JEI Journal of Electronic Imaging

RESEARCH PAPER

Test-time adaptation via self-training with future information

Xin Wen,[a] Hao Shen,[a] and Zhongqiu Zhao[a,b,c,*]
[a]Hefei University of Technology, School of Computer Science and Information Engineering, Hefei University of Technology, Hefei, China
[b]Hefei University of Technology, Intelligent Interconnected Systems Laboratory of Anhui Province, Hefei, China
[c]Guangxi Academy of Sciences, China

ABSTRACT. Test-time adaptation (TTA) aims to address potential differences in data distribution between the training and testing phases by modifying a pretrained model based on each specific test sample. This process is especially crucial for deep learning models, as they often encounter frequent changes in the testing environment. Currently, popular TTA methods rely primarily on pseudo-labels (PLs) as supervision signals and fine-tune the model through backpropagation. Consequently, the success of the model's adaptation depends directly on the quality of the PLs. High-quality PLs can enhance the model's performance, whereas low-quality ones may lead to poor adaptation results. Intuitively, if the PLs predicted by the model for a given sample remain consistent in both the current and future states, it suggests a higher confidence in that prediction. Using such consistent PLs as supervision signals can greatly benefit long-term adaptation. Nevertheless, this approach may induce overconfidence in the model's predictions. To counter this, we introduce a regularization term that penalizes overly confident predictions. Our proposed method is highly versatile and can be seamlessly integrated with various TTA strategies, making it immensely practical. We investigate different TTA methods on three widely used datasets (CIFAR10C, CIFAR100C, and ImageNetC) with different scenarios and show that our method achieves competitive or state-of-the-art accuracies on all of them.

© 2024 SPIE and IS&T [DOI: 10.1117/1.JEI.33.3.033012]

Keywords: domain adaptation; test-time adaptation; self-training; distribution shift; image classification

Paper 231261G received Nov. 9, 2023; revised Mar. 19, 2024; accepted Apr. 18, 2024; published May 13, 2024.

1 Introduction

Deep learning has made considerable progress in various visual tasks, such as semantic segmentation, video understanding,[1] image classification,[2] and image restoration.[3,4] Nonetheless, these successful implementations hinge on a fundamental presupposition: the data distribution of test samples must align with that of the training dataset.[5,6] In actual scenarios, this assumption does not always hold true because there are often distribution differences between test samples and training data, a phenomenon known as "domain shift." When confronted with domain shift, even subtle variations can lead to notable degradation in the performance of deep neural networks.[7,8] For instance, a model trained exclusively on sunny-day images may exhibit a substantial decrease in recognition accuracy when presented with environmental conditions such as rain, fog, snow, or nighttime. Due to privacy concerns or legal restrictions, source data are often deemed unavailable during inference, making this setting more challenging than unsupervised domain adaptation

*Address all correspondence to Zhongqiu Zhao, z.zhao@hfut.edu.cn

(UDA) but also more realistic. To tackle these issues, there is a growing need for approaches that can adapt models during test time, particularly in an online fashion. Test-time adaptation (TTA) methods emerge as a promising solution to enhance the performance of deep learning models in the face of domain shifts.

The categories of TTA approaches include batch normalization (BN), pseudo-label (PL), information entropy, and feature alignment. BN replaces the statistics inherited from the source dataset with those computed from the current batch of test data.[9] Recent research[10] has shown that the batch size significantly impacts the accuracy of BN adaptation. PL generates pseudo-labels for test data and uses them to fine-tune the model.[11] Nevertheless, label noise and confirmation bias can affect PL.[12] Information entropy fine-tunes the pretrained model by minimizing the prediction entropy of the test data.[13] Feature alignment, such as TTT+,[14] introduces a test-time feature-matching strategy that stores the mean and covariance matrix of the features during training and matches the current features with the previous ones during testing.

Although existing methods have demonstrated impressive performance, their performance is limited by the quality of PLs generated by pretrained models. Conventional self-training approaches use the classifier's predictions on test data as PLs and fine-tune the classifier to adapt to the PLs.[15] Even so, this approach is prone to error accumulation.[16] Since the model is updated based on the current test data, if the current PL predictions are inaccurate, the updates based on the incorrect PLs will affect the subsequent model adaptation process. In addition, conventional self-training strategies do not consider the reliability of the current PLs but merely filter out the data that satisfy certain conditions from the present data for updating.

To address these issues, we propose a self-training framework based on an intuitive assumption: if the current model's prediction on test data is consistent with the prediction of the model updated based on the future state of the same data, then we can consider the current PL to be accurate. Based on this supposition, we design an effective strategy to enhance the accuracy of PLs and reduce the error accumulation problem. On the other hand, we introduce a regularization term to penalize the confident output distribution, which increases the weight of misclassified categories to prevent the model from being overconfident in the prediction. Furthermore, to mitigate the forgetting problem, a fraction of the model weights are randomly reset to their initial values at each iteration to help preserve source knowledge in the long run.

The following are this paper's main contributions: (1) we suggest a technique for trustworthy sample selection using future PL information to increase the accuracy of PLs and reduce the issue of error accumulation. (2) We include a regularization term to punish overconfident output distributions for preventing the model from being overconfident about the output distribution. (3) We employ an approach of randomly recovering the model weights to address the forgetting problem of the model throughout the continual adaptation process, allowing the model to adjust to the changing environment. Through experiments in various scenarios, we confirm the effectiveness and benefits of our method.

2 Related Work

We divide the discussion on related works based on the different adaptation settings summarized in Table 1.

Table 1 Characteristics of problem settings that adapt a trained model to a potentially shifted test domain. "Offline" adaptation assumes access to the entire source or target dataset, whereas "online" adaptation can automatically predict a single or batch of incoming test samples. UDA, unsupervised domain adaptatio; TTT, test-time training; and TTA, test-time adaptation.

Setting	Source data	Target data	Training	Testing	Offline	Online	Source Acc
Fine-tune	No	x^t, y^t	$L(x^t, y^t)$	—	Yes	No	Not considered
UDA	x^s, y^s	x^t	$L(x^t, y^t) + L(x^s, x^t)$	—	Yes	No	Maintained
TTT	x^s, y^s	x^t	$L(x^t, y^t) + L(x^s)$	$L(x^t)$	Yes	No	Not considered
TTA	No	x^t	—	$L(x^t)$	No	Yes	Not considered
TTA (ours)	No	x^t	—	$L(x^t)$	No	Yes	Maintained

2.1 Unsupervised Domain Adaptation

UDA is a technique that uses a model trained on a data-rich and accurately labeled source domain to learn an unlabeled target domain by transferring knowledge from the source domain.[17] The most general idea of UDA is to learn a domain-invariant feature to align the target and source domains by reducing the difference between the domains. Two domains are aligned by choosing an appropriate difference measure function to mitigate the domain shift between the target and source domains. Rozantsev et al.[18] proposed a two-stream network by minimizing the parameter difference at each layer between the two networks, one on the source dataset and the other on the target dataset. Sun et al.[19] matched the second-order statistics of the source and target distributions using a linear transformation. Damodaran et al.[20] learned the joint distribution of the source and target domains using optimal transport. Another method uses adversarial loss to align the source and target domains, Saito et al.[21] proposed a weak global alignment model to reduce the difference between the source and target domains. The feature distribution[22] is considered at the local category level, and the categories are aligned utilizing an adaptive adversarial loss. The method[23] used entropy minimization for semantic segmentation tasks and presented an adversarial loss to align the source and target domains.

2.2 Source-Free Domain Adaptation

Unlike UDA, source-free domain adaptation requires a model trained on the source domain and an unlabeled target domain, not access to the source domain during testing. SHOT[24] froze the classifier of the model obtained by training on the source domain and implicitly aligned the representation of the target domain to the source hypothesis using information maximization and self-supervised pseudolabeling. TTT[25] proposed test-time training by transforming a single unlabeled test sample into a self-supervised learning problem that updates model parameters before making predictions. TENT[13] proposes for the first time to update model parameters using entropy minimization to update model parameters. MEMO[26] suggested a straightforward approach to use various data augmentations to evaluate the data and then reduce the entropy of the average or marginal output distribution of these different data augmentations. Qiu et al.[27] proposed a method that uses the information in the source model to generate source incarnation prototypes (i.e., representative features for each source class) that adapt the features generated for the test data to the source prototypes. Most existing methods use the input test data to update the source model to improve model performance, but Chen et al.[16] proposed to update the target data using a generative diffusion model to project all test inputs into the source domain, keeping the model parameters fixed during the testing phase.

2.3 Domain Generalization

Domain generalization aims to improve the performance of a model to unseen domains by learning the model over multiple visible source domains, thereby achieving out-of-distribution (OOD) generalization.[28] Domain alignment is a popular approach for domain generalization, and its core idea is also similar to the principle of domain alignment for UDA. Li et al.[29] used a maximum mean difference metric to align distributions across domains and match the aligned distributions to arbitrary prior distributions through adversarial feature learning. In machine learning, metalearning has recently received much attention. The concept of metalearning has been applied to domain generalization, Li et al.[30] proposed to sequentially learn a set of source domains and train at each step to maximize performance on the following domain.[31] MetaNorm is a simple and effective method for normalization using metalearning. Data augmentation has been one of the most common methods for training models in machine learning. Zhou et al.[32] used a data generator to generate new pseudodomains to expand the source data and then employed the obtained pseudodomains and the source data domains for training the model. Xu et al.[33] improved the model's generalization ability using random convolution as a data augmentation method.

3 Methodology

3.1 Problem Formulation

Assume that we have the training data and the corresponding probability distributions $D^s = \{(x^s, y^s)\}_{i=1}^{n} \sim P_s(x^s, y^s)$, where $x^s \in X$ is the input and $y^s \in \{1, 2, \dots, K\}$ is the target label; $f_{\theta_0}(x)$ denotes the model pretrained in the source domain, where θ_0 denotes the model parameter. Without loss of generality, we denote the test data and the corresponding probability distributions as $D^t = \{x^t\}_{j=1}^m \sim P_t$, where D_s and D_t share the output space, i.e., $\{y_i\} \sim D^s = \{y_j\} \sim D^t$. If the training dataset and the test dataset do not satisfy the assumption of the identical independent distribution, i.e., $P^s \neq P^t$. In this case, applying the classifiers trained only on the source data directly to the test data can become unreliable and lead to performance degradation.

3.2 Self-Training with Future Information

Most earlier research uses the small loss criterion to choose reliable samples while taking advantage of DNNs' memory. Recent research suggests choosing the correct PLs based on a threshold or confidence criterion[10,34] to lessen the detrimental effects of inaccurate PLs. In light of this, instead of blindly believing the PLs of all samples, we suggest double-checking the sample selection and only allowing samples where the forward and backward models diverge slightly. Test-time domain adaptation improves OOD performance by updating the source model with PLs produced by the source model. Despite this advantage, they lack a mechanism to check the reliability of the generated sample PLs.

Given the target data x_t^T and a pretrained model f_{θ_t} in the source domain, we use the most commonly used entropy loss as the test time objective function, following TENT and EATA:

$$L_t(x_t^T) = -\sum_c y_t' \log y_t'. \tag{1}$$

Similar to standard self-training, we use a classifier trained on the source dataset and get a PL from that classifier:

$$y' = \operatorname{argmax}\{f_{\theta_t}(x_t^T)\}. \tag{2}$$

Previous self-training methods choose trustworthy PLs using confidence thresholding or reweighting for each x in the target dataset.[35] PLs are selected with softmax values. Wang et al.[34] used weight-averaged and augmentation-averaged predictions to generate PLs that they believe are generally more accurate on average. MEMO[26] subjects a piece of data from the test dataset to different data augmentations and uses the average prediction obtained from various data augmentations as the PL. EATA[10] adds entropy reweighting to the prediction results to rely on samples with more confidence. The primary premise behind TAST's[36] proposal is that under domain offsets, it is likely that the test data and its closest neighbors in the embedding space will share the same label. Test-time domain adaptations update the source model using its generated PLs to enhance performance OOD. However, despite this advantage, they need a technique to verify the validity of the generated sample PLs.

We design a complementary step with the following insights to improve self-training. Intuitively, to solve the problem that the model only makes reliable sample selections for the current input data based on past model states, we propose a new selection strategy that allows the model to learn from its future information. Figure 1 shows a conceptual diagram of our approach. The prior technique obtains the PL of the input data at step t using the time step $t-1$ model and then uses this knowledge to choose a trustworthy sample to update the model. In addition, as opposed to the prior approach, we employ the $t+k$-step model and future label information to assess the reliability of the PL. The model weights at $t+k$ time step cannot be employed at the start of the time step t since they rely on the optimized model at t time step. To overcome this obstacle, we use a virtual update strategy.[37] First, the parameters of the model of t are stored, then the model of the $t+k$ time step is obtained through a virtual update, and then the pseudo label of the input data at time t is obtained. Finally, if the model differs slightly between some samples' PLs at time steps $t+k$ and t, depending on these samples, reliable samples can be created using a specific selection strategy. This time, we apply the gradient to the realistic weights of the pretrained model.

Since $t+k$ has no model weight at the time step t, a virtual update strategy is adopted, and we denote the virtual update model as $f_{\hat{\theta}_t}$. The model parameters of the virtual update at time $t+k$ are as follows:

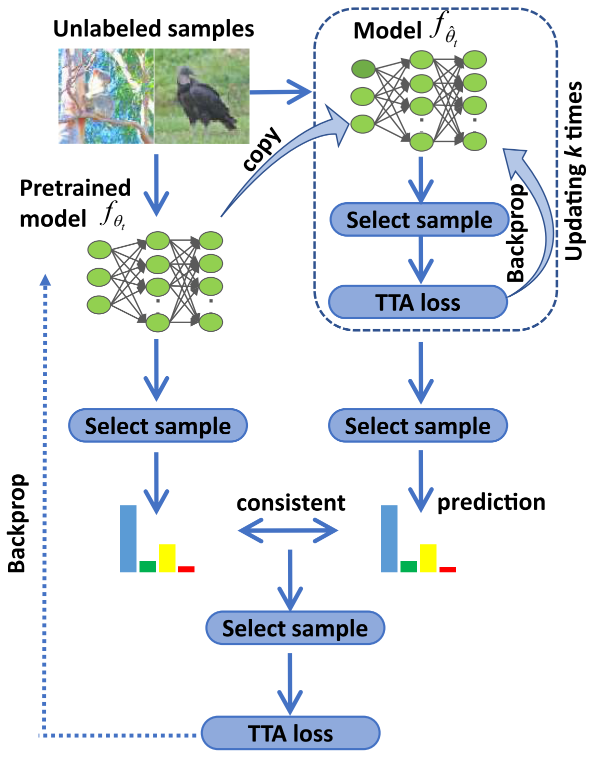

Fig. 1 An overview of the proposed method. Given an existing pretrained model f_{θ_t} for the source domain, we cache the model parameters $f_{\hat{\theta}_t}$ at that time. During adaptation, when a batch of target samples arrives, we use a sample selection strategy to obtain reliable samples, virtual update the $f_{\hat{\theta}_t}$ k times to obtain the model $f_{\hat{\theta}_{t+k}}$ and then predict PLs for this batch of samples. We compare the PL after virtual update k times with the prediction of f_{θ_t}, if consistent, we consider this sample more reliable, then use the sample selection strategy and calculate the actual update loss, and then do the real backpropagation update this time.

$$\hat{\theta}_{t+k} = \hat{\theta}_{t+k-1} - \alpha \nabla_{\hat{\theta}} L_{\hat{\theta}_{t+k-1}}, \tag{3}$$

where α is the learning rate, $k = \{1, 2, 3, \dots\}$ is the number of virtual updates. The final virtual model is used to create a new PL. The PL of this sample is likely to be reliable if the PL predicted by the model at time step t matches the outcome anticipated by the model after the k steps of virtual updating. Finally, an actual update is performed on these samples.

3.3 Confidence Penalization and Model Recovery

Deep neural networks are prone to produce overconfident predictions even though the predicted categories are far from the true ones.[38] Therefore, we use a regularization term L_p to penalize the confident output distribution and can mitigate the problem of model overconfidence:[39]

$$L_p = -\frac{1}{K}\delta(p_k) \cdot \log\ p_k, \tag{4}$$

where $\delta(p_k)$ is a coefficient that can be computed by

$$\delta(p_k) = \max\left(0, \tau - \frac{p_k}{p_y}\right), \tag{5}$$

where p denotes the output of the neural network, p_y denotes the element in p for the label y, and p_k denotes the k'th element of the output result. Here, τ is a hyper-parameter of the confidence threshold. After introducing the confidence penalty, the final optimization formula for our method can be formulated as

$$L = L_t(x_t) + \lambda L_p(f_\theta(x_t), y'), \tag{6}$$

where λ is a weighting hyper-parameter.

Although information from future models can be used to improve the quality of pseudo-labels, the problem of catastrophic forgetting will arise in the case of continual adaptation over a long period. If the model encounters a significant shift in the domain distribution, it will inevitably make incorrect predictions. If the model is updated on this basis, the problem of error accumulation will occur, affecting the future adaptability of the model. Hence, to avoid the problem of inevitably forgetting the model after long-term test domain adaptation, we also introduce a model recovery strategy.[34] By randomly resetting a small number of tensor elements in the trainable weights to their initial values, the network ensures moderate deviation from the initial source model during training, effectively circumventing the risk of catastrophic forgetting. This strategy preserves the model's stability and generalization capabilities by preventing excessive deviation from its original state while adapting to new domains:

$$M \sim \text{Bernoulli}(p), \tag{7}$$

$$W_{t+1} = M \odot W_0 + (1 - M) \odot W_{t+1} \tag{8}$$

where W_0 denotes the initial weight obtained from the pretraining model, W_t is the model weight to be updated after t time steps, and M represents a portion of the weight randomly selected from W_t to return to the initial value.

3.4 Composing with Prior Methods

Our sample selection framework is flexible as a plug-and-play method that can be integrated with different sample selections or sample weighting strategies, thereby exploiting each individual's performance improvements. With our approach, pretrained models already exist and do not need to be modified or retrained on the source data. Consequently, to investigate the effectiveness of the method described in this paper, we combine our proposed method with the widely used TTA models EATA, TENT, and PL.

4 Experiments

We will present our experiments' datasets, models, and implementation details. Following that, we will assess our method's performance in image classification on three benchmark tasks with continual test-time domain adaptation. In addition, we will evaluate our method's effectiveness in a real-world experimental setting, precisely a mixed-domain adaptation experiment (CIFAR10C, CIFAR100C, and ImageNetC). Finally, we will conduct ablation experiments to demonstrate the effectiveness of our method.

4.1 Benchmarks and Tasks

4.1.1 *Datasets and models*

We conducted experiments on three widely used benchmarks for OOD generalization: CIFAR10C, CIFAR100C, and ImageNetC.[40] Each dataset contains 15 types of corruption with 5 levels of severity. To assess the effectiveness of our approach, we compared it to several state-of-the-art techniques, such as PL, BN Stats Adapt, TENT, EATA, and CoTTA. PL[11] uses a pretrained classifier to predict the test data and select confident samples of PLs for the classifier to fine-tune. BN Stats Adapt[41] retains the weights of the source model but replaces the statistics of each BN layer in the source data with the statistics of the target domain of the current iteration. TENT[13] adapts the model by minimizing the entropy of model predictions and updates the normalization statistics and transformation parameters on test samples. EATA[10] selects reliable and nonredundant samples from the test samples for backpropagation, thus improving the efficiency of model adaptation. CoTTA[34] suggests reducing error accumulation using weight-averaged and augmentation-averaged predictions. SAR[42] propose a sharpness-aware and reliable entropy minimization method. We combine our method with PL, TENT, and EATA in this work.

4.1.2 *Evaluation metrics*

We measure the error rate and average error rate on 15 different types with 5 levels of severity on CIFAR10C, CIFAR100C, and ImageNetC datasets.

4.1.3 *Implementation details*

For CIFAR10-to-CIFAR10C, we followed the standard implementation of CoTTA, TENT, and EATA for the CIFAR10C experiments. We use the Adam optimizer with a learning rate of 1×10^{-3} and batch size of 200. The same pretrained model is adopted, which is a WideResNet-28[43] model from the RobustBench benchmark.[44] We have made some specific adjustments for different models and benchmark tests. In our method, we set the virtual update step k to 3, the hyperparameter τ of the regularization term to 0.2, the trade-off parameter λ to 0.1, and the probability parameter p to 0.01. For CIFAR100C, we use the pretrained model ResNeXt-29[45] from the RobustBench benchmark, and other hyperparameters are the same as those used for CIFAR10C. In the case of ImageNetC, we use SGD as the update rule with a momentum of 0.9, a batch size of 64, and a learning rate of 0.0025. The pretraining model we use is the standard pretrained ResNet50 model in RobustBench.[44] We set the virtual update step k to 1, the hyperparameter τ of the regularization term to 0.1, the tradeoff parameter λ to 0.1, and the probability parameter p to 0.001. For PL and TENT, the normalization statistics and transformation parameters are updated on test samples. The confidence threshold in PL is set to 0.4, which can give acceptable results in most cases. The hyperparameters for CoTTA are the same as in the original paper. To ensure optimal performance, EATA uses specific thresholds and parameters. The entropy constant threshold is set to $0.4 \times \ln K$ (K is the number of task classes), and the non-redundant threshold is set to 0.4/0.05 for CIFAR/ImageNet experiments. Additionally, for CIFAR10, the trade-off parameter is set to 1, and for CIFAR100 and ImageNet, the value is set to 2000.

4.2 Image Corruption

4.2.1 *Experiments on CIFAR10-to-CIFAR10C*

Table 2 shows the results for CIFAR10C. We compare our method with the source-only baseline and BN adapt, PL, TENT, CoTTA, and EATA. Directly applying the pretrained model to the target dataset with a domain shift from the source dataset, the error rate is higher than that after domain adaptation, indicating that domain adaptation is essential and effective. Our method, when combined with other domain adaptation methods, significantly improves accuracy. Specifically, we observed gains of 1.08% with PL, 2.49% with TENT, and 0.6% with EATA. The experimental results show that the integration of many approaches, including the one suggested in this work, does not produce a significant improvement and may even significantly underperform when compared to the baseline during the early stages of adaptation. However, the strategy described in this study shows a distinct advantage over the original methodology when the adaptation process moves into later stages. This benefit arises from its capacity to efficiently reduce mistake accumulation by utilizing the direction of upcoming model weights. Essentially, the suggested approach continually minimizes error propagation during the extended adaptation process, demonstrating its superiority step by step.

4.2.2 *Experiments on CIFAR100-to-CIFAR100C*

To further demonstrate the effectiveness of the proposed method, we evaluate it on the comprehensive CIFAR100-to-CIFAR100C task. The experimental results are collected in Table 3. TENT performs admirably for the initial few corruptions. However, under continual long-term adaptation, TENT exhibits a marked decline in the performance, which suggests that it experiences error accumulation and catastrophic forgetting. After combining TENT with our proposed method, the effect is significantly improved and even higher than the original EATA method. Our method produces more accurate labels by carefully choosing more trustworthy samples and comparing them with future models. Error accumulation is lessened by updating the model using

Table 2 Classification error rate (%) for the standard CIFAR10-to-CIFAR10C online continual TTA task. All results are evaluated on the WRN-28-10 architecture with the largest corruption severity level 5. The bold number indicates the best result.

Method	Source	BN Ada	CoTTA	SAR	PL	PL + ours	TENT	TENT + ours	EATA	EATA + ours
Gauss.	72.32	28.07	**24.25**	27.71	26.74	27.79	24.79	25.73	24.87	25.43
Shot	65.75	26.12	22.57	24.92	22.62	22.88	20.51	21.8	**19.65**	20.4
Impul.	72.93	36.26	**27.46**	34.07	31.47	31.15	28.53	30.88	27.75	29.34
Defoc.	46.99	12.81	12.09	12.65	13.13	12.68	14.43	12.06	12.87	**12.05**
Glass	54.33	35.28	**27.96**	34.88	33.23	31.54	31.38	31.25	29.56	30.60
Motion	34.76	14.17	**13.25**	13.90	15.16	14.66	16.02	13.51	15.22	13.34
Zoom	42.01	12.11	**10.81**	12.09	13.32	12.45	14.06	11.49	12.68	11.13
Snow	25.13	17.27	15.74	17.15	17.75	16.59	19.93	15.89	16.72	**15.30**
Frost	41.32	17.39	**14.78**	17.11	17.17	16.11	19.66	16.14	15.55	15.47
Fog	26.01	15.25	13.35	14.90	15.87	15.24	18.9	14.19	14.89	**13.06**
Brit.	9.30	8.38	8.10	8.39	9.38	9.43	11.84	8.31	10.11	**7.88**
Contr.	46.64	12.64	11.51	12.64	12.46	11.91	15.94	11.51	13.90	**11.03**
Elastic	26.6	23.76	**19.23**	23.56	23.67	20.77	25.79	21.95	21.94	22.05
Pixel	58.44	19.67	**13.85**	19.00	19.26	16.2	22.54	16.75	16.93	16.38
JPEG	30.28	27.30	**18.03**	26.28	26.01	21.71	27.13	22.52	21.80	22.22
Avg	43.52	20.43	**16.87**	19.95	19.82	18.74	20.76	18.27	18.30	17.71

Table 3 Classification error rate (%) for the standard CIFAR100-to-CIFAR100C online continual TTA task. All results are evaluated on the ResNeXt-29 architecture with the largest corruption severity level 5. The bold number indicates the best result.

Method	Source	BN Ada	CoTTA	SAR	PL	PL + ours	TENT	TENT + ours	EATA	EATA + ours
Gauss.	73	42.12	40.42	39.90	38.10	39.50	37.16	37.28	**37.03**	37.96
Shot	68	40.66	38.09	34.78	35.04	36.30	35.76	34.58	34.21	**34.09**
Impul.	39.35	42.72	40.25	36.63	38.05	37.36	41.64	35.51	35.89	**33.72**
Defoc.	29.32	27.64	27.41	26.17	31.01	27.76	37.81	27.06	29.23	**25.61**
Glass	54.11	41.82	38.35	37.12	40.58	39.07	51.12	38.01	39.66	**36.55**
Motion	30.82	29.73	28.48	28.19	33.76	29.9	48.45	29.54	32.41	**28.18**
Zoom	28.75	27.87	26.79	25.92	32.21	27.72	49.20	26.84	29.92	**25.72**
Snow	39.49	34.9	33.6	31.8	38.13	32.13	59.38	31.72	34.97	**30.3**
Frost	46.81	35.01	32.48	30.46	40.46	31.84	66.17	31.79	35.47	**29.95**
Fog	50.28	41.51	40.71	35.68	47.29	35.94	73.58	34.75	38.46	**32.36**
Brit.	29.53	26.52	25.32	25.24	41.55	25.77	73.87	25.91	30.49	**24.24**
Contr.	55.10	30.31	27.46	27.97	52.38	29.23	84.31	28.96	33.30	**26.82**
Elastic	37.23	35.67	32.93	**31.75**	55.07	35.24	88.99	35.41	37.52	32.61
Pixel	74.68	32.94	**28.39**	29.15	54.56	30.89	90.97	32.09	33.75	28.47
JPEG	41.23	41.16	**33.91**	37.22	64.35	38.18	94.2	41.05	43.12	36.29
Avg	46.45	35.37	32.97	31.87	42.84	33.15	62.17	32.70	35.03	**30.86**

these chosen examples. The current batch of data frequently influences later data in typical model updating processes, which causes a slow accumulation of mistakes. The influence on subsequent target data during weight updating will, however, be greatly minimized if more precise samples and PLs can be acquired during the updating phase, increasing the model's prediction accuracy. In conclusion, our approach improves model prediction performance by streamlining the sample selection and label generation procedures, increasing the accuracy of model updates. Overall, on the CIFAR100C dataset, our proposed method improved the PL by 9.69% and the TENT by 29.47%, while the EATA enhanced by 4.17% after being combined with our method.

4.2.3 *Experiments on ImageNet-to-ImageNetC*

Compared to CIFAR10C and CIFAR100C, adaptation to the ImageNet-C dataset is considerably more challenging, and even the previous methods suffer from severe performance degradation. The experimental results are summarized in Table 4. As can be seen, the proposed method consistently improves the previous adaptation approaches, PL (gain 6.36%) and TENT (gain 31.23%). Due to the pretrained model's abysmal performance on severe corruptions, evident from the source method's extremely low accuracy, our approach does not perform as well on the ImageNetC dataset as on CIFAR10C and CIFAR100C. Even though our strategy can reduce the accumulation of errors, it inevitably leads to incorrect predictions when the confidence in the model itself is relatively low. Although their labels are expected to be the same before and after, they still obtain inaccurate results, and the results do not improve.

4.3 Mixed Domain Adaptation on Corruption Benchmark

In actuality, it is more likely that a batch of entering test samples will have various domain shift distributions than they will all have the same distribution as in the prior continual domain

Table 4 Classification accuracy (%) for the standard ImageNet-to-ImageNetC online continual TTA task. All results are evaluated on the ResNet-50 architecture with the largest corruption severity level 5.

Method	Source	BN Ada	CoTTA	SAR	PL	PL + ours	TENT	TENT + ours	EATA	EATA + ours
Gauss.	0.11	14.56	16.12	25.70	22.30	20.52	27.27	26.53	33.63	32.85
Shot	0.11	5.30	19.51	34.48	30.06	24.83	32.59	32.43	39.28	37.89
Impul.	0.12	15.14	21.92	34.74	30.98	25.39	29.52	31.49	38.25	36.41
Defoc.	0.16	14.40	19.51	25.60	22.00	21.86	16.32	24.85	32.84	32.25
Glass	0.14	14.76	20.96	29.95	22.74	22.62	7.53	23.75	34.14	32.97
Motion	0.15	25.66	30.94	36.30	29.31	34.61	3.34	33.47	45.32	45.18
Zoom	0.21	37.68	40.60	44.67	35.87	44.41	1.76	44.78	51.64	51.58
Snow	0.11	33.41	32.70	37.92	29.74	38.80	0.90	40.60	49.20	49.16
Frost	0.15	32.36	32.52	38.56	29.67	37.52	0.60	37.20	44.64	44.52
Fog	0.35	46.37	45.25	48.37	36.28	51.64	0.64	52.81	58.11	57.86
Brit.	0.15	64.53	56.91	61.76	49.27	65.54	0.83	64.64	66.38	66.88
Contr.	7.09	15.07	21.75	33.34	22.35	26.15	0.40	26.56	43.31	42.25
Elastic	0.11	42.77	42.45	50.81	33.54	47.43	0.61	48.77	55.91	56.06
Pixel	0.13	47.75	46.07	54.34	37.3	52.87	0.63	54.72	58.90	58.80
JPEG	0.10	38.73	39.95	50.37	32.55	45.43	0.61	50.49	54.32	54.08
Avg	0.61	30.57	32.48	40.46	30.94	37.30	8.24	39.47	47.06	46.70

Table 5 We conduct mixed domain adaptation on CIFAR10C, CIFAR100C, and ImageNetC datasets. Average accuracy (%) mixed domain with severity level 5 and level 3.

	CIFAR10C		CIFAR100C		ImageNetC	
Method	Level 5	Level 3	Level 5	Level 3	Level 5	Level 3
Source	37.41	43.64	19.19	20.42	0.61	1.47
BN Ada	65.97	78.96	52.52	64.24	17.66	38.67
CoTTA	57.7	78.42	28.64	47.10	17.76	44.93
SAR	68.78	80.61	45.87	63.02	20.63	50.91
PL	67.35	81.25	53.24	**68.69**	4.2	50.03
PL + ours	66.10	79.02	53.49	64.73	7.91	45
TENT	41.35	68.77	6.38	16.22	2.06	39.33
TENT + ours	64.28	80.93	31.36	58.93	3.74	47.64
EATA	**71.97**	**83.48**	30.50	50.44	26.70	**51.5**
EATA + ours	67.71	80.09	**55.10**	67.10	**30.30**	50.48

adaptation experiments, which were all focused on the same domain distribution shift problem. We use different corruption types to simulate mixed domain adaptation experiments with multiple domain distribution shifts. We use the CIFAR10C, CIFAR100C, and ImageNetC, the batch size is 64, and other parameters are the same as in previous experiments. We evaluate different methods on a mixture of 15 corruption types at different severity levels (5 and 3). As shown in Table 5, we observe that the results of mixed domain adaptation on the three datasets are much worse than those of a single domain adaptation, particularly on ImageNetC, where many methods fail. In the mixed domain adaption context, these strategies are prone to failure since they update the model based on the current output and forecast the next batch of data distribution. Due to continual domain adaptation, the model may suffer from strong distribution shifts, leading to incorrect PL predictions. In this case, makes self-training exacerbates wrong predictions. The model might not recover after encountering hard-to-adaptation examples, even with new data having no significant domain shift.

4.4 Ablation Study

We conduct an ablation study to see how important each part of our approach is. The results are listed in Table 6. Our strategy has three parts: a self-training method, an overconfidence penalty function, and model restoration. The hyperparameters were the same as our previous work on the CIFAR100-to-CIFAR100C task. We have evaluated each of them based on PL and found that our method significantly reduces the error rate compared to PL. Using self-training with future information reduces the error rate on CIFAR100C from 42.84% to 39.67%, showing that our method can improve PL accuracy and reduce error accumulation. We further improve the accuracy with regularization, achieving a lower error rate of 36.13% on CIFAR100C. Although our method improves the quality of the pseudo-labels and minimizes the error rate, the neural network may still make wrong predictions in some cases where there is much noise. In this case, random weight restoration of pretrained models can help. We reduce the error rate of PL on the CIFAR100C dataset from 36.13% to 33.15%.

Using the self-training framework improves PL accuracy, but over-reliance on these trustworthy PLs can cause overfitting of the model. We include a regularization term in order to address this problem. In the meantime, the model's performance on source data may gradually deteriorate as it gradually adjusts to target data. This is due to the possibility of a growing bias in the model parameters throughout the continuous update process, which ignores data from

Table 6 Effects of components in proposed method (the top is the result of PL). We use the CIFAR100C dataset with ResNeXt-29 as the backbone network to obtain error rate (%) results for 15 corruption cases with a severity level of 5.

Future	Loss	Recover	Gauss.	Shot	Impul.	Defoc.	Glass	Motion	Zoom	Snow
			38.10	35.04	38.05	31.01	40.58	33.76	32.21	38.13
✓			37.82	34.14	37.58	30.48	41.33	33.37	31.66	37.27
✓	✓		39.79	35.81	38.24	29.92	41.46	33.45	31.58	35.96
✓	✓	✓	39.50	36.30	37.36	27.76	39.07	29.90	27.72	32.13
			Frost	Fog	Brit.	Contr.	Elastic	Pixel	JPEG	Avg
			40.46	47.29	41.55	52.38	55.07	54.56	64.35	42.84
✓			39.45	43.59	36.31	41.93	47	46.60	56.50	39.67
✓	✓		35.56	39.96	31.65	33.59	38.59	34.51	41.92	36.13
✓	✓	✓	31.84	35.94	25.77	29.23	35.24	30.89	38.18	33.15

the source domain in favor of the target domain. We use a straightforward model parameter restoration technique to preserve the accuracy of the model on the source data domain. Using this method, the model can adapt to the target domain and maintain its performance on the source domain, striking a balance between the two. Our self-training methodology prevents overfitting and source domain performance degradation while achieving significant improvements in PL accuracy by combining regularization and model parameter restoration techniques.

4.4.1 *Effect of the serial exploration steps k*

k is the number of virtual updates. The experimental results are summarized in Table 7. We combined our approaches with TENT and EATA methods. We performed ablation experiments

Table 7 Effect of the serial exploration steps k. We tested the TENT and EATA methods on the CIFAR10C and CIFAR100C datasets and calculated the average error rate (%) under different k values. The triangles indicate how much performance has been improved (how much the error rate has been reduced) compared to the original method.

	CIFAR10C			CIFAR100C		
Method	k	Avg error	Δ	k	Avg error	Δ
TENT	—	27.59	—	—	42.84	—
TENT + ours	1	18.92	8.67	1	33.17	9.67
TENT + ours	2	19.08	8.51	2	32.96	9.88
TENT + ours	3	18.84	8.75	3	32.79	10.05
TENT + ours	4	19.15	8.44	4	34.24	8.6
EATA	—	20.21	—	—	41.93	—
EATA + ours	1	18.33	1.88	1	31.55	10.38
EATA + ours	2	18.48	1.73	2	31.5	10.43
EATA + ours	3	18.4	1.81	3	31.66	10.27
EATA + ours	4	18.5	1.71	4	31.83	10.1

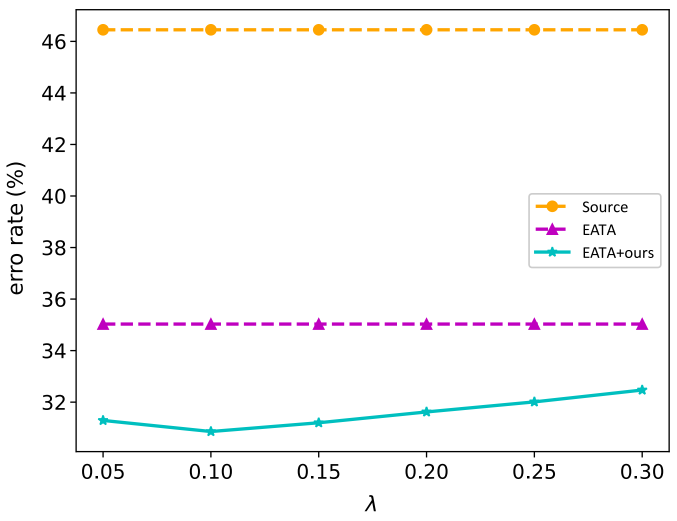

Fig. 2 Effect of the λ. We performed an experiment using the CIFAR100C dataset, set the severity level to 5, and used the same settings as the continual CIFAR100C experiment to get the average error rate. According to experimental findings, our technique can produce superior results in various λ values.

on one of the essential parameters, k. We separately tested the effect of different k values on different methods. We find that when $k = 3$, our method can achieve significant performance improvement on both datasets, while the performance improvement becomes insignificant when the value of k continues to increase. Table 7 shows that $k = 3$ is an appropriate choice to balance the robustness and efficiency of the model.

4.4.2 *Effect of the λ*

We evaluate our proposed method with different λ, chosen from $[0.05, 0.3]$. Figure 2 provides the results of ablation experiments on the CIFAR100C dataset. This figure shows that λ has a slight impact on the experimental results; too large or too small will hinder the performance of the model. Its size causes the loss to increase, more high-entropy samples are generated, and more samples are to be filtered out, resulting in a decline in generalization performance. Therefore, we choose to set λ to 0.1.

4.5 More Discussions

4.5.1 *Impact of batch size*

We conducted an evaluation of various TTA methods across different batch sizes, and the experimental outcomes are presented in Tables 8 and 9. The results indicate that the performance of PL, EATA, and TENT is affected by the batch size to varying degrees, with their performance declining considerably as the batch size decreases. Since these three methods update the parameters on the BN layer, the BN layer is vulnerable to batch size, and a small number of samples cannot accurately estimate the statistical information. Therefore, these methods are inferior to those without domain adaptation at small batch sizes. COTTA to update all parameters is more robust than the BN method. Our approach enhances the performance of EATA, PL, and TENT by combining it with the previous methods, proving its robustness to changes in test batch size. When facing the challenges of small batch sizes, we can adopt several strategies to enhance the effectiveness of BN, such as accumulating sufficient samples to form larger batches before updating and skillfully incorporating statistical information from both the source domain and test batches. However, this remains a topic worthy of further exploration. In the future, we will continue to dedicate ourselves to researching relevant strategies and optimizing and refining BN techniques for small batch sizes.

Table 8 Average error rates (%) for different batch sizes on the CIFAR10 to CIFAR10C online continual TTA task. All results are based on the WRN-28-10 architecture with the largest severity of five corruptions.

Method	200	64	32	16	8	4	Avg
Source	43.52	43.52	43.52	43.52	43.52	43.52	43.52
BN Ada	20.43	20.92	21.69	22.93	26.37	31.61	23.99
CoTTA	16.87	17.65	22.18	32.25	60.85	92.19	40.33
SAR	19.95	19.55	19.88	20.88	24.59	35.72	23.43
PL	19.82	26.28	41.74	59.61	78.67	86.98	52.18
PL + ours	18.74	20.24	22.46	23.02	26.08	31.6	23.69
TENT	20.76	27.59	39.48	63.85	79.53	87.39	53.1
TENT + ours	18.27	18.84	20.03	21.8	25.7	31.47	22.68
EATA	18.3	20.21	23.29	36.37	41.65	61.35	33.52
EATA + ours	17.71	18.4	19.4	21.31	25.4	31.19	**22.23**

Table 9 Average error rates (%) for different batch sizes on the CIFAR100 to CIFAR100C online continual TTA task. All results are based on the ResNeXt-29 architecture with the largest severity 5 corruption.

Method	200	64	32	16	8	4	Avg
Source	46.45	46.45	46.45	46.45	46.45	46.45	46.45
BN Ada	35.37	36.22	37.28	39.46	44.07	49.69	40.34
CoTTA	32.97	35.73	38.42	47.41	66.78	92.19	52.25
SAR	31.87	34.51	50.87	66.22	83.30	75.28	57.01
PL	42.84	80.75	87.88	93.94	96.69	98.29	83.4
PL + ours	33.15	34.27	35.56	37.57	42.5	61.27	40.72
TENT	62.17	84.6	92.75	95.11	97.83	98.66	88.51
TENT + ours	32.7	32.79	34.24	37.11	46.58	89.33	45.45
EATA	35.03	41.93	76.68	89.38	94.8	97.04	72.47
EATA + ours	30.86	31.66	33.18	35.83	41.83	54.11	**37.91**

4.5.2 *Comparisons under different severity levels on CIFAR100C*

To evaluate the performance of our method on the continual TTA CIFAR100-to-CIFAR100C experimental task, we give not only the highest severity results but also the results of different severity levels. In this section, we discuss the impact of our proposed method on the CIFAR100C dataset at various severity levels. Figure 3 shows that the TENT's prediction error rate gradually increases when it encounters varying degrees of distribution shift. This indicates that it has suffered from error accumulation due to continual TTA and the severe consequences of catastrophic forgetting. Figure 4 demonstrates that the EATA performs better than the TENT at different severity levels by reducing the forgetting problem with the fisher regularizer and improving the quality of PLs. Our solution, based on EATA, also incorporates future information to improve the correctness of PLs and resolves the forgetting problem by randomly restoring the model weight, allowing it to perform at its best over the full range of severity levels.

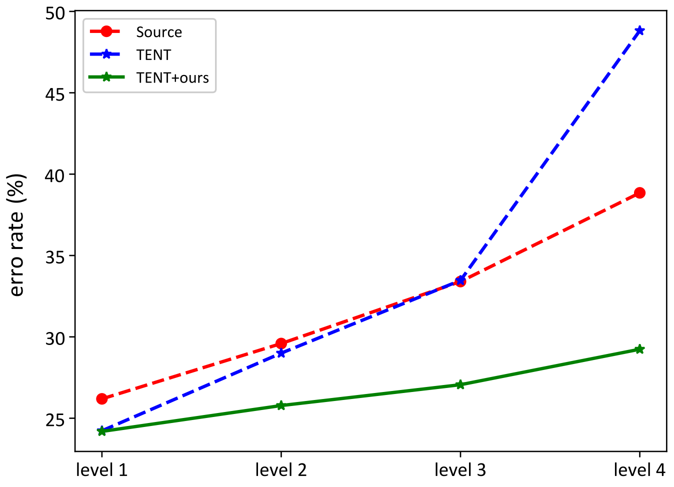

Fig. 3 Combine our method with TENT and compare their average error rates on severity levels 1 to 4 on the CIFAR100C dataset.

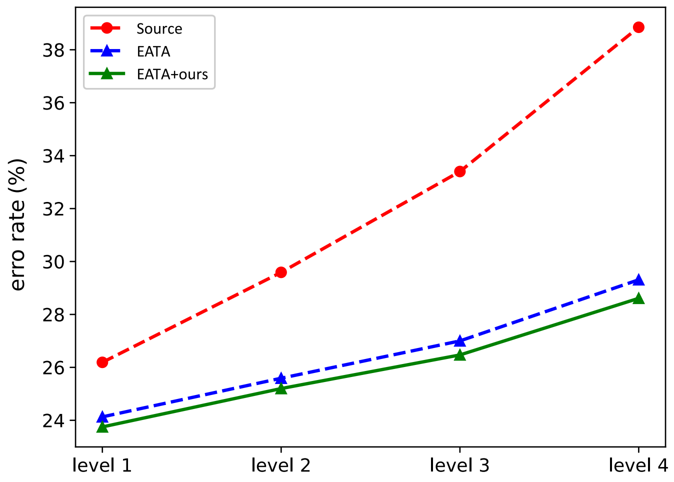

Fig. 4 Combine our method with EATA and compare their average error rates on severity levels 1 to 4 on the CIFAR100C dataset.

4.5.3 *Cost analysis of model training*

The cost analysis of model training is presented in Table 10. We utilize the pretrained ResNeXt-29 model from the RobustBench benchmark and employ the CIFAR100C dataset for our experiments. The model parameter we use is $k = 3$. It is noteworthy that the computational complexity, measured in terms of FLOPs, scales linearly with the value of k. Additionally, storing the original pretrained model and performing virtual updates necessitates a considerable amount of storage space.

4.5.4 *Demonstration of preventing forgetting*

In this part, we delve into the capability of our proposed method to maintain accuracy on the source dataset during TTA. The experiments are conducted on the CIFAR100C dataset. We specifically look at the model's performance changes before and after adapting to new data, particularly the accuracy on the original CIFAR100 validation set. This allows us to assess how effectively the model retains the knowledge from the source data while incorporating new information. The experimental procedure is as follows: first, we adapt the model to an OOD dataset. Subsequently, we evaluate the accuracy of the adjusted model when handling uncontaminated (i.e., "clean") data. This process is repeated five times consecutively, without resetting the model's parameters, to mimic continual learning scenarios in real-world applications.

Table 10 Cost analysis of model training.

Method	Flops (M)	Param. (M)	Acc
Source	1085.7	6.9	53.55
CoTTA	37999.5	20.7	67.03
PL	1085.7	6.9	57.16
PL + ours	5428.5	6.9	66.85
TENT	1085.7	6.9	37.83
TENT + ours	5428.5	6.9	67.3
EATA	1085.7	6.9	34.97
EATA + ours	5428.5	6.9	69.14

Table 11 Online continual TTA task. To comprehensively evaluate the long-term adaptation performance of the model, we are conducting a series of five consecutive assessments. In each assessment, the model experiences an adaptation process to both OOD data and the clean dataset. We meticulously record the model's accuracy (%) on the corrupted test sets as well as its accuracy (%) on the in-distribution (ID) clean datasets following OOD adaptation, providing a thorough analysis of its performance throughout the continual adaptation process.

Round	1		2		3		4		5	
Method	OOD	Clean	OOD	Clean	OOD	Clean	OOD	Clean	OOD	Clean
Source	53.55	78.9	53.55	78.9	53.55	78.9	53.55	78.9	53.55	78.9
PL	57.16	41.76	12.1	4.02	1.92	1.73	1.42	1.23	1.31	1.54
PL + ours	66.52	75.33	66.57	75.91	66.75	74.98	66.7	75.49	66.79	75.4
EATA	65.1	70.98	58.18	62.54	47.52	51.85	39.43	42.07	22.51	12.82
EATA + ours	68.61	75.63	67.2	75.8	67.2	75.34	67.28	75.79	67.24	75.57

The experimental results from Table 11 show that our method significantly outperforms PL and EATA in terms of accuracy when dealing with OOD data. More importantly, even after adapting to OOD data, the model's accuracy on the source dataset remains comparable to that of the initial model, providing compelling evidence of the effectiveness of our approach. Specifically, after the first round of adaptation, the accuracy of the PL method on both clean and corrupted data rapidly declines (until reaching 0%), indicating poor stability in lifelong adaptation. In contrast, while the EATA method maintains some level of accuracy in the first two rounds, its performance also deteriorates significantly as the adaptation process progresses. Our method, on the other hand, demonstrates consistent stability throughout the entire lifelong adaptation process, achieving high accuracy on corrupted data while maintaining accuracy on clean data close to that of the model without any OOD adaptation (i.e., the original accuracy). These findings strongly suggest the superiority of our method in overcoming the forgetting of source data.

5 Conclusion

In this paper, we propose a self-training framework that can effectively boost the performance of pretrained models even when there is a domain shift. Although previous methods are effective, their performance is limited by the quality of the PLs generated by the pretrained model and prone to error accumulation. Our proposed approach addresses this issue by improving the quality of the PLs through consistency in the current and future states and by adding a regularization term to penalize confident output distributions. Our method is practical and can be combined with various TTA methods (TENT, EATA, and PL), and the results demonstrate the versatility of our approach across different datasets and scenarios.

Disclosures

This article has no conflicts of interest.

Code and Data Availability

The data presented in this article will be publicly available in GitHub at https://github.com/tuwangwenx/ttt-fl The archived version of the code described in this manuscript will be freely accessed through https://github.com/tuwangwenx/ttt-fl After the paper is accepted, we will publish them uniformly.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 61976079), the Anhui Key Research and Development Program (Grant No. 202004a05020039), the Anhui High-Level Talents Program (Grant No. T000642), the National Key R&D Program of China (Grant No. 2018AAA0100100), the Science and Technology Innovation 2030 Major Projects of China (Grant No. 2021Z D0201904), the Key Project of Science and Technology of Guangxi (Grant Nos. AA22068057 and 2021AB20147), the Natural Science Foundation of Guangxi (Grant Nos. 2021JJA170204 and 2021JJA170199), and Guangxi Science and Technology Base and Talents Special Project (Grant Nos. 2021AC19354 and 2021AC19394).

References

1. D. Guo et al., "DadNet: dilated-attention-deformable ConvNet for crowd counting," in *Proc. 27th ACM Int. Conf. Multimedia*, pp. 1823–1832 (2019).
2. K. He et al., "Deep residual learning for image recognition," in *Comput. Vision and Pattern Recognit. (CVPR)*, pp. 770–778 (2016).
3. X. Cong et al., "Discrete haze level dehazing network," in *Proc. 28th ACM Int. Conf. Multimedia*, pp. 1828–1836 (2020).
4. H. Shen, Z.-Q. Zhao, and W. Zhang, "Adaptive dynamic filtering network for image denoising," in *Proc. AAAI Conf. Artif. Intell.*, Vol. 37, No. 2, 2227–2235 (2023).
5. C. Sakaridis, D. Dai, and L. Van Gool, "ACDC: the adverse conditions dataset with correspondences for semantic driving scene understanding," in *Comput. Vision and Pattern Recognit. (CVPR)*, pp. 10765–10775 (2021).
6. Y. Chen et al., "Domain adaptive faster R-CNN for object detection in the wild," in *Comput. Vision and Pattern Recognit. (CVPR)*, pp. 3339–3348 (2018).
7. M. J. Mirza et al., "An efficient domain-incremental learning approach to drive in all weather conditions," in *Comput. Vision and Pattern Recognit. (CVPR)*, pp. 3001–3011 (2022).
8. Y. Zou et al., "Unsupervised domain adaptation for semantic segmentation via class-balanced self-training," in *Eur. Conf. Comput. Vision (ECCV)*, pp. 289–305 (2018).
9. M. J. Mirza et al., "The norm must go on: dynamic unsupervised domain adaptation by normalization," in *Comput. Vision and Pattern Recognit. (CVPR)*, pp. 14765–14775 (2022).
10. S. Niu et al., "Efficient test-time model adaptation without forgetting," in *ICML*, PMLR, pp. 16888–16905 (2022).
11. D.-H. Lee et al., "Pseudo-label: the simple and efficient semi-supervised learning method for deep neural networks," in *ICML Workshop*, Vol. 3, p. 896 (2013).
12. C. Guo et al., "On calibration of modern neural networks," in *ICML*, PMLR, pp. 1321–1330 (2017).
13. D. Wang et al., "Tent: fully test-time adaptation by entropy minimization," in *International Conference on Learning Representations* (2021).
14. Y. Liu et al., "Ttt++: when does self-supervised test-time training fail or thrive?," in *NeuIPS 34*, pp. 21808–21820 (2021).
15. Y. Iwasawa and Y. Matsuo, "Test-time classifier adjustment module for model-agnostic domain generalization," in *NeuIPS 34*, pp. 2427–2440 (2021).
16. C. Chen et al., "Progressive feature alignment for unsupervised domain adaptation," in *Comput. Vision and Pattern Recognit. (CVPR)*, pp. 627–636 (2019).

17. S. J. Pan et al., "Domain adaptation via transfer component analysis," *IEEE Trans. Neural Networks* **22**(2), 199–210 (2010).
18. A. Rozantsev, M. Salzmann, and P. Fua, "Beyond sharing weights for deep domain adaptation," *IEEE Trans. Pattern Anal. Mach. Intell.* **41**(4), 801–814 (2018).
19. B. Sun, J. Feng, and K. Saenko, "Return of frustratingly easy domain adaptation," in *AAAI*, Vol. 30 (2016).
20. B. B. Damodaran et al., "DeepJDOT: deep joint distribution optimal transport for unsupervised domain adaptation," in *Eur. Conf. Comput. Vision (ECCV)* (2018).
21. K. Saito et al., "Strong-weak distribution alignment for adaptive object detection," in *Comput. Vision and Pattern Recognit. (CVPR)* (2019).
22. Y. Luo et al., "Taking a closer look at domain shift: category-level adversaries for semantics consistent domain adaptation," in *Comput. Vision and Pattern Recognit. (CVPR)* (2019).
23. T.-H. Vu et al., "Advent: adversarial entropy minimization for domain adaptation in semantic segmentation," in *Comput. Vision and Pattern Recognit. (CVPR)* (2019).
24. Y. Li et al., "Deep domain generalization via conditional invariant adversarial networks," *Lect. Notes Comput. Sci.* **11219**, pp. 624–639 (2018).
25. Y. Sun et al., "Test-time training with self-supervision for generalization under distribution shifts," in *ICML*, PMLR, pp. 9229–9248 (2020).
26. M. Zhang, S. Levine, and C. Finn, "Memo: test time robustness via adaptation and augmentation," in *NeuIPS 35*, pp. 38629–38642 (2022).
27. Z. Qiu et al., "Source-free domain adaptation via avatar prototype generation and adaptation," in *Int. Joint Conf. Artif. Intell.* (2021).
28. K. Zhou et al., "Domain generalization: a survey," *IEEE Trans. Pattern Anal. Mach. Intell.* **45**, 4396–4415 (2022).
29. H. Li et al., "Domain generalization with adversarial feature learning," in *Comput. Vision and Pattern Recognit. (CVPR)*, pp. 5400–5409 (2018).
30. D. Li et al., "Sequential learning for domain generalization," *Lect. Notes Comput. Sci.* **12535**, 603–619 (2021).
31. Y. Du et al., "Metanorm: learning to normalize few-shot batches across domains," in *ICLR* (2021).
32. K. Zhou et al., "Learning to generate novel domains for domain generalization," *Lect. Notes Comput. Sci.* **12361**, 561–578 (2020).
33. Z. Xu et al., "Robust and generalizable visual representation learning via random convolutions," in *Int. Conf. Learn. Represent.* (2020).
34. Q. Wang et al., "Continual test-time domain adaptation," in *Comput. Vision and Pattern Recognit. (CVPR)*, pp. 7201–7211 (2022).
35. K. Sohn et al., "Fixmatch: simplifying semi-supervised learning with consistency and confidence," in *NeuIPS 33*, pp. 596–608 (2020).
36. M. Jang and S.-Y. Chung, "Test-time adaptation via self-training with nearest neighbor information," in *Int. Conf. Learn. Represent.* (2023).
37. Y. Gu et al., "Not just selection, but exploration: online class-incremental continual learning via dual view consistency," in *Comput. Vision and Pattern Recognit. (CVPR)*, pp. 7442–7451 (2022).
38. A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: high confidence predictions for unrecognizable images," in *Comput. Vision and Pattern Recognit. (CVPR)*, pp. 427–436 (2015).
39. Q. Wei et al., "Self-filtering: a noise-aware sample selection for label noise with confidence penalization," *Lect. Notes Comput. Sci.* **13690**, 516–532 (2022).
40. D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," in *ICLR* (2019).
41. S. Schneider et al., "Improving robustness against common corruptions by covariate shift adaptation," in *NeuIPS 33*, pp. 11539–11551 (2020).
42. S. Niu et al., "Towards stable test-time adaptation in dynamic wild world," in *Int. Conf. Learn. Represent.* (2023).
43. S. Zagoruyko and N. Komodakis, "Wide residual networks," in *BMVC*, British Machine Vision Association (2016).
44. F. Croce et al., "Robustbench: a standardized adversarial robustness benchmark," in NeuIPS Datasets and Benchmarks Track (2021).
45. S. Xie et al., "Aggregated residual transformations for deep neural networks," in *Comput. Vision and Pattern Recognit. (CVPR)*, pp. 1492–1500 (2017).

Xin Wen received her BS degree from Neijiang Normal University, China, in 2020. She is pursuing her master's degree at Hefei University of Technology. Her research interests include computer vision and machine learning.

Hao Shen is currently pursuing his PhD. His current research interests include image processing, computer vision, and deep learning.

Zhongqiu Zhao received his PhD in pattern recognition and intelligent system from the University of Science and Technology of China, Hefei, China, in 2007. From 2013 to 2014, he was a research fellow in image processing in the Department of Computer Science, Hong Kong Baptist University, Hong Kong. He is currently a professor at Hefei University of Technology, Hefei, China. His current research interests include pattern recognition, image processing, and computer vision.