DomainAdaptor: A Novel Approach to Test-time Adaptation

Jian Zhang[1,2]   Lei Qi[3,*]   Yinghuan Shi[1,2,*]   Yang Gao[1,2]

[1] State Key Laboratory for Novel Software Technology, Nanjing University
[2] National Institute of Healthcare Data Science, Nanjing University
[3] School of Computer Science and Engineering, Southeast University

zhangjian7369@smail.nju.edu.cn, qilei@seu.edu.cn, syh@nju.edu.cn, gaoy@nju.edu.cn

Abstract

To deal with the domain shift between training and test samples, current methods have primarily focused on learning generalizable features during training and ignore the specificity of unseen samples that are also critical during the test. In this paper, we investigate a more challenging task that aims to adapt a trained CNN model to unseen domains during the test. To maximumly mine the information in the test data, we propose a unified method called DomainAdaptor for the test-time adaptation, which consists of an AdaMixBN module and a Generalized Entropy Minimization (GEM) loss. Specifically, AdaMixBN addresses the domain shift by adaptively fusing training and test statistics in the normalization layer via a dynamic mixture coefficient and a statistic transformation operation. To further enhance the adaptation ability of AdaMixBN, we design a GEM loss that extends the Entropy Minimization loss to better exploit the information in the test data. Extensive experiments show that DomainAdaptor consistently outperforms the state-of-the-art methods on four benchmarks. Furthermore, our method brings more remarkable improvement against existing methods on the few-data unseen domain. The code is available at https://github.com/koncle/DomainAdaptor.

1. Introduction

To overcome the domain shift (*i.e.*, training source and test target data come from distinct domains, for deep learning models), previous studies (*e.g.*, unsupervised domain adaptation [32] or domain generalization [30]) have mainly focused on designing sophisticated models in the training stage. Despite their efforts, when a large domain gap exists in the test stage, they still inevitably suffer drastic performance degeneration. Given that abundant information exists in the unlabeled unseen data during the test, which is ne-

---

*Corresponding authors: Yinghuan Shi and Lei Qi.
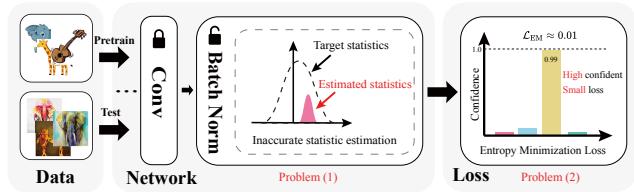
Figure 1: Illustration of the problem of (1) inaccurate statistics estimation in BN and (2) small loss produced by entropy minimization for the test-time adaptation methods.

glected when solely considering generalization in the training phase, a more practical approach is to adapt a trained model to the unlabeled unseen data by incorporating this information during the test (*i.e.*, test-time adaptation [43]).

Incorporating both source data and unlabeled target data can improve adaptation and enable a model to handle unseen domains in real-world scenarios. However, this approach is infeasible in practice due to the high computational cost of processing these data during the test. Besides, data privacy is paramount in many real-world scenarios where only model weights are accessible (*e.g.*, clinical data [19, 41] or commercial data [43]). These restrictions remind us to consider a more practical problem: adapt a trained CNN model to an arbitrarily unseen domain in the test time without access to the source data, namely fully test-time adaptation [43], to expand the generalization ability of a trained model without bearing retraining costs.

Several methods [37, 43, 2] have been developed recently for fully test-time adaptation. One of the widely employed adaptation paradigm [43, 31, 10, 1] to exploit the information of unseen samples is to finetune the Batch Normalization [16] layers in a trained model with an unsupervised loss, which is both simple and computationally efficient. However, most of these methods only show their effectiveness on artificially corrupted datasets (*e.g.*, CIFAR-10-C [14]), which is different from the real-world scene with diverse cross-domain styles in the domain generalization task. For instance, when we adapt the model trained by

photo data to the art painting data using existing methods, there still exist the following two issues, as shown in Fig. 1:

(1) The success of current test-time adaptation methods relies on an accurate estimation of the normalization statistics, which is hard to achieve by solely employing the test statistics obtained from the limited unseen data with a large domain gap. We argue that the source statistics can help the estimation, which is neglected by previous methods [43].

(2) During adaptation, previous unsupervised losses [3] (*e.g.*, Entropy Minimization (EM) loss) tend to bias the training procedure to the samples that have low confidence by producing large gradients and overlook the highly confident samples that also can help the adaptation procedure.

Considering the above limitations of current methods, we present the DomainAdaptor, a novel approach for adapting a pre-trained CNN model to unseen domains, which comprises an AdaMixBN module and a Generalized Entropy Minimization (GEM) loss. The AdaMixBN module overcomes inaccurate estimation of test statistics by combining training and test statistics and adapting the mixture coefficient based on the current batch. However, directly finetuning AdaMixBN may lead to performance degradation due to the weight mismatch problem caused by the combined source statistics after finetuning. To address this issue, we transform the source statistics into affine parameters in the normalization layers before finetuning. This not only maintains the effectiveness of AdaMixBN but also eliminates the negative impact of mismatched source statistics. Moreover, the traditional Entropy Minimization (EM) loss is not effective for finetuning AdaMixBN due to the sharp probability distribution predicted by the model for confident samples. Hence, we propose a GEM loss that emphasizes the role of temperature scaling in the traditional EM loss. GEM loss softens the probability distribution of each sample with temperature, generating large gradients for confident samples and encouraging further learning.

The contributions of our proposed DomainAdaptor can be summarized as follows:

- We propose AdaMixBN to adaptively mix the training and test stats. in the transformed normalization layer, which can trade off the training and test information.
- To better exploit unlabeled test samples, we propose the Generalized Entropy Minimization loss to effectively optimize the parameters of AdaMixBN.
- Our proposed method exhibits significant improvement over existing approaches on four benchmark datasets for domain generalization.

2. Related Work

Test time adaptation (TTA) [37, 10, 47, 17] is proposed to learn the test distribution by leveraging unlabeled test images, which provide hints about distribution information. Test-time training [37] employs a manually designed self-supervised learning task to learn the test distribution, which requires altering the training stage and finetuning all the layers. To mitigate this issue, Tent [43] is proposed by only finetuning the batch normalization layers with an unsupervised entropy minimization loss. Following works try different unsupervised losses to help test time adaptation, such as consistency loss [10], contrastive loss [29] or log-likelihood ratio loss [31]. However, when applied to the test data with a large domain gap, these methods commonly fail due to the inaccurate estimation of statistics and produce small gradients. In contrast, our method can alleviate these issues and also succeed in the few-data scenarios.

Domain generalization (DG) has attracted significant attention recently for its ability to generalize to unseen domains by only learning from source domains [30, 14]. To achieve the generalization ability, current methods primarily aim to learn invariant features across all domains [54, 24, 5, 4], augment data [25, 55, 57, 8, 48, 13] to learn diverse features or regularize network with training schemes or losses [22, 23, 27, 3, 51]. While these methods only consider the training stage, several methods alter model behaviors according to the test samples for better adaptation to the unseen domains. Instance Normalization [40] and AdaBN [26, 50] are simple but effective modules that utilize test statistics to perform normalization. Du *et al.* [7] generates accurate statistics for each test sample with a trained statistics prediction network. In addition to normalization, Pandey *et al.* [35] train a generative network to generate the nearest neighbor for each test sample in the source latent space. ARM [52] applies a meta-learning training scheme to extract batch-specific features during training and test for adaptation. Despite their efforts to adapt the model at the test stage, they require modifying the training stage and cannot be applied to an already trained model. In contrast, we propose our DomainAdaptor, which can be employed in any trained model, making it more practical in the real world.

Temperature scaling has been studied in different fields. In knowledge distillation [15, 28, 46, 53], it is used to soften the probability distribution over classes, providing additional class relationship information for the student model. In confidence calibration [11, 6, 38, 18], the temperature is finetuned as a parameter to ensure the predicted class confidence accurately reflects the likelihood of its ground truth correctness. In self-supervised learning [44, 49, 20], the temperature is used in contrastive loss to penalize hard negative samples. Differently, in this work, we employ temperature to encourage effective learning by fully exploiting unlabeled test data.

3. Our Method

Let $\mathcal{X}, \mathcal{Y}$ denote the set of images and their corresponding label. In the test stage, we are presented with a trained model $f(\cdot|\theta) : \mathcal{X} \rightarrow \mathcal{Y}$ parametrized by $\theta \in \Theta$. At each
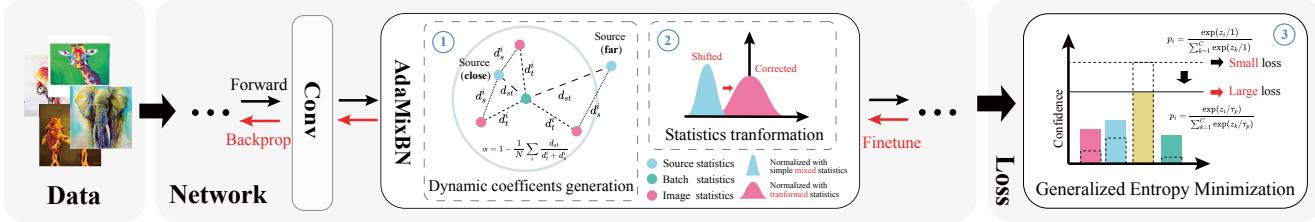
Figure 2: **Method Overview**. With an unlabeled test batch, **(1)** our proposed AdaMixBN first obtains the dynamic mixture coefficient α with Eq. (2). **(2)** Then, the affine parameters (γ, β) are transformed with Eq. (5) to avoid negative side-effect of mixed statistics. **(3)** Finally, we finetune the transformed affine parameters with Generalized Entropy Minimization (GEM) loss to fully exploit batch information. The labels of the same batch are predicted with the finetuned network.

timestep t, a batch of data points $\{(x_1^t, y_1^t), \ldots, (x_N^t, y_N^t)\}$ is sampled from the unseen joint distribution $\mathcal{P}_{\mathcal{X}\mathcal{Y}}$ and only the unlabeled batch of images $x^t = \{x_1^t, \ldots, x_N^t\}$ are observed, where N is the batch size. For simplicity, we drop the time t in the following paragraphs. Our goal is to predict the labels of the batch correctly by adapting the parameters θ of the trained model to the test distribution with only unlabeled data. To achieve this goal, we propose AdaMixBN with statistics transformation operation and Generalized Entropy Minimization loss to effectively exploit the information inside the unlabeled data. The whole framework is illustrated in Fig. 2.

3.1. AdaMixBN

3.1.1 Dynamic coefficient generation

Given a batch of data, despite the lack of label information, the statistics (*i.e.*, mean and variance) of the extracted feature within the unlabeled data can also provide clues about the underlying distribution of the data, which can be utilized to help the model adapt to the corresponding domain. Since the difference between the source statistics accumulated during training and batch statistics during the test causes domain shift in the Batch Normalization layers [33, 10], previous methods (*e.g.*, Tent [43]) only employ the test batch statistics for normalization and drop the source statistics, which is inaccurate for the real-world data and causes performance degradation. Considering that higher layers with high-level information are more transferable than lower layers (the experiment is conducted in Sec. 4.3), the source statistics are also useful for adaptation. Therefore, we propose AdaMixBN that can dynamically fuse both statistics to obtain a more accurate statistics estimation during the test:

$$\hat{x} = \frac{x - (\alpha\mu_s + (1-\alpha)\mu_t)}{\sqrt{(\alpha\sigma_s^2 + (1-\alpha)\sigma_t^2)}}\gamma + \beta, \qquad (1)$$

where (μ_s, σ_s^2) and (μ_t, σ_t^2) are the statistics estimated from the source and batch data, and (γ, β) are the affine parameters in BN. The layer index is omitted for simplicity.

Table 1: The average distances between the single test image, test batch, and the source statistics in the network.

Statistics Distance	Art	Cartoon	Photo	Sketch	Avg.
img2batch (Min)	0.016	0.013	0.013	0.000	0.011
img2batch (Max)	3.334	5.418	2.772	0.061	2.896
src2batch	1.031	0.269	1.148	3.251	1.425

Instead of simply employing a manually defined fixed mixture coefficient α which cannot handle the varying unseen domains, we design a novel generation module to dynamically mix the training and test statistics according to the Euclidean distance $d = \|\mu_1 - \mu_2\|_2 + \|\sigma_1 - \sigma_2\|_2$ of the statistics between source and test statistics:

$$\alpha = 1 - \frac{1}{N}\sum_i \frac{d_{st}}{d_t^i + d_s^i}. \qquad (2)$$

There are three distances in the formula: d_{st} is the distance between source statistics and test statistics; d_s^i and d_t^i are the distances of the statistics of a single image x_i to the source and batch statistics, respectively. Although a single distance d_{st} can be employed to calculate α, a fixed threshold is required to determine its relationship to the value of α. For instance, α can be obtained as $\alpha = \text{sigmoid}(d_{st} - \text{threshold})$, where the threshold needs to be manually set for each batch, which is impossible. We observe that the statistics of the images in the batch should not be far away from the overall batch statistics. If the source-to-batch distance d_{st} is smaller than the image-to-batch distance d_t^i, this implies that the source domain is similar to the test domain, and a large α can be used to incorporate more source statistics, as shown in Fig. 2. For example, in the above Tab. 1, the larger source-to-batch distance than the image-to-batch distance indicates a large domain gap for the Sketch domain. To reduce this domain gap, more test statistics should be incorporated (*i.e.*, a small α). Therefore, instead of manually designing α or adopting a threshold, we propose to employ the single image statistics as a proxy to dynamically measure the distance. We introduce the image-level distances d_s^i and d_t^i as a relative distance measure that adapts to differ-

Table 2: The test accuracy (%) before/after fientuning (FT) with different statistics. T is statistics transformation.

	Test	Mixed	Source	Mixed w/ T
Before FT	80.53	83.43	79.94	83.43
After FT	81.96	65.01↓	17.36↓	85.04

ent batches and layers. When the source and target statistics differ significantly, d_{st} and d_t^i would be large while d_s^i keeps the same. Then the ratio would be large and result in a small α, and vice versa. We average the ratio to obtain the final α.

3.1.2 Statistics transformation

Although AdaMixBN can improve the test-time performance, directly finetuning the network with AdaMixBN leads to significant degradation in performance. As shown in Tab. 2, when fintuning with only test statistics, the performance is improved, while when incorporating more source statistics in the normalization process, the performance degrades significantly. To the extreme, if we only employ source statistics without using test statistics, the performance cannot even defeat the random guess.

This problem arises due to the weight mismatch between the finetuned parameters (*i.e.*, γ, β in all BN layers) and source statistics, as illustrated in the middle row of Fig. 3. After finetuning, BN weights (*e.g.*, γ_0 and β_0 in Fig. 3) are changed, which outputs the feature map x_1' with the shifted distribution. However, the source statistics (*e.g.*, μ_{s1} and σ_{s1}^2 in Fig. 3) are fixed across the finetuning process. Then if we continue to utilize the source statistics that are only suitable for the original distribution to normalize feature map x_1' that has shifted distribution, the performance inevitably degrades. The more source statistics involved, the more serious the degradation. However, the performance of using dynamic test statistics does not suffer from this degradation. Therefore, *to both keep the better performance of mixed statistics while eliminating the side-effect of source statistics*, we propose to incorporate the source statistics *in an implicit way* by transforming the normalization formulation of AdaMixBN to match the formulation that only incorporates the test statistics. To achieve this goal, we can transform the normalization process of AdaMixBN in Eq. (1) to be independent of the source statistics (*i.e.*, μ_s and σ_s). With several simple deductions, we obtain the following transformation (details are in Supplementary Material):

$$\hat{x} = \frac{x - \mu_t}{\sigma_t}\gamma' + \beta', \qquad (3)$$

$$\gamma' = \frac{\sigma_t}{\sqrt{\alpha\sigma_s^2 + (1-\alpha)\sigma_t^2}}\gamma + \beta, \qquad (4)$$

$$\beta' = \frac{\alpha(\mu_t - \mu_s)}{\sigma_t}\gamma' + \beta. \qquad (5)$$
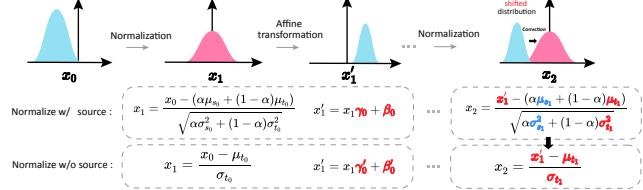

Figure 3: The normalization process after finetuning with only test (w/o source) or mixed statistics (w/ source).

This transformation can be viewed as re-initializing γ and β using the source and test statistics. In this way, the normalization process only depends on the test statistics μ_t and σ_t, which can dynamically change according to the distribution of the current feature map. Note that, the operation is done after the batch has been fed into the network and *before* the finetuning process. As a result, we finetune γ' and β' in each layer instead of the initial affine parameters. Furthermore, since we have altered the normalization process, the training and test mode of BN can not affect it.

3.2. Generalized Entropy Minimization

3.2.1 GEM framework

After feeding an unseen batch into the model and transforming its affine parameters for test-time adaptation with AdaMixBN, we can adapt the model with its predictions. However, we can not perform supervised finetuning as in few-shot learning [9] since the label is unavailable. Instead, we can consider using several unsupervised losses with human prior (*e.g.*, rotation prediction task [37] or jigsaw task [3, 34]). Different from these losses that require manually defined labels, Entropy Minimization (EM) loss is a simple but effective unsupervised loss that learning from data by encouraging the model to be confident [43]:

$$\mathcal{L}_{\text{EM}} = -\sum_{i=1}^{C} p_i \log p_i, \;\; p_i = \frac{\exp(z_i/\tau_p)}{\sum_{k=1}^{C}\exp(z_k/\tau_p)}, \quad (6)$$

where z_i is the logits predicted by the model, C is the number of classes and $\tau_p = 1$ is the temperature of the EM loss.

During the finetuning of a network with EM loss, highly confident samples that have a high probability of predicted classes, produce small contributions for weight updating as illustrated in the left figure of Fig. 4, resulting in little improvement on the trained model. We observe that highly confident samples have a sharp logit distribution as demonstrated in the right figure in Fig. 4. To reduce the sharp distribution, we propose to use temperature scaling for two reasons. First, it does not change the model's prediction, which can preserve the original semantic information. Second, for highly confident samples, we can obtain a larger loss by softening the logit distribution to enable fast learning. Meanwhile, the smoothness effect of temperature scal-
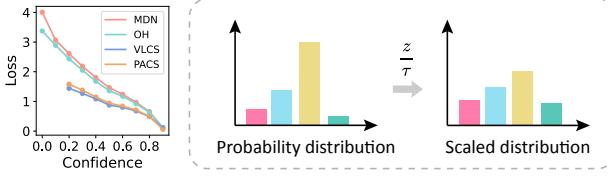
Figure 4: Loss decreases when confidence increases (left). Probability changes after temperature scaling (right).

ing decays when a sample has lower confidence, and the model also can learn from the low-confident samples. To this end, we propose a novel Generalized Entropy Minimization (GEM) framework with temperature scaling to adapt the trained model to the test unlabeled data:

$$\mathcal{L}_{\text{GEM}} = -\tau_q^2 \sum_{i=1}^{C} p_i \log q_i, q_i = \frac{\exp\left(z_i/\tau_q\right)}{\sum_{k=1}^{C} \exp\left(z_k/\tau_q\right)}, \quad (7)$$

where τ_q is the temperature for q_i, and we set $\tau_q \geq \tau_p \geq 1$ to soften the logits distribution. Note that we multiply GEM loss by τ_q^2 to ensure normal gradient magnitudes since the temperature scaling downscales not only the logits to $1/\tau_q$, but also the gradients to $1/\tau_q^2$ [15]. Different from traditional EM loss that employs the same probability in Eq. (6), we utilize two different probabilities as a general version of the cross-entropy loss, which imposes two different behaviors of GEM loss as analyzed in the following section.

3.2.2 Gradient analysis of GEM and GEM variants

For the logit z_k of class k, the gradient from GEM loss is (details can be found in Supplementary Material):

$$\frac{\partial L_{\text{GEM}}}{\partial z_k} = -\tau_q^2 \big[\frac{1}{\tau_q} \left(p_k - q_k \right)$$
$$+ \frac{p_k}{\tau_p} \big(\log q_k - \sum_j p_j \log q_j \big) \big]. \quad (8)$$

There are two terms in the right hand. If the gradient of p_i is stopped in Eq. (7) (similar to the one-hot label in the supervised cross-entropy loss), only the first term in Eq. (8) is left, which actually takes the same form as the knowledge distillation loss employed in the *teacher-student* network [15], where p_k can be viewed as the probability output from the teacher model and q_k from the student model. Optimizing with the first term could encourage the model to output the distribution similar to p. Therefore, the first term produces a Self-Knowledge Distillation (SKD) gradient that learns from itself. By increasing the temperature of q, the discrepancy between p_k and q_k would be large, and a larger gradient is produced. Besides, if $\tau_q < \tau_p$, the distribution of q is sharper than p. The model will learn from a softer

distribution and perform poorer. Thus, the τ_p should always be greater or equal to τ_q for better performance. When $p_k = q_k$, the first term is zero, and only the second term is left in Eq. (8), which plays the same role in the original entropy minimization loss that sharpens the distribution. Note that neither of these two terms can change the model's prediction when only a single sample is involved since they can only produce sharper logits distribution without changing its prediction. Instead, we can learn diverse features and improve the performance by *learning from a batch*.

Under the GEM framework, when $\tau_p = \tau_q = 1$, GEM loss is the same as the vanilla EM loss. We notice that p_i can be viewed as a weighting factor for GEM loss. If p_i is a one-hot tensor, only one term is left after summation. It is actually the same as the traditional cross-entropy loss that encourages the model to learn from a single class. But when p_i becomes smoother (*i.e.*, τ_p is larger), it encourages the model to learn from all classes. Thus, we propose several variants by setting different τ_p to produce different p_i. For τ_q, we set it to $\tau_q = \tau_a = \frac{s}{N} \sum_{i=1}^{N} \sigma_{z_i}$, where s is a hyper-parameter to adjust the scaling strength, which follows previous work [12] that uses the standard deviation of logits σ_{z_i} as a dynamic temperature to scale the logits.

GEM-T. We set $\tau_p = \tau_a$. The first term in Eq. (8) becomes 0, and both probabilities are scaled smoother to produce a large gradient. In this way, since all classes in the distribution are scaled and no distribution of samples is too sharp, the model is easier to make different predictions.

GEM-SKD. We set $\tau_p = 1$ and the gradient of p_i is stopped. In this manner, we only employ the Self-Knowledge Distillation term in Eq. (8) for learning, where p_i is teacher's output. Different from GEM-T, which encourages learning from all classes, it prevents the model from changing too far away from its original prediction.

GEM-Aug. Being aware that p_i can be viewed as a teacher's prediction in the GEM-SKD loss, the distillation process can be further improved if p_i is more accurate. Thus, we propose to adopt Test Time Augmentation to logit z_i in GEM-SKD to obtain a more accurate prediction of p_i. $p_i = \text{softmax}(\frac{1}{m} \sum_{j=1}^{m} z_i^j)$, where m is the number of augmented images for each sample.

Our DomainAdaptor equipped with these three different GEM losses is named DomainAdaptor-T, DomainAdaptor-SKD, and DomainAdaptor-Aug, respectively.

Remark: 1) Note that, with the above contributions that fully take advantage of unlabeled data by fusing source and target statistics and scaling the temperature, our method can adapt to the data with a single finetune step without permanently changing its weights. 2) Benefiting from it, DomainAdaptor offers a significant advantage over previous methods that require online updating of model weights, making it suitable for the few-data unseen domain. This will be demonstrated in the experiment section.

4. Experiments

In the experiment, we first compare our method to previous methods and analyze the ablation study of each component. Next, we apply our method to several trained DG SOTA methods to validate its wide application. Finally, we further analyze the hyperparameter sensitivity of our proposed AdaMixBN and how GEM loss works. More experiments can be found in the Supplementary Material.

We employ four domain generalization datasets to test the performance of adapting to unseen domains with a large distributional gap, including **PACS** [21], **VLCS** [39], **OfficeHome** (OH) [42], and **MiniDomainNet** (MDN) [56], a subset of DomainNet [36]. PACS is composed of $9,991$ images with 7 classes. VLCS contains $10,729$ images with 5 classes. OfficeHome has $15,500$ images with 65 classes. MiniDomainNet consists of $140,006$ images with 126 classes. All of these datasets contain 4 domains. PACS and MiniDomainNet have a large domain gap, while VLCS and OfficeHome have a relatively small domain gap. Following the DG training and test scheme, we leave one domain out as the test domain and the others as training domains to pre-train the network. More experimental details can be found in the Supplementary Material.

4.1. Comparison to Previous Methods

We compare our method to several test-time adaptation methods (*i.e.*, Tent [43], SLR [31], LAME [2], and ARM [52]) as shown in Tab. 3. Note that the methods employing online updating are marked with (o) while the others update the parameter temporarily for the current batch. In these methods, ARM requires altering the training stage and has its own trained network (denoted as ARM (Base)), while other methods utilize DeepAll as the trained network. The performance of the ARM baseline is low (*e.g.*, 74.49% vs. 79.44%) because the weight space that the meta-learning training paradigm finds is not suitable for a specific domain, which needs to be finetuned in the test stage to achieve better performance. Besides, Tent, SLR, and ARM all adopt AdaBN [26] that employs test batch statistics for normalization and finetuning the network during adaptation. Differently, LAME does not perform AdaBN and finetune. Instead, it exploits the relationship between images in a batch to refine the predictions with iterative optimization.

Given a trained model, these methods can achieve better performance than the baseline on the PACS dataset, especially for Tent and SLR which employ an online adaption strategy, since a batch of data is enough for the model to produce an approximately accurate statistics estimation on the PACS dataset. However, when encountered with other datasets (*i.e.*, VLCS, OfficeHome, and MiniDomainNet), they cannot even outperform the baseline. For Tent, SLR, and ARM, the application of AdaBN degrades their performance due to inaccurate estimation of test statistics. Be-

Table 3: Comparison to SOTA with Resnet-18 and Resnet-50 as backbone. The best performance (%) is marked as **bold**. Methods marked with (o) are updated online.

	PACS	VLCS	OH	MDN	Avg.
Resnet-18					
DeepAll	$79.44_{\pm0.44}$	$75.77_{\pm0.29}$	$64.61_{\pm0.18}$	$65.12_{\pm0.11}$	71.24
ARM(Base) [52]	$74.49_{\pm3.86}$	$72.10_{\pm0.51}$	$63.33_{\pm1.22}$	$64.97_{\pm0.33}$	68.72
AdaBN [26]	$80.44_{\pm0.29}$	$69.44_{\pm0.48}$	$63.38_{\pm0.12}$	$64.20_{\pm0.11}$	69.37
ARM [52]	$82.47_{\pm0.59}$	$68.21_{\pm1.68}$	$63.15_{\pm0.61}$	$64.91_{\pm0.23}$	69.69
SLR [31]	$81.33_{\pm0.22}$	$69.69_{\pm0.44}$	$63.64_{\pm0.09}$	$64.51_{\pm0.10}$	69.79
SLR (o) [31]	$\mathbf{85.15}_{\pm0.35}$	$74.13_{\pm0.58}$	$64.71_{\pm0.12}$	$43.69_{\pm0.84}$	66.92
Tent [43]	$80.59_{\pm0.26}$	$69.69_{\pm0.44}$	$63.58_{\pm0.14}$	$64.37_{\pm0.09}$	69.56
Tent (o) [43]	$83.56_{\pm0.47}$	$73.37_{\pm0.31}$	$64.54_{\pm0.06}$	$48.50_{\pm1.86}$	67.49
LAME [2]	$80.28_{\pm0.33}$	$75.59_{\pm0.96}$	$63.16_{\pm0.28}$	$64.42_{\pm0.28}$	70.86
DomainAdaptor-T	$85.04_{\pm0.23}$	$77.54_{\pm0.14}$	$65.39_{\pm0.19}$	$66.39_{\pm0.08}$	73.59
DomainAdaptor-SKD	$84.37_{\pm0.28}$	$78.10_{\pm0.14}$	$65.61_{\pm0.14}$	$66.42_{\pm0.08}$	73.63
DomainAdaptor-Aug	$84.93_{\pm0.19}$	$\mathbf{78.50}_{\pm0.22}$	$\mathbf{66.73}_{\pm0.25}$	$\mathbf{68.23}_{\pm0.08}$	**74.60**
Resnet-50					
DeepAll	$84.37_{\pm0.42}$	$76.96_{\pm0.54}$	$70.87_{\pm0.16}$	$71.49_{\pm0.10}$	75.92
ARM (Base) [52]	$70.37_{\pm7.19}$	$77.49_{\pm0.71}$	$49.01_{\pm3.01}$	$71.51_{\pm0.11}$	67.08
AdaBN [26]	$85.29_{\pm0.43}$	$70.95_{\pm0.61}$	$69.37_{\pm0.24}$	$70.21_{\pm0.08}$	73.96
ARM [52]	$86.10_{\pm0.74}$	$78.28_{\pm0.71}$	$66.66_{\pm1.00}$	$70.85_{\pm0.10}$	75.50
SLR [31]	$86.06_{\pm0.47}$	$71.75_{\pm0.62}$	$69.61_{\pm0.19}$	$70.53_{\pm0.07}$	74.49
SLR (o) [31]	$\mathbf{90.32}_{\pm0.27}$	$76.08_{\pm0.31}$	$70.61_{\pm0.18}$	$43.52_{\pm1.15}$	70.13
Tent [43]	$85.38_{\pm0.43}$	$71.11_{\pm0.58}$	$69.58_{\pm0.21}$	$70.43_{\pm0.07}$	74.12
Tent(o) [43]	$88.70_{\pm0.34}$	$74.86_{\pm0.46}$	$71.14_{\pm0.27}$	$46.67_{\pm1.96}$	70.34
LAME [2]	$84.77_{\pm0.29}$	$76.66_{\pm0.77}$	$69.46_{\pm0.14}$	$70.82_{\pm0.12}$	75.43
DomainAdaptor-T	$88.74_{\pm0.30}$	$78.52_{\pm0.57}$	$71.62_{\pm0.14}$	$72.10_{\pm0.09}$	77.75
DomainAdaptor-SKD	$88.57_{\pm0.38}$	$79.11_{\pm0.38}$	$71.89_{\pm0.10}$	$72.51_{\pm0.10}$	78.02
DomainAdaptor-Aug	$88.45_{\pm0.16}$	$\mathbf{79.55}_{\pm0.36}$	$\mathbf{72.84}_{\pm0.05}$	$\mathbf{73.82}_{\pm0.10}$	**78.67**

sides, based on AdaBN, ARM and SLR can improve the performance of the PACS dataset, while Tent has little improvement on all datasets. For LAME that does not employ AdaBN, since it considers the relationship between images in a batch, it requires the trained network to make a correct classification for most classes, which could help correctly infer the classes of these images. However, for VLCS, OfficeHome, and MDN, the trained model has low performance, which is hard for LAME to infer correctly. Besides, OfficeHome and MDN have a large number of classes, which also increases the difficulty of inference.

To better analyze this problem, we calculated the variance of statistics during the training process in Tab. 4. Notably, we observe the highest variance in PACS compared to the other three datasets. Since training with diverse statistics can effectively enhance the model's robustness to test-time distribution shifts, adopting test statistics that have a different distribution of source statistics can benefit the adaptation process and vice versa. As a result, most methods demonstrate obvious improvement on the PACS dataset but fail to perform well on the other datasets. Moreover, despite the sensitivity of the trained model to the distribution shift in the training stage, combining the source statistics enables us to obtain a more robust model, as demonstrated by the improved performance of DomainAdaptor.

By employing AdaMixBN and different GEM variants, our method can mitigate the drawbacks of ARM (requirement of altering training stage), AdaBN (inaccurate statis-

Table 4: The variance of statistics for different datasets.

DataSets	PACS	VLCS	OH	MDN
Variance	1.08	0.05	0.78	0.43

Table 5: Ablation study of our method with AdaMixBN and GEM-Aug. AdaBN is also included for clearer analysis. T is the shorthand of statistics transformation.

AdaBN	AdaMixBN w/o T	GEM-Aug	w/ T	PACS	VLCS	OH	MDN
				79.44	75.77	64.61	65.12
✓				80.44	69.44	63.38	64.20
	✓			83.43	76.62	65.08	65.98
✓		✓		82.85	72.28	65.37	66.97
	✓	✓		59.47	62.63	44.84	25.68
	✓	✓	✓	84.93	78.50	66.73	68.23

tics estimation), and Tent (small gradient for adaptation) and achieves better performance than the SOTA methods on three datasets. Specifically, since the predictions on the PACS dataset are not reliable due to the large domain gap, GEM-T can obtain 5.6% performance improvement on the PACS dataset (*i.e.*, 85.04% vs. 79.44%) by encouraging learning from all classes. Differently, GEM-SKD is better than GEM-T on other datasets. Because predictions on the dataset with a small domain gap (*i.e.*, VLCS) are more reliable and for the dataset with a large number of classes (*i.e.*, OfficeHome, MDN), only a part of classes are critical for a sample. Thus, learning from all classes is meaningless, and GEM-SKD can perform better. Since GEM-Aug is based on GEM-SKD, its performance can be further improved based on it by employing more accurate predictions.

4.2. Ablation Study

We conducted an ablation study on our DomainAdaptor to investigate the efficacy of each component. The loss is vanilla Entropy Minimization (EM) loss if not mentioned.

AdaBN. From Tab. 5, we can observe that applying AdaBN to the baseline can improve the performance on PACS (*i.e.*, 80.44% vs. 79.44%) but degrades on other datasets (*e.g.*, 69.44% vs. 75.77% on VLCS). We hypothesize that in PACS, the images in the same domain have similar styles, providing more accurate statistics estimation for the batch. Besides, the domain gap in PACS is large, which enhances the effectiveness of AdaBN. On the contrary, in other datasets, the intra-domain images have diverse styles, resulting in inaccurate statistics estimation. Their domain gaps are also relatively small compared to PACS and thus achieve performance lower than the baselines.

AdaMixBN. Different from AdaBN, AdaMixBN employs source statistics to ease the inaccurate estimation of statistics. As seen in Tab. 5, AdaMixBN can improve the performance largely on PACS data (*i.e.*, 83.43% vs. 79.44%), and it also can improve on other three datasets. The improvement on PACS shows that by incorporating

Table 6: Performance (%) changes with respect to α for AdaMixBN on different datasets. The best performance of α is marked as **bold**.

α	0.5	0.6	0.7	0.8	0.9	0.99	AdaMixBN
PACS	83.01	83.41	**83.64**	83.42	82.35	79.87	83.43
VLCS	73.02	74.03	75.04	75.86	**76.33**	75.87	76.62
OH	64.68	64.95	65.20	**65.35**	65.24	64.72	65.08
MDN	65.51	65.77	66.03	**66.21**	66.11	65.29	65.98

more source statistics, not only the inaccurate statistics in VLCS and OfficeHome can be accurate, but the already accurate statistics also can be further improved.
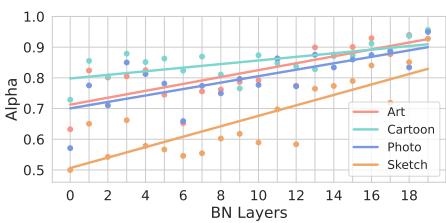
GEM-Aug. Simply applying EM loss cannot effectively learn from highly confident samples. For instance, Tent obtains 80.59% on PACS in Tab. 3 while AdaBN itself can achieve 80.44% in Tab. 5. There is only 0.15% improvement brought by EM loss. Differently, by encouraging further learning from highly confident samples, GEM-Aug loss based on AdaBN can improve more on all four datasets (*e.g.*, 2.41% improvement (82.85% vs. 80.44%) on PACS).

Transformation. However, when AdaMixBN is finetuned without the statistics transformation, GEM-Aug loss degrades the performance significantly (*e.g.*, 83.43% vs. 59.47% on PACS in Tab. 5) due to the weight mismatch problem. Note that not only GEM loss but also other losses that change weight drastically have the same issue, which will be shown in Sec. 4.3. By transforming the source statistics into affine parameters in AdaMixBN, we can mitigate this issue and the performance can be further improved based on AdaMixBN (*e.g.*, 84.93% vs. 83.43% on PACS), which validates the efficacy of our method.
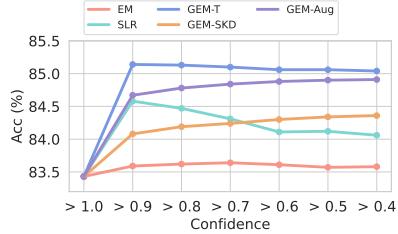
4.3. Further Analysis

Whether AdaMixBN learns the optimal α. We propose AdaMixBN to dynamically adjust the coefficient α between source and batch statistics. To validate whether it learns the optimal α, we finetune α from 0.5 to 0.99 for different datasets to find its optimal value and the corresponding performance. We compare them with the performance produced by the dynamically learned α. As shown in Tab. 6, the optimal α in different datasets are different, and AdaMixBN can approximate the performance of these optimal α values by employing a relative distance measure.
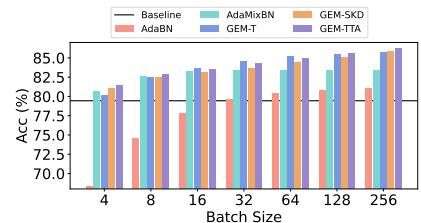
Performance degradation with different α. To investigate whether the performance degradation phenomenon is only caused by GEM loss or also other losses and how α in AdaMixBN influences the degree of degradation, we experiment with three losses (*i.e.*, EM loss, SLR loss, and GEM-Aug loss) equipped with AdaMixBN on PACS with different α. As shown from Tab. 7, when α becomes large, not only does using GEM-Aug loss cause the performance drop but also SLR degrades the performance. Differently,

(a) The values of alpha on the PACS dataset.

(b) Influence of confident samples.

(c) Comparison with different batch sizes.

Figure 5: The values of calculated α in the AdaMixBN (a). The influence of sample (b) and batch size (c) on the performance.

Table 7: Performance (%) degradation with respect to α using different losses in AdaMixBN.

α	0.1	0.3	0.5	0.7	0.9	0.99
AdaMixBN	80.94	82.01	83.01	83.64	82.35	79.87
Finetune *without* statistics transformation						
EM	81.05	82.22	83.14	83.79	82.41	79.10
SLR	81.98	83.10	84.17	83.53	65.66	43.08
GEM-Aug	83.07	82.06	74.40	48.43	22.80	19.77
Finetune *with* statistics transformation						
GEM-Aug	83.32	84.07	84.61	85.08	82.49	82.49

the performance of EM loss does not drop significantly because both GEM loss and SLR loss change model weights drastically, while EM loss makes little changes to the model weights due to the small gradients it produces, resulting in a similar performance to AdaMixBN (*e.g.*, when $\alpha = 0.5$, 83.01% vs. 83.14%). Besides, we find that with more source statistics mixed, the performance drops more significantly. This is caused by the mismatch between fixed source statistics and finetuned model weights and can be addressed with the statistics transformation. As shown in the last line in Tab. 7, after we applying the statistics transformation operation, GEM-Aug loss improves the performance consistently instead of hurting the performance.

The values of dynamic α. In Sec. 3.1.1 we argue that lower layers contain more domain-specific information while higher layers contain less. Therefore a dynamic mixture coefficient is proposed for statistic fusing. To investigate whether this is the case, we plot α obtained by Eq. (2) in different layers of BN and their regression lines on the PACS dataset. As seen from Fig. 5a, in all four domains, α increases as the layer becomes higher, which indicates that source statistics in higher layers are more transferable and should be incorporated with a larger value of α.

Contributions of different confident samples. To investigate whether our proposed GEM loss can better learn from highly confident samples, we compare different losses by finetuning AdaMixBN with samples whose confidence predicted by the model is higher than a threshold. As shown in Fig. 5b, with more low-confident samples incorporated,

the performance of EM loss only slightly improves and then decreases due to the noise introduced by the more low-confident samples. While SLR loss can better exploit samples with confidence larger than 0.9, it decreases a lot when more low-confident samples are added since it cannot inhibit the noise introduced by the low-confident samples. Different from these two losses, by only finetuning samples with confidence higher than 0.9, our three losses can achieve higher performance than EM loss. Besides, by employing the temperature to soften the probability, noise can be reduced and our two losses (*i.e.*, GEM-Aug and GEM-SKD) can all benefit from more low-confident samples and achieve better performance. The performance of GEM-T drops slightly since it encourages the model to make different predictions, which is more sensitive to batch noise.

The influence of batch size. Since both AdaMixBN and the adaptation process are closely related to the batch size, we compare AdaBN, AdaMixBN, and our proposed three losses to see how the batch size influences their performances. As shown in Fig. 5c, with a batch size of 4, the performance of AdaBN drops drastically from 79.44% to 68.40% because the statistics estimated from only 4 images are very inaccurate and cannot effectively normalize the features. On the contrary, our proposed AdaMixBN can still achieve higher performance than the baseline by incorporating more source statistics. As the number of images increases, the performance can be further improved and then plateaus, which means the statistics are accurate and can not be improved. Although the performance of AdaBN also increases, AdaMixBN can still outperform it with the source statistics. When more unlabeled images are utilized, the performances of adaptation with different losses also increase. We hypothesize that sample diversity plays a significant role in the adaptation process. With more samples incorporated, the noise produced by incorrectly predicted samples is alleviated, and the optimization direction toward the global minimum can be more accurate.

Comparison with few test data. To compare with other methods under limited test data, we divide the original dataset into several subsets of the same size and evaluate all the methods on each subset, with model parameters reset before finetuning on a new subset. The adaption batch size

Table 8: Performance (%) comparison with few data.

Subset Size	64	128	256	512	1024	2048	4096
Tent	80.79	81.00	81.24	81.75	82.41	82.95	83.52
SLR	81.33	81.99	82.55	83.46	84.26	84.79	85.15
DomainAdaptor-T	85.04	85.04	85.04	85.04	85.04	85.04	85.04

Table 9: Performance (%) of applying DomainAdaptor to previously pretrained models.

	P	A	C	S	Avg.
DeepAll	96.44	78.25	75.57	67.48	79.44
+ DomainAdaptor-T	97.40	82.51	80.65	79.59	85.04
+ DomainAdaptor-SKD	97.01	82.22	80.46	77.78	84.37
+ DomainAdaptor-Aug	97.20	83.04	80.20	79.28	84.93
MLDG [22]	94.91	80.59	76.20	77.65	82.34
+ DomainAdaptor-T	96.35	82.84	81.28	80.96	85.36
+ DomainAdaptor-SKD	96.36	83.04	81.41	81.02	85.46
+ DomainAdaptor-Aug	96.16	82.95	81.37	81.54	85.50
FSDCL [45]	95.63	85.30	81.31	81.17	85.85
+ DomainAdaptor-T	96.65	87.01	83.53	81.04	87.06
+ DomainAdaptor-SKD	96.65	87.40	83.66	79.28	86.75
+ DomainAdaptor-Aug	96.53	86.57	83.70	80.89	86.92
MVRML [51]	95.29	85.20	79.97	83.11	85.89
+ DomainAdaptor-T	96.18	85.99	83.98	83.73	87.47
+ DomainAdaptor-SKD	96.23	85.92	83.86	83.59	87.40
+ DomainAdaptor-Aug	96.41	86.69	84.33	85.05	88.12

is 64. The resulting accuracies are then aggregated across all subsets. As shown from Tab. 8, the performances of both Tent and SLR are low when the subset size is small, indicating their poor adaptation ability in the few-data scenario. Differently, DomainAdaptor can achieve consistent improvement regardless of subset size due to its ability to perform adaptation with only a single batch.

The performance of being applied to the trained model. Since our method does not require altering the training stage and thus can be applied to any trained CNN models, we apply it to several trained recent DG SOTA models on PACS. As seen from Tab. 9, our method can significantly improve the baseline method's performance (*e.g.*, 85.04% vs. 79.44%). When applied to existing DG SOTA models, our method can make further improvements (*e.g.*, 88.12% vs. 85.89% for MVRML) without the requirement of altering the training stage and retraining the model.

Comparison to self-supervised losses. To determine the effectiveness of our method compared to previous self-supervised losses, we conducted a comparison with rotation [37] and jigsaw [3] loss. Since these self-supervised losses require an extra classification head, to assess their performance, we re-train the DeepAll with a new head (denoted as Base) and report the results with a single finetuning for each batch. As shown in Tab. 10, these losses also do not yield significant performance improvement since they cannot produce a sufficiently large gradient for weight up-

Table 10: Performance (%) comparison of adapting with different self-supervised losses or different statistics. The baselines of rotation and jigsaw losses are denoted as Base.

	PACS	VLCS	OH	MDN	Avg.
DeepAll	$79.44_{\pm0.44}$	$75.77_{\pm0.29}$	$64.61_{\pm0.18}$	$65.12_{\pm0.11}$	71.24
Rot (Base)	$80.91_{\pm0.67}$	$75.02_{\pm0.38}$	$64.24_{\pm0.15}$	$64.65_{\pm0.09}$	71.22
+ RotLoss	$81.26_{\pm0.46}$	$69.82_{\pm0.61}$	$63.00_{\pm0.24}$	$63.50_{\pm0.09}$	69.39
+ GEM-Aug	$83.16_{\pm0.76}$	$72.77_{\pm0.63}$	$63.87_{\pm0.32}$	$65.41_{\pm0.10}$	71.29
Jig (Base)	$78.60_{\pm0.73}$	$74.91_{\pm0.60}$	$61.70_{\pm0.58}$	$62.55_{\pm0.10}$	69.43
+ JigLoss	$80.24_{\pm0.21}$	$70.53_{\pm1.07}$	$62.47_{\pm0.44}$	$63.85_{\pm0.11}$	69.25
+ GEM-Aug	$84.51_{\pm0.40}$	$73.64_{\pm1.19}$	$63.95_{\pm0.33}$	$64.92_{\pm0.10}$	71.75
AdaBN	$80.44_{\pm0.29}$	$69.44_{\pm0.48}$	$63.38_{\pm0.12}$	$64.20_{\pm0.11}$	69.37
AdaBN (src+test)	$81.80_{\pm0.45}$	$76.11_{\pm0.31}$	$65.26_{\pm0.20}$	$65.89_{\pm0.12}$	72.27
AdaMixBN (α)	$83.43_{\pm0.23}$	$76.62_{\pm0.27}$	$65.08_{\pm0.12}$	$65.98_{\pm0.10}$	72.78
AdaMixBN ($1-\alpha$)	$81.31_{\pm0.27}$	$69.51_{\pm0.50}$	$63.47_{\pm0.13}$	$64.36_{\pm0.10}$	69.66
DomainAdaptor-Aug	$84.93_{\pm0.19}$	$78.50_{\pm0.22}$	$66.73_{\pm0.25}$	$68.23_{\pm0.08}$	74.60

dating. Differently, benefiting from GEM loss, our method still can produce a substantial performance boost.

Normalization with different statistics. Since calculating the statistics with source and target data may serve as an upper bound, we recalculate the statistics, as shown in Tab. 10 (denoted as AdaBN (src+test)). While we observe an improvement compared to AdaBN, it only results in marginal improvement over the baseline. Differently, our proposed AdaMixBN consistently achieves better performance than the baseline. We hypothesize that AdaMixBN benefits from the dynamically mixing statistics between the source and target statistics from different layers, which cannot be achieved by simply accumulating the statistics. Besides, we also adopt $1-\alpha$ during mixing to verify whether the similarity-based distance works. As shown in Tab. 10, it degrades significantly compared to using α, which validates the effectiveness of our proposed measurement.

5. Conclusion

In this paper, we proposed DomainAdaptor to deal with the fully test-time adaptation problem for any incoming test batch with a large domain gap. It consists of AdaMixBN and Generalized Entropy Minimization loss to effectively exploit unlabeled test data. Specifically, AdaMixBN dynamically fuses statistics between source and test batch statistics for an accurate statistics estimation and Generalized Entropy Minimization loss effectively enhances the adaptation ability of the AdaMixBN module. Besides, a statistics transformation operation was incorporated to prevent performance degradation in AdaMixBN. Extensive experiments validated the effectiveness of our method.

References

[1] Mathilde Bateson, Hervé Lombaert, and Ismail Ben Ayed. Test-time adaptation with shape moments for image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2022. 1

[2] Malik Boudiaf, Romain Mueller, Ismail Ben Ayed, and Luca Bertinetto. Parameter-free online test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 1, 6

[3] Fabio M Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain generalization by solving jigsaw puzzles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. 2, 4, 9

[4] Yang Chen, Yu Wang, Yingwei Pan, Ting Yao, Xinmei Tian, and Tao Mei. A style and semantic memory mechanism for domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 2

[5] Zhun Deng, Frances Ding, Cynthia Dwork, Rachel Hong, Giovanni Parmigiani, Prasad Patil, and Pragya Sur. Representation via representations: Domain generalization via adversarially learned invariant representations. *arXiv*, 2020. 2

[6] Zhipeng Ding, Xu Han, Peirong Liu, and Marc Niethammer. Local temperature scaling for probability calibration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. 2

[7] Yingjun Du, Xiantong Zhen, Ling Shao, and Cees GM Snoek. Metanorm: Learning to normalize few-shot batches across domains. In *International Conference on Learning Representations*, 2020. 2

[8] Xinjie Fan, Qifei Wang, Junjie Ke, Feng Yang, Boqing Gong, and Mingyuan Zhou. Adversarially adaptive normalization for single domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 2

[9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017. 4

[10] François Fleuret et al. Test time adaptation through perturbation robustness. In *Advances in Neural Information Processing Systems Workshop*, 2021. 1, 2, 3

[11] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, 2017. 2

[12] Jia Guo, Chen Minghao, Hu Yao, Zhu Chen, He Xiaofei, and Cai Deng. Spherical knowledge distillation. *arXiv*, 2020. 5

[13] Jintao Guo, Na Wang, Lei Qi, and Yinghuan Shi. ALOFT: A lightweight mlp-like architecture with dynamic low-frequency transform for domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. 2

[14] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv*, 2019. 1, 2

[15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv*, 2015. 2, 5

[16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv*, 2015. 1

[17] Yusuke Iwasawa and Yutaka Matsuo. Test-time classifier adjustment module for model-agnostic domain generalization. In *Advances in Neural Information Processing Systems*, 2021. 2

[18] Tom Joy, Francesco Pinto, Ser-Nam Lim, Philip HS Torr, and Puneet K Dokania. Sample-dependent adaptive temperature scaling for improved calibration. *arXiv*, 2022. 2

[19] Neerav Karani, Ertunc Erdil, Krishna Chaitanya, and Ender Konukoglu. Test-time adaptable neural networks for robust medical image segmentation. *Medical Image Analysis*, 2021. 1

[20] Anna Kukleva, Moritz Böhle, Bernt Schiele, Hilde Kuehne, and Christian Rupprecht. Temperature schedules for self-supervised contrastive methods on long-tail data. In *International Conference on Learning Representations*, 2023. 2

[21] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2017. 6

[22] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Learning to generalize: Meta-learning for domain generalization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. 2, 9

[23] Da Li, Jianshu Zhang, Yongxin Yang, Cong Liu, Yi-Zhe Song, and Timothy M Hospedales. Episodic training for domain generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019. 2

[24] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain generalization with adversarial feature learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. 2

[25] Lei Li, Veronika A Zimmer, Wangbin Ding, Fuping Wu, Liqin Huang, Julia A Schnabel, and Xiahai Zhuang. Random style transfer based domain generalization networks integrating shape and spatial information. *arXiv*, 2020. 2

[26] Yanghao Li, Naiyan Wang, Jianping Shi, Xiaodi Hou, and Jiaying Liu. Adaptive batch normalization for practical domain adaptation. *Pattern Recognition*, 2018. 2, 6

[27] Yiying Li, Yongxin Yang, Wei Zhou, and Timothy M Hospedales. Feature-critic networks for heterogeneous domain generalization. In *International Conference on Machine Learning*, 2019. 2

[28] Sihao Lin, Hongwei Xie, Bing Wang, Kaicheng Yu, Xiaojun Chang, Xiaodan Liang, and Gang Wang. Knowledge distillation via the target-aware transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 2

[29] Yuejiang Liu, Parth Kothari, Bastien van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. TTT++: When does self-supervised test-time training fail or thrive? In *Advances in Neural Information Processing Systems*, 2021. 2

[30] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant fea-

ture representation. In *International Conference on Machine Learning*, 2013. 1, 2

[31] Chaithanya Kumar Mummadi, Robin Hutmacher, Kilian Rambach, Evgeny Levinkov, Thomas Brox, and Jan Hendrik Metzen. Test-time adaptation to distribution shift by confidence maximization and input transformation. *arXiv*, 2021. 1, 2, 6

[32] Jaemin Na, Heechul Jung, Hyung Jin Chang, and Wonjun Hwang. FixBi: Bridging domain spaces for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 1

[33] Hyeonseob Nam and Hyo-Eun Kim. Batch-instance normalization for adaptively style-invariant neural networks. In *Advances in Neural Information Processing Systems*, 2018. 3

[34] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *Proceedings of the European Conference on Computer Vision*, 2016. 4

[35] Prashant Pandey, Mrigank Raman, Sumanth Varambally, and Prathosh Ap. Generalization on unseen domains via inference-time label-preserving target projections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 2

[36] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019. 6

[37] 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*, 2020. 1, 2, 4, 9

[38] Christian Tomani, Sebastian Gruber, Muhammed Ebrar Erdem, Daniel Cremers, and Florian Buettner. Post-hoc uncertainty calibration for domain drift scenarios. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 2

[39] Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2011. 6

[40] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv*, 2016. 2

[41] Jeya Maria Jose Valanarasu, Pengfei Guo, Vibashan VS, and Vishal M Patel. On-the-fly test-time adaptation for medical image segmentation. *arXiv*, 2022. 1

[42] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017. 6

[43] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021. 1, 2, 3, 4, 6

[44] Feng Wang and Huaping Liu. Understanding the behaviour of contrastive loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 2

[45] Yue Wang, Lei Qi, Yinghuan Shi, and Yang Gao. Feature-based style randomization for domain generalization. *IEEE Transactions on Circuits and Systems for Video Technology*, 2022. 9

[46] Hui Wu, Min Wang, Wengang Zhou, Houqiang Li, and Qi Tian. Contextual similarity distillation for asymmetric image retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 2

[47] Qilong Wu, Xiangyu Yue, and Alberto Sangiovanni-Vincentelli. Domain-agnostic test-time adaptation by prototypical training with auxiliary data. In *Advances in Neural Information Processing Systems Workshop*, 2021. 2

[48] Qinwei Xu, Ruipeng Zhang, Ya Zhang, Yanfeng Wang, and Qi Tian. A fourier-based framework for domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 2

[49] Chaoning Zhang, Kang Zhang, Trung X Pham, Axi Niu, Zhinan Qiao, Chang D Yoo, and In So Kweon. Dual temperature helps contrastive learning without many negative samples: Towards understanding and simplifying moco. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 2

[50] Jian Zhang, Lei Qi, Yinghuan Shi, and Yang Gao. Generalizable model-agnostic semantic segmentation via target-specific normalization. *Pattern Recognition*, 2022. 2

[51] Jian Zhang, Lei Qi, Yinghuan Shi, and Yang Gao. MVDG: A unified multi-view framework for domain generalization. In *Proceedings of the European Conference on Computer Vision*, 2022. 2, 9

[52] Marvin Zhang, Henrik Marklund, Nikita Dhawan, Abhishek Gupta, Sergey Levine, and Chelsea Finn. Adaptive risk minimization: Learning to adapt to domain shift. In *Advances in Neural Information Processing Systems*, 2021. 2, 6

[53] Borui Zhao, Quan Cui, Renjie Song, Yiyu Qiu, and Jiajun Liang. Decoupled knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Pecognition*, pages 11953–11962, 2022. 2

[54] Shanshan Zhao, Mingming Gong, Tongliang Liu, Huan Fu, and Dacheng Tao. Domain generalization via entropy regularization. In *Advances in Neural Information Processing Systems*, 2020. 2

[55] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain generalization with mixstyle. In *International Conference on Learning Representations*, 2020. 2

[56] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain adaptive ensemble learning. *IEEE Transactions on Image Processing*, 2021. 6

[57] Ziqi Zhou, Lei Qi, Xin Yang, Dong Ni, and Yinghuan Shi. Generalizable cross-modality medical image segmentation via style augmentation and dual normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 2