, averaged over 10 runs. The dimension of latent space is 1024 for RN50 and 512 for ViT-B/32.

3.1. Simple Augmentation Strategy

Typically, existing test-time augmentation methods tend to employ sophisticated augmentation functions to be applied to the test image [35]. Although this is a reasonable approach to extract invariant encodings from data augmentations, it may lead to serious corruption of original attributes. Fig. 2 (a) shows that the differences between classes are reflected in a few attribute variations for class-dense or fine-grained tasks. Complex augmentations, such as altering the brightness, contrast, saturation, and hue of the image, will impair the representative attributes of the category. Another type of augmentation, such as image rotation and adding Gaussian noise, will cause high-confidence erroneous predictions, especially for models without the use of similar transformations during training, *e.g.* CLIP [25].

To this end, we consider the standard policy (crops, scales, and horizontal flips) for test-time augmentation. On the one hand, crops and scales can zoom in on the local regions of the image, reduce the interference of background pixels, and increase the model's attention to tiny objects. On the other hand, we preserve the position information of augmented versions in the original coordinate system and project the predictions to the primal image space, which is beneficial for solving regression problems, e.g. detection and segmentation. Specifically, we define the ratio of the cropped region to the original area as $r \sim U(r_{min}, r_{max})$ and the aspect ratio as $a \sim U(a_{min}, a_{max})$. Given an image x with the size of $H \times W$, the length and width of the cropped region are calculated as

$$w = \sqrt{H \times W \times r \times a}, \quad h = \sqrt{H \times W \times r \,/\, a}. \quad (1)$$

Then, we randomly select the top-left corner of the cropped region (i, j) from $U(0, H-h)$ and $U(0, W-w)$, respectively. The cropped sub-image is flipped horizontally with a probability of p_{hf} and scaled to the original size $H \times W$. In general, we set $r_{min} = 0.7, r_{max} = 1.0$ to ensure the clarity of augmented images, and $a_{min} = 0.75 \times H/W, a_{max} = 1.33 \times H/W$ to avoid the distortion of the aspect ratio. p_{hf} is set to 0.5 by default. We save the set of parameters (i, j, h, w, hf) for each augmented version, where hf is a binary variable indicating whether the image is flipped.

We discard the operation of averaging the predictions of augmented samples, since it implicitly assumes that the augmentation functions have no influence on which predictions are corrected and which are corrupted [28]. Instead, motivated by Non-Maximum Suppression (NMS) [23], we extract the predictive probabilities of images in the augmentation space $\{\widetilde{x}_j\}_{j=1}^m \sim \mathcal{U}(x_i)$, and select the most confident class as the final prediction, which is formulated as

$$\hat{y}(x_i) = \arg\max_c \, \{p_c(\widetilde{x}_j)\}_{j=1}^m, \quad (2)$$

where $c \in \mathcal{Y}$ is the class label, and p_c is the probability of class c. In contrast to the image classification task, NMS is applied to the set of detected boxes with high confidence scores and IOU values for object detection, and applied to the set of predictions for a single pixel from augmented versions for semantic segmentation.

We compare the performance with MEMO [35], using AugMix [13] and averaging predictive probabilities for test-time augmentation, referred to Fig. 2 (a, b). Results demonstrate that SimpAug is more robust in class-dense and fine-grained benchmarks, *e.g.* ImageNet [6] and Food101 [1].

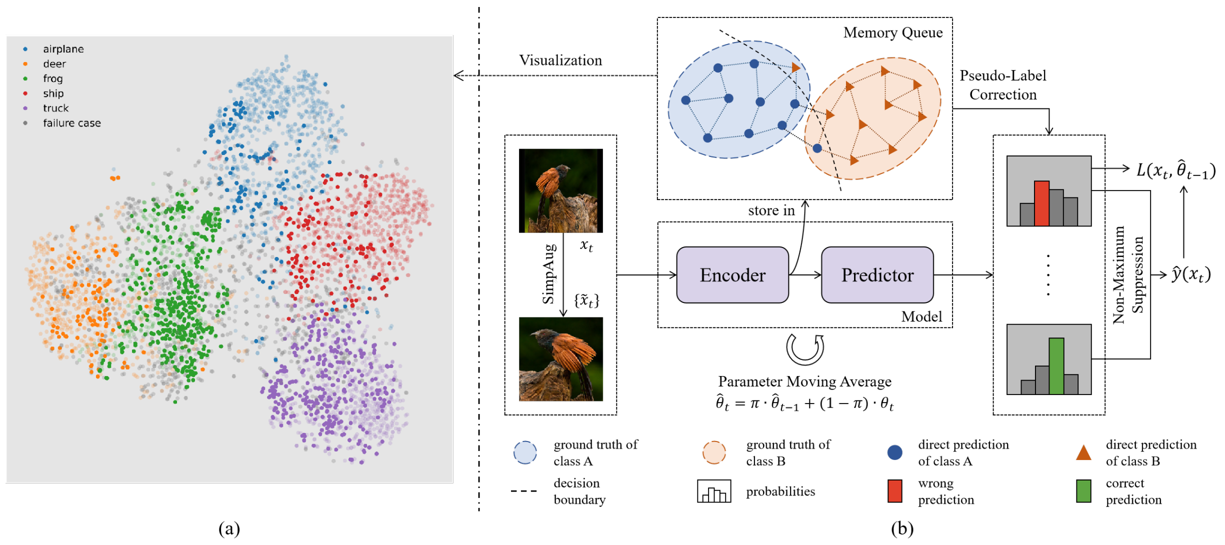


Figure 3. **The framework of our approach.** At the beginning of test-time adaptation, the model is initialized with the source parameters $\hat{\theta}_0 = \theta_S$. A target sample x_t arriving at time step t is transformed by SimpAug. We maintain a fixed-size memory queue for storing feature vectors and probabilities of target samples and their augmented versions. PLCA utilizes the graph structure formed by the stored vectors to correct the pseudo-labels generated by the model $\hat{\theta}_{t-1}$, and then employs NMS to select the most reliable prediction. PMA modifies the optimized parameters θ_t to preserve the knowledge of the source model. We visualize the features extracted by the image encoder of CLIP (RN50) on CIFAR10. Translucent-colored dots represent correctly classified images, gray dots represent failure cases, and opaque dots represent misclassified samples corrected by PLCA.

3.2. Pseudo-Label Correction Algorithm

Although test-time augmentation can improve the robustness of distribution shifts, the pseudo labels annotated by the source model are still inaccurate and noisy for target data. To obtain cleaner labels, a series of methods [3, 19] explore the latent space of the source model to correct erroneous labeling. By visualizing the features on target domains using CLIP's visual encoder [25] in Fig. 3, we observe that the representation of different classes in the latent space exhibits clustering patterns. However, numerous target samples in the same cluster or manifold are misclassified. Inspired by [37], we propose a pseudo-label correction algorithm through learning from the graph structure constructed with only target data.

We consider feature vectors in the latent space $\{z_i\}_{i=1}^n \in \mathcal{Z}$ as vertices in a graph, and the edges between vertices are defined by the distances between a point and its K-nearest neighbors. Specifically, the edges are weighted by

$$w_{ij} = \begin{cases} 1, & i = j, \\ 1 - \frac{\|z_i - z_j\|_2}{\max_j \|z_i - z_j\|_2}, & z_j \in \mathcal{N}_K(z_i), \\ 0, & \text{otherwise}. \end{cases} \tag{3}$$

Note that self-reinforcement is added to prevent the instability caused by the accumulation of erroneous predictions in the spread process. By default, we set $K = 50$. To guarantee the information spread symmetrically, we build a symmetric non-negative matrix $W = (w_{ij})_{n\times n} + (w_{ji})_{n\times n}$, and normalize it by $S = D^{-1/2}WD^{-1/2}$ for following iterations, where D is a diagonal matrix with $D_{ii} = \sum_j w_{ij}$.

We define an initial label matrix $Y(0) = (y_{ij})_{n\times m}$, where $y_{ij} = p_j(x_i)$ is the predictive probability of the sample x_i belonging to the class j. Afterward, we iteratively update the label matrix by

$$Y(t+1) = \alpha S Y(t) + (1-\alpha)Y(0), \tag{4}$$

where α is a hyper-parameter to balance the influence of initial labels and the information spread. By repeating the Eqn. 4 for T times, we obtain the final label matrix

$$Y(T) = \alpha^T S^T Y(0) + (1-\alpha)\sum_{t=0}^{T-1} \alpha^t S^t Y(0). \tag{5}$$

Since $0 < \alpha < 1$ and the spectral radius of S is less than or equal to 1, we have

$$\begin{aligned} &\lim_{T\to\infty} \alpha^T S^T Y(0) = 0, \\ &\lim_{T\to\infty} \sum_{t=0}^{T-1} \alpha^t S^t Y(0) = (I - \alpha S)^{-1} Y(0). \end{aligned} \tag{6}$$

Therefore, the final label matrix can be approximated by [2]

$$Y^* = \lim_{T\to\infty} Y(T) = (1-\alpha)(I-\alpha S)^{-1} Y(0). \tag{7}$$

Finally, we correct the pseudo labels in Eqn. 2 by

$$\hat{y}(x_i) = \arg\max_c \{p_c(\tilde{x}_j) \times \mathbb{1}_{\arg\max_k y^*_{jk} = c}\}_{j=1}^m, \tag{8}$$

[2]This linear system can be solved by the conjugate gradient method because $I - \alpha S$ is symmetric and positive definite. By default, we set $\alpha = 0.99$ and iterate for at most 20 times following [14].

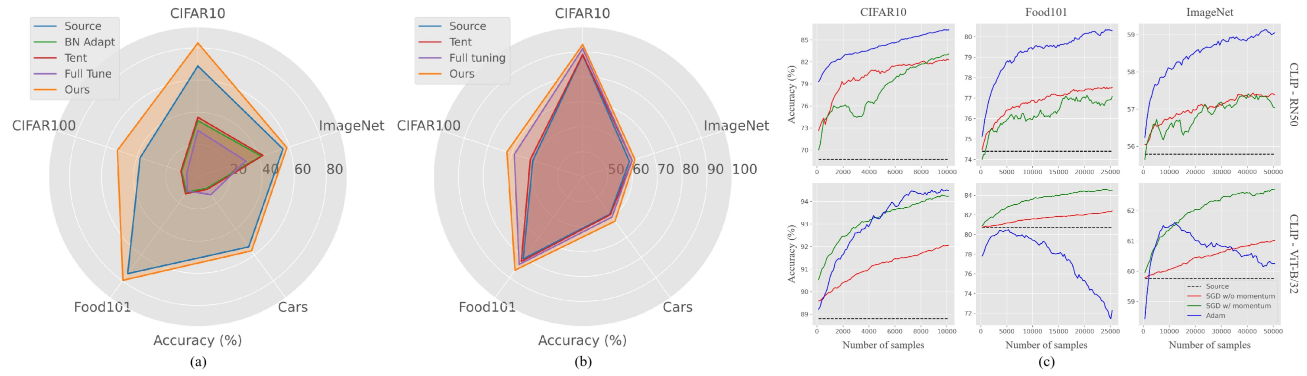


Figure 4. Online test-time adaptation following [29]. We compare the impact of updating different parts of CLIP and show the top-1 accuracy (%) in five benchmarks with RN50 (a) and ViT-B/32 (b). To explore the impact of three optimizers on CNN-based and Transformer-based models, we freeze the text encoder in CLIP and update the image encoder. The adaptation process is visualized in (c).

Methods	Models	CIFAR10	CIFAR100	Food101	Cars	ImageNet	Models	CIFAR10	CIFAR100	Food101	Cars	ImageNet
CLIP (zero-shot) [25]	ResNet-50	68.7±0.0	37.9±0.0	74.4±0.0	53.9±0.0	55.8±0.0	ResNet-101	78.1±0.0	41.2±0.0	77.5±0.0	60.7±0.0	58.2±0.0
Ours (SimpAug)		69.0±0.1	38.7±0.1	75.4±0.0	54.0±0.0	56.8±0.0		78.8±0.1	42.0±0.1	78.7±0.0	61.0±0.0	59.2±0.0
Ours (PLCA)		78.9±0.1	49.0±0.1	80.8±0.0	62.0±0.0	**60.9**±0.0		84.2±0.1	55.0±0.1	83.9±0.0	68.8±0.0	**63.2**±0.0
Ours (Online)		83.0±0.2	52.8±0.4	79.5±0.2	56.9±0.4	58.3±0.1		89.1±0.1	59.0±0.2	82.6±0.2	64.0±0.3	60.1±0.0
Ours (Offline)		**84.6**±0.1	**54.1**±0.5	**81.0**±0.0	**65.9**±0.1	60.2±0.0		**92.1**±0.1	**63.4**±0.1	**85.1**±0.0	**71.4**±0.2	62.6±0.0
CLIP (zero-shot) [25]	ViT-B/32	88.8±0.0	61.1±0.0	80.7±0.0	58.6±0.0	59.8±0.0	ViT-B/16	89.3±0.0	64.0±0.0	86.6±0.0	63.9±0.0	64.5±0.0
Ours (SimpAug)		89.6±0.1	62.1±0.1	82.3±0.0	58.9±0.0	61.1±0.0		90.1±0.1	65.7±0.1	87.6±0.0	64.3±0.0	65.5±0.0
Ours (PLCA)		92.8±0.1	69.7±0.1	86.1±0.0	65.7±0.0	65.8±0.0		94.3±0.0	73.0±0.1	90.3±0.0	71.0±0.0	69.9±0.0
Ours (Online)		92.9±0.1	72.1±0.2	86.3±0.1	62.1±0.3	62.1±0.1		94.8±0.0	74.6±0.1	90.2±0.0	67.0±0.1	67.2±0.1
Ours (Offline)		**95.7**±0.0	**75.9**±0.1	**87.1**±0.1	**69.3**±0.0	**66.2**±0.0		**96.4**±0.0	**78.5**±0.1	**91.8**±0.0	**73.4**±0.1	**72.0**±0.1

Table 1. Top-1 Classification Accuracy (%) for five benchmarks with CLIP models. Higher is better. For ease of reading, we highlight the highest accuracy in **bold** and the second best as underline, which also applies to the following tables.

where y^*_{jk} is the k-th element of the j-th row of the matrix Y^*, and $\mathbb{1}.$ is the indicator function.

Similarly, SHOT [19] employs class-wise prototypes for K-means clustering, and AdaContrast [3] utilizes KNN to rectify the pseudo-labeling bias by averaging the probabilities of K-nearest neighbors. In contrast to SHOT which focuses on the global distribution in the latent space and KNN which concentrates on local similarity, our PLCA builds a local relation network for a sample and iteratively updates its pseudo-label in the global graph structure. We compare the effectiveness and computational speed for these three algorithms in Fig. 2 (b, c). Our PLCA achieves fewer pseudo-labeling errors and maintains a more stable speed in the latent space with different dimensions.

Remark. We treat the corrected pseudo-labels as targets for test-time training and adopt the same loss functions as the training stage, which scales our method to various tasks.

3.3. Test-Time Optimization

In this subsection, we explore a scalable and efficient optimization method for test-time adaptation, and this problem involves two key questions: *Which parameters need to be updated and how to update them?*

To analyze the impact of updating different parts of CLIP, we compare several common policies:

- BN Adapt [27] advices to combine the normalization statistics computed on the training dataset with the estimated ones on target samples, which is suitable for image corruption and has no extra training cost at test time.
- Tent [30] removes the statistics from the source data and updates the affine transformation parameters of BN layers, which is widely used in recent methods [22].
- Full-tune the model, which is the most expensive one.

We show the experimental results in Fig. 4 (a, b) and ensure that only the updated scopes are different. Because counting the mean and variance of target domains on a mini-batch leads to severe bias, especially when the batch size is small, it brings hard-to-recover errors to pseudo-labeling at the first few iterations. However, for Vision Transformer (ViT) [7] with LayerNorm, normalization statistics are counted for each sample, and it will not suffer from the above problem. Therefore, we freeze all the normalization layers and update other parameters to avoid such model corruption.

The choice of optimizer has a great impact on test-time adaptation [8]. We experiment with various learning rates for three representative optimizers, including stochastic gra-

Methods	SF	Gauss	Shot	Impul	Defcs	Gls	Mtn	Zm	Snw	Frst	Fg	Brt	Cnt	Els	Px	Jpg	Mean
Source	✓	28.8	22.9	26.2	9.5	20.6	10.6	9.3	14.2	15.3	17.5	7.6	20.9	14.7	41.3	14.7	18.3±0.00
TTT++ (Offline) [20]	✓	12.8	11.1	11.2	7.3	17.1	8.2	6.5	9.4	9.9	7.9	5.0	5.1	13.7	8.8	10.6	9.6±0.00
SHOT (Offline) [19]	✓	13.4	11.6	16.3	7.3	15.9	8.2	7.1	9.4	9.4	10.2	6.3	8.3	12.8	9.8	13.6	10.6±0.00
BN Adapt [27]	✓	15.9	13.7	18.0	7.8	18.3	8.9	8.0	10.8	9.6	12.7	6.1	9.4	13.5	14.3	14.5	12.1±0.01
TENT [30]	✓	14.5	12.4	17.7	7.7	17.7	8.8	7.9	10.3	9.6	12.0	6.1	9.0	13.4	11.3	14.5	11.5±0.02
MEMO [35]	✓	13.9	12.2	16.3	7.4	16.6	8.2	7.4	9.8	9.3	10.7	6.1	9.3	12.6	**10.0**	14.3	10.9±0.02
TTT++ (Online) [20]	✓	15.5	14.1	23.6	9.1	25.1	11.4	8.1	13.2	13.1	13.4	6.6	**6.9**	17.6	12.5	13.6	13.6±0.03
SHOT (Online) [19]	✓	14.5	12.3	17.7	7.8	17.8	8.7	7.9	10.4	9.6	12.1	6.1	9.0	13.4	11.4	14.4	11.5±0.02
Ours (Online)	✓	**12.8**	**11.4**	**14.9**	**6.7**	**15.8**	**7.7**	**6.9**	**8.9**	**8.6**	**10.1**	**5.6**	8.0	**11.9**	10.5	**12.8**	**10.2**±0.04

Table 2. Top-1 Classification Error (%) for all corruptions on CIFAR-10C (level 5). Lower is Better. SF denotes source free.

Methods	SF	Gauss	Shot	Impul	Defcs	Gls	Mtn	Zm	Snw	Frst	Fg	Brt	Cnt	Els	Px	Jpg	Mean
Source	✓	98.4	97.7	98.4	90.6	92.5	89.8	81.8	89.5	85.0	86.3	51.1	97.2	85.3	76.9	71.7	86.2±0.00
SHOT (Offline) [19]	✓	73.8	70.5	72.2	79.2	80.6	58.5	54.0	53.6	63.0	47.3	39.2	97.7	48.7	46.1	53.0	62.5±0.00
BN Adapt [27]	✓	87.1	90.6	89.5	87.6	93.4	80.0	71.9	70.6	81.5	65.9	46.8	89.8	73.5	63.2	67.5	77.3±0.30
TTT [29]	✓	73.7	71.4	73.1	**76.3**	93.4	71.3	66.6	64.4	81.3	52.4	41.7	**64.7**	55.7	52.2	55.7	66.3±0.00
MEMO [35]	✓	85.4	81.0	83.5	94.1	92.5	72.5	60.5	64.0	72.5	52.0	41.4	97.1	55.8	50.7	57.5	70.7±0.05
TENT [30]	✓	80.8	78.6	80.4	82.5	82.5	72.1	60.5	63.7	66.7	52.1	**39.2**	84.2	55.5	50.8	58.2	67.2±0.02
SHOT (Online) [19]	✓	83.9	82.3	83.7	83.9	83.8	72.6	61.9	65.7	68.6	54.8	39.4	85.9	58.1	53.1	62.3	69.3±0.03
Ours (Online)	✓	**72.9**	**70.8**	**73.1**	80.7	**79.7**	**69.6**	**57.4**	**59.8**	**63.1**	**50.0**	39.3	83.9	**51.8**	**48.5**	**50.8**	**63.4**±0.03

Table 3. Top-1 Classification Error (%) for all corruptions on ImageNet-C (level 5). Lower is Better. SF denotes source free.

Methods	Source	BN Adapt	TENT	MEMO	CoTTA	Ours
CIFAR-10C	18.3±0.0	13.8±0.4	12.1±0.7	10.9±0.3	11.5±0.1	**10.4**±0.1
ImageNet-C	86.2±0.0	74.9±0.4	70.7±0.6	79.8±1.0	67.8±0.3	**65.4**±0.1

Table 4. Continuous online adaptation following CoTTA [31]. We report the top-1 classification error (%). Lower is Better.

dient descent (SGD), SGD with momentum, and Adam [15]. Fig. 4 (c) exhibits the online adaptation process under the best learning rate. Optimizers with first/second-order momentum can adapt fast to the target distribution in most cases, but also cause fluctuations (e.g., SGD with momentum for RN50) and even continuing decline due to error accumulation (e.g., Adam for ViT-B/32). In contrast, vanilla SGD with a small learning rate can achieve stable and ascending performance. Since vanilla SGD can update the model parameters in the current optimal direction without being affected by historical gradients.

Avoiding catastrophic forgetting during long-time adaptation is a problem worth paying attention to [24, 31]. This issue is also important for maintaining the zero-shot robustness of CLIP. We propose the parameter moving average to preserve the knowledge from the source model. Specifically, at any iteration $t \geq 1$, we define the parameters after optimization as θ_t and update the moving average $\hat{\theta}_t$ as

$$\hat{\theta}_t = \pi \cdot \hat{\theta}_{t-1} + (1 - \pi) \cdot \theta_t, \tag{9}$$

where $\hat{\theta}_0$ is initialized as the parameters of source model θ_S and $\pi \in [0, 1]$ is the momentum coefficient, which determines the degree of knowledge preservation.

4. Experiment

4.1. Setup

Datasets. We evaluate on the following datasets:

- CIFAR10 [17] and CIFAR100 [17] are two datasets consisting of low-resolution images, which differ from the common size of the training data of CLIP. We use them to evaluate the adaptability of our method to the distribution shift from image distortion caused by resize operations.
- Food101 [1] and Stanford Cars [16] are fine-grained datasets contains 101 food categories and 196 car categories, respectively. We use them for fine-grained tasks.
- ImageNet [6] is a large-scale dataset with 1000 classes, which is suitable for evaluating our method on category-intensive and real-world classification tasks.
- CIFAR-10C [12] and ImageNet-C [12] are created by test images of CIFAR10 and ImageNet with 15 types of corruptions in 5 levels, which are widely used to evaluate the robustness of test-time adaptation methods. Following the common protocol [27, 29, 30], we evaluate our method on the 5-level (worst) corruptions of all 15 types.
- For object detection, we use KITTI [9] to evaluate our method for automotive applications. KITTI comprises 8 categories of traffic participants and is mostly recorded in clear weather conditions. We follow Mirza *et al.* [22] to simulate degrading weathers of fog, rain and snow.
- For semantic segmentation, we use CarlaTTA [21] to conduct experiments. CarlaTTA contains five gradually changing scenarios, including day2night, clear2fog, clear2rain, dynamic, and highway. It challengs the methods to adapt to distribution shift over time.

Implementation details. For test-time adaptation with

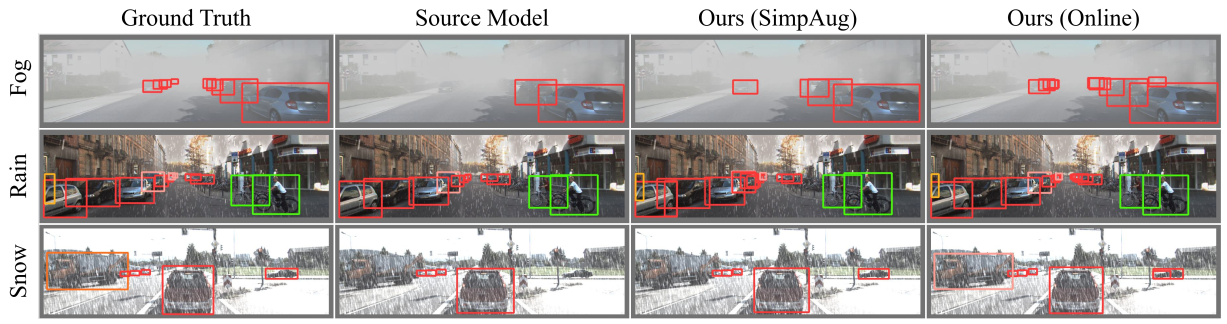


Figure 5. Visualization of the detection results on KITTI in the weather conditions of fog, rain, and snow.

Methods	SF	Fog	Rain	Snow
Source	✓	22.8±0.0	65.5±0.0	25.6±0.0
ActMAD [22]	✗	47.7±1.0	81.4±0.2	64.5±0.0
BN Adapt [27]	✓	32.3±0.1	70.6±0.1	44.9±0.2
TTT [29]	✓	29.4±0.5	71.5±0.4	47.5±0.7
Ours (SimpAug)	✓	27.3±0.4	75.2±0.4	36.0±0.3
Ours (Online)	✓	**43.2**±0.3	**81.6**±0.2	**52.3**±0.2

Table 5. Mean Average Precision (mAP@50) in KITTI with the most severe weather conditions, *i.e.* fog with only 30m visibility, rain with 200mm/hr intensity, and snow in level 5. Higher is better. SF denotes source free.

Methods	SF	day2night	clear2fog	clear2rain	dynamic	highway
Source	✓	58.4	52.8	71.8	46.6	28.7
BN Adapt [27]	✓	62.0	56.8	71.4	**52.6**	32.8
TENT [30]	✓	61.5	56.0	70.9	50.3	32.0
MEMO [35]	✓	61.0	55.1	71.6	50.3	**35.2**
CoTTA [31]	✓	61.4	56.8	70.7	46.4	33.8
Ours (SimpAug)	✓	60.1	53.9	72.6	48.9	31.9
Ours (Online)	✓	**62.4**	**56.8**	**73.9**	51.5	34.7

Table 6. Mean Intersection over Union (mIoU, %) on CarlaTTA. Higher is Better. SF denotes source free.

CLIP [25], we use the publicly available pretrained model as the source model, and evaluate our method on the test/validation sets of CIFAR10, CIFAR100, Food101, Cars, and ImageNet. We uniformly use “a photo of a {class}” as the prompt template, where the class token is replaced by the specific class name, such as ”airplane”, ”deer”, and ”frog”. Our method is run with a learning rate of 1e-6 for ResNet [11] backbone and 1e-5 for Vision Transformer [7] backbone. A batch size of 64 is adopted considering the memory constraint, while different batch sizes have little impact on our method, analyzed in the ablation study 4.4. Following Mirza *et al.* [22], we adapt a Wide-ResNet-40-2 [34] (WRN-40-2) pre-trained with AugMix [13] on CIFAR-10C, and an ImageNet pre-trained ResNet-18 [11] from PyTorch model zoo on ImageNet-C. We adopt a batch size of 128 and a learning rate of 1e-3, while all the compared methods and baselines use the hyperparameters reported by their authors. For object detection task, we adapt a YOLOv3 [26] pretrained on KITTI in clear weather to fog, rain and snow. For semantic segmentation task, we adapt a DeepLab-V2 [5] with a ResNet-101 backbone pre-trained on clear split of CarlaTTA to the other five scenarios. We run experiments on 5 random seeds and report the mean and standard deviation for these runs.

Offline and online adaptation. We evaluate our method in both offline and online settings. In the offline scenario, we can access all the test data and train the model on the entire test sets for 50 epochs, without considering the storage limitation. In the online scenario, the test samples arrive sequentially and do not reappear. We maintain a memory queue to store a certain amount of feature vectors, $10K$ by default, for our PLCA. We also study the effect of adopting different memory capacities on our method in Sec. 4.4. Furthermore, our method maintains the performance in the more challenging single sample online adaptation task.

4.2. Image Classification

Results for adapting CLIP model are reported in Table 1. We find that updating both image and text encoders simultaneously can further improve the adaptability of CLIP model than freezing the text encoder as shown in Fig. 4 (c). We also compare with the method of using SimpAug and PLCA without re-training. The number of augmented verions for each test sample is set to 8 by default, which contains an original version. PLCA outperforms the our online method on multiple benchmarks, which may be due to the fact that online training does not sufficiently learn from the test samples with pseudo-labels, while offline adaptation addresses this issue by training repeatedly and providing more accurate pseudo-labels after each epoch.

We present the results on CIFAR-10C and ImageNet-C in Table 2 and 3. Our method outperforms other online methods by a fair margin. Compared to works designed for offline adaptation, our method achieves comparable or even higher performance. It is noteworthy that MEMO [35] achieves higher performance than other baselines on adapting WRN-40-2, perhaps because of applying the same augmentation strategy, AugMix [13], for TTA as the training stage. However, when adapting ResNet-18 pre-trained with standard augmentation, MEMO does not maintain advantages due to the poor calibration of deep neural networks to augmented images with complex unseen transformations.

Models	Methods	ImageNet	-V2	-Sketch	-A	-R
RN-50	CLIP (zero-shot)	55.8	51.1	33.2	21.8	56.0
	Ours ($\pi = 0$)	57.2	53.1	33.8	22.5	59.2
	Ours ($\pi = 0.9$)	**58.3**	**53.8**	36.0	**24.0**	**60.3**
	Ours ($\pi = 0.99$)	57.4	52.6	**36.2**	24.0	59.1
RN-101	CLIP (zero-shot)	58.2	54.3	38.0	28.1	63.9
	Ours ($\pi = 0$)	59.7	55.9	39.8	27.0	66.9
	Ours ($\pi = 0.9$)	**60.1**	**56.2**	**40.9**	30.4	**67.3**
	Ours ($\pi = 0.99$)	59.3	55.4	40.8	**31.0**	66.6
ViT-B/32	CLIP (zero-shot)	59.8	54.7	40.9	30.2	66.7
	Ours ($\pi = 0$)	**63.7**	**58.1**	**44.1**	32.7	**71.4**
	Ours ($\pi = 0.9$)	62.1	57.1	43.6	**33.4**	70.5
	Ours ($\pi = 0.99$)	60.9	56.5	43.2	33.1	69.7
ViT-B/16	CLIP (zero-shot)	64.5	60.6	46.0	47.8	73.8
	Ours ($\pi = 0$)	**69.1**	**63.6**	**50.1**	51.4	**79.1**
	Ours ($\pi = 0.9$)	67.2	62.5	49.1	**52.9**	77.6
	Ours ($\pi = 0.99$)	65.9	61.8	48.5	52.0	76.6

Table 7. Comparison with different π values on robustness to distribution shift. Higher is Better.

Continuous online adaptation. Following CoTTA [31] to continuously adapt the model to different corruptions, we test our method on CIFAR-10C and ImageNet-C, referred to Table 4. Our method outperforms other baselines.

4.3. Object Detection and Semantic Segmentation

To evaluate the effectiveness of our method in solving regression problems, we conduct experiments on KITTI using YOLOv3. We report the results in Table 5 and visualize the detecting boxes in Fig. 5. Our method outperforms BN Adapt [27] and TTT [29], which are scalable to other models and not limited by the main task, and also beats ActMAD [22], which utilizes statistics in source data to TTA, in clear-to-rain task. Note that when only using SimpAug for test-time augmentation, we improve the source model up to 10.4%. The visualization shows that SimpAug helps the model recognize distant and occluded objects that are hard to detect, while online training with pseudo-labels further boosts the accuracy and filter out duplicate boxes.

For the semantic segmentation task, building a graph with all pixels is computationally expensive. We are inspired by COMUS [33] and collect object-level features, which represent the nodes in the graph. The results are listed in Table 6. BN Adapt exhibits robustness in the distribution shifts over time, while other methods seem to have limited improvement in gradually changing environments. Our method outperforms CoTTA [31], which is designed for continuous test-time adaptation.

More implementation details and comparison results can be found in the supplemental material.

4.4. Ablation Study

Number of augmentations. We study how the number of augmentation for single sample affects the performance of SimpAug, and conduct experiments on KITTI using YOLOv3, referred to Table 8. SimpAug improves the accuracy with the increase of augmentation numbers until the performance bottleneck. Large number of augmentation requires computational resources and storage.

Num	128	64	32	16	8	4	2	1 (Source)
Fog	**28.2**	27.7	27.3	26.4	25.9	25.0	24.6	22.8
Rain	75.9	**75.9**	75.2	73.8	73.4	72.1	68.9	65.5
Snow	**37.6**	37.0	36.0	34.6	32.8	30.6	28.7	25.6

Table 8. mAP@50 of SimpAug with different augmentation numbers on KITTI. Higher is Better.

Memory Size		100K	10K	1K	100	10	0	Source
RN50	CIFAR10	83.0	83.0	**83.4**	82.9	82.2	82.1	68.7
ViT-B/32	Food101	**86.4**	86.3	85.8	85.7	85.7	85.7	80.7

Table 9. Accuracy (%) of our online method with CLIP model in different sizes of memory queue. Higher is Better.

Batch Size		128	64	32	16	8	1	Source
RN50	CIFAR10	82.8	83.0	83.5	82.9	83.3	81.0	68.7
ViT-B/32	Food101	86.4	86.3	86.3	86.6	86.5	87.1	80.7

Table 10. Accuracy (%) of our online method with different batch sizes. Source denotes zero-shot performance of the CLIP model.

Size of memory queue. The memory size affects the inference speed and the accuracy of corrected pseudo-labels by PLCA. We conduct ablation experiments for online adaptation, as shown in Table 9. A large memory improves the ability of graph modeling the relations between samples. However, in online settings, old stored feature vectors have a higher error rate than the new ones, which affects the prediction of pseudo-labels for the current image.

Momentum coefficient. π is a momentum coefficient that affects the models' update speed and avoids catastrophic forgetting. We test the performance of the CLIP model without using PMA and with two π values. The results listed in Table 7 show that ViT backbone is more resistant to catastrophic forgetting than ResNet, and our method can maintain the robustness of the CLIP model to distribution shift while performing test-time adaptation.

Batch size. Most existing methods suffer from a small batch size in online adaptation, especially for those that rely on modifying normalization statistics of BN layers. As shown in Table 10, our method maintains comparable performance even in single sample online adaptation.

5. Conclusion

In this work, we propose an improved self-training method for fully test-time adaptation. In contrast to previous TTA methods, our IST explores the adaptability of the foundation model by employing CLIP as the source model. It goes beyond image classification and can be applied to object detection and semantic segmentation. IST outperforms existing TTA methods on different benchmarks and tasks by improving the quality of pseudo-labels and stabilizing the adaptation process. Experiments show that IST is scalable to a variety of pre-trained models and application tasks.

References

[1] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VI 13*, pages 446–461. Springer, 2014. 3, 6

[2] Collin Burns and Jacob Steinhardt. Limitations of post-hoc feature alignment for robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2525–2533, 2021. 1

[3] Dian Chen, Dequan Wang, Trevor Darrell, and Sayna Ebrahimi. Contrastive test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 295–305, 2022. 1, 2, 4, 5

[4] Liang Chen, Yong Zhang, Yibing Song, Ying Shan, and Lingqiao Liu. Improved test-time adaptation for domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24172–24182, 2023. 1, 2

[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017. 7

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 3, 6

[7] 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. *arXiv preprint arXiv:2010.11929*, 2020. 5, 7

[8] Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei Efros. Test-time training with masked autoencoders. *Advances in Neural Information Processing Systems*, 35:29374–29385, 2022. 1, 2, 5

[9] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 6

[10] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017. 1

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 7

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

[13] 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. *arXiv preprint arXiv:1912.02781*, 2019. 3, 7

[14] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Label propagation for deep semi-supervised learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5070–5079, 2019. 4

[15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[16] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561, 2013. 6

[17] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6

[18] Xianfeng Li, Weijie Chen, Di Xie, Shicai Yang, Peng Yuan, Shiliang Pu, and Yueting Zhuang. A free lunch for unsupervised domain adaptive object detection without source data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8474–8481, 2021. 1, 2

[19] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International conference on machine learning*, pages 6028–6039. PMLR, 2020. 1, 2, 4, 5, 6

[20] 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? *Advances in Neural Information Processing Systems*, 34: 21808–21820, 2021. 1, 2, 6

[21] Robert A Marsden, Mario Döbler, and Bin Yang. Introducing intermediate domains for effective self-training during test-time. *arXiv preprint arXiv:2208.07736*, 1(2):5, 2022. 6

[22] Muhammad Jehanzeb Mirza, Pol Jané Soneira, Wei Lin, Mateusz Kozinski, Horst Possegger, and Horst Bischof. Actmad: Activation matching to align distributions for test-time-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24152–24161, 2023. 1, 5, 6, 7, 8

[23] Alexander Neubeck and Luc Van Gool. Efficient non-maximum suppression. In *18th international conference on pattern recognition (ICPR'06)*, pages 850–855. IEEE, 2006. 3

[24] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *International conference on machine learning*, pages 16888–16905. PMLR, 2022. 1, 6

[25] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 2, 3, 4, 5, 7

[26] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 7

[27] Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving

robustness against common corruptions by covariate shift adaptation. *Advances in neural information processing systems*, 33:11539–11551, 2020. 1, 2, 5, 6, 7, 8

[28] Divya Shanmugam, Davis Blalock, Guha Balakrishnan, and John Guttag. Better aggregation in test-time augmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1214–1223, 2021. 1, 3

[29] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, pages 9229–9248. PMLR, 2020. 1, 2, 5, 6, 7, 8

[30] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020. 1, 2, 5, 6, 7

[31] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7201–7211, 2022. 1, 2, 6, 7, 8

[32] Longhui Yuan, Binhui Xie, and Shuang Li. Robust test-time adaptation in dynamic scenarios. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15922–15932, 2023. 1, 2

[33] Andrii Zadaianchuk, Matthaeus Kleindessner, Yi Zhu, Francesco Locatello, and Thomas Brox. Unsupervised semantic segmentation with self-supervised object-centric representations. *arXiv preprint arXiv:2207.05027*, 2022. 8

[34] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 7

[35] Marvin Zhang, Sergey Levine, and Chelsea Finn. Memo: Test time robustness via adaptation and augmentation. *Advances in Neural Information Processing Systems*, 35: 38629–38642, 2022. 1, 2, 3, 6, 7

[36] Hao Zhao, Yuejiang Liu, Alexandre Alahi, and Tao Lin. On pitfalls of test-time adaptation. *arXiv preprint arXiv:2306.03536*, 2023. 1, 2

[37] Dengyong Zhou, Olivier Bousquet, Thomas Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. *Advances in neural information processing systems*, 16, 2003. 4