Improved Test-Time Adaptation for Domain Generalization

Liang Chen[1] Yong Zhang[2*] Yibing Song[3] Ying Shan[2] Lingqiao Liu[1*]
[1] The University of Adelaide [2] Tencent AI Lab [3] AI[3] Institute, Fudan University
{liangchen527, zhangyong201303, yibingsong.cv}@gmail.com
yingsshan@tencent.com lingqiao.liu@adelaide.edu.au

Abstract

*The main challenge in domain generalization (DG) is to handle the distribution shift problem that lies between the training and test data. Recent studies suggest that test-time training (TTT), which adapts the learned model with test data, might be a promising solution to the problem. Generally, a TTT strategy hinges its performance on two main factors: selecting an appropriate auxiliary TTT task for updating and identifying reliable parameters to update during the test phase. Both previous arts and our experiments indicate that TTT may not improve but be detrimental to the learned model if those two factors are not properly considered. This work addresses those two factors by proposing an **I**mproved **T**est-**T**ime **A**daptation (ITTA) method. First, instead of heuristically defining an auxiliary objective, we propose a learnable consistency loss for the TTT task, which contains learnable parameters that can be adjusted toward better alignment between our TTT task and the main prediction task. Second, we introduce additional adaptive parameters for the trained model, and we suggest only updating the adaptive parameters during the test phase. Through extensive experiments, we show that the proposed two strategies are beneficial for the learned model (see Figure 1), and ITTA could achieve superior performance to the current state-of-the-art methods on several DG benchmarks. Code is available at https://github.com/liangchen527/ITTA.*

1. Introduction

Recent years have witnessed the rapid development of deep learning models, which often assume the training and test data are from the same domain and follow the same distribution. However, this assumption does not always hold in real-world scenarios. Distribution shift among the source and target domains is ubiquitous in related areas [35], such as autonomous driving or object recognition tasks, resulting in poor performances for delicately designed models and hindering the further application of deep learning techniques.

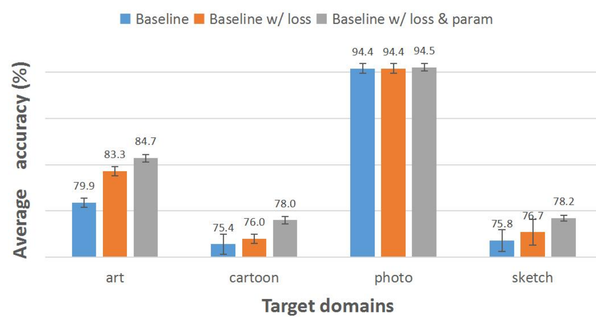


Figure 1. Performance improvements from the proposed two strategies (*i.e.* introducing a learnable consistency **loss** and including additional adaptive **param**eters to improve TTT) for the baseline model (*i.e.* ResNet18 [30] with existing augmentation strategy [74]). Experiments are conducted on the PACS dataset [37] with the leave-one-out setting. Following [27], we use 60 sets of random seeds and hyper-parameters for each target domain. The reported average accuracy and error bars verify the effectiveness of our method.

Domain generalization (DG) [2,8,16,23,24,31,38–40,40,44,46,50,51,68], designed to generalize a learned model to unseen target domains, has attracted a great deal of attention in the research community. The problem can be traced back to a decade ago [7], and various approaches have been proposed to push the DG boundary ever since. Those efforts include invariant representation learning [28,46,48,57], adversarial learning [23,40,44,68], augmentation [9,41,42,65,74], or meta-learning [2,16,38,39]. Despite successes on certain occasions, a recent study [27] shows that, under a rigorous evaluation protocol, most of these arts are inferior to the baseline empirical risk minimization (ERM) method [60]. This finding is not surprising, as most current arts strive to decrease the distribution shift only through the training data while overlooking the contributions from test samples.

Recently, the test-time training (TTT) technique [59] has been gaining momentum for easing the distribution shift problem. TTT lies its success in enabling dynamic tuning of the pretrained model with the test samples via an auxiliary TTT task, which seems to be a promising effort when

[*]Corresponding authors. This work is done when L. Chen is an intern in Tencent AI Lab.

confronting data from different domains. However, TTT is not guaranteed to improve the performance. Previous arts [45, 62] indicate that selecting an appropriate auxiliary TTT task is crucial, and an inappropriate one that does not align with the main loss may deteriorate instead of improving the performance. Meanwhile, it is pointed out in [62] that identifying reliable parameters to update is also essential for generalization, which is in line with our experimental findings in Sec. 5.3. Both of these two tasks are non-trivial, and there are limited efforts made to address them.

This paper aims to improve the TTT strategy for better DG. First, different from previous works that empirically define auxiliary objectives and assume they are aligned with the main task, our work does not make such assumptions. Instead, we suggest learning an appropriate auxiliary loss for test-time updating. Specifically, encouraged by recent successes in multi-view consistency learning [13, 26, 29], we propose to augment the consistency loss by adding learnable parameters based on the original implementation, where the parameters can be adjusted to assure our TTT task can be more aligned with the main task and are updated by enforcing the two tasks share the same optimization direction. Second, considering that identifying reliable parameters to update is an everlasting job given the growing size of current deep models, we suggest introducing new adaptive parameters after each block during the test phase, and we only tune the new parameters by the learned consistency loss while leaving the original parameters unchanged. Through extensive evaluations on the current benchmark [27], we illustrate that the learnable consistency loss performs more effectively than the self-supervised TTT tasks adopted in previous arts [59, 62], and by tuning only the new adaptive parameters, our method is superior to existing strategies that update all the parameters or part of them.

This work aims to ease the distribution shift problem by improving TTT, and the main contributions are three-fold:

- We introduce a learnable consistency loss for test-time adaptation, which can be enforced to be more aligned with the main loss by tuning its learnable parameters.
- We introduce new adaptive parameters for the trained model and only update them during the test phase.
- We conduct experiments on various DG benchmarks and illustrate that our ITTA performs competitively against current arts under the rigorous setting [27] for both the multi-source and single-source DG tasks.

2. Related Works

2.1. Domain Generalization.

Being able to generalize to new environments while deploying is a challenging and practical requirement for current deep models. Existing DG approaches can be roughly categorized into three types. **(1) Invariant representation learning:** The pioneering work [5] theoretically proves that if the features remain invariant across different domains, then they are general and transferable to different domains. Guided by this finding, [46] uses maximum mean discrepancy (MMD) to align the learned features, and [25] proposes to use a multi-domain reconstruction auto-encoder to obtain invariant features. More recently, [57] suggests maximizing the inner product of gradients from different domains to enforce invariance, and a similar idea is proposed in [51] where these gradients are expected to be similar to their mean values. **(2) Optimization algorithms:** Among the different optimization techniques adopted in DG, prevailing approaches resort to adversarial learning [23, 40, 44, 68] and meta-learning [2, 16, 38, 39]. Adversarial training is often used to enforce the learned features to be agnostic about the domain information. In [23], a domain-adversarial neural network (DANN) is implemented by asking the mainstream feature to maximize the domain classification loss. This idea is also adopted in [44], where adversarial training and an MMD constraint are employed to update an auto-encoder. Meanwhile, the meta-learning technique is used to simulate the distribution shifts between seen and unseen environments [2, 16, 38, 39], and most of these works are developed based on the MAML framework [20]. **(3) Augmentation:** Most augmentation skills applied in the generalization tasks are operated in the feature level [34, 41, 47, 74] except for [11,65,67] which mix images [67] or its phase [65] to synthesize new data. To enable contrastive learning, we incorporate an existing augmentation strategy [74] in our framework. This method originated from AdaIN [32], which synthesizes new domain information by mixing the statistics of the features. Similar ideas can be found in [42, 47].

2.2. Test-Time Training and Adaptation

Test-Time Training (TTT) is first introduced in [59]. The basic paradigm is to employ a test-time task besides the main task during the training phase and updates the pre-trained model using the test data with only the test-time objective before the final prediction step. The idea is empirically proved effective [59] and further developed in other related areas [3, 10, 12, 14, 21, 22, 43, 55, 62, 64, 72, 73]. Most current works focus on finding auxiliary tasks for updating during the test phase, and the efforts derive from self-supervion [3, 10, 21, 22, 43, 59], meta-learning [64, 72, 73], information entropy [62], pseudo-labeling [12, 14], to name a few. However, not all empirically selected test-time tasks are effective. A recent study [45] indicates that only when the auxiliary loss aligns with the main loss can TTT improve the trained model. Inspired by that, we propose a learnable consistency loss and enforce alignment between the two objectives. Results show that our strategy can be beneficial for

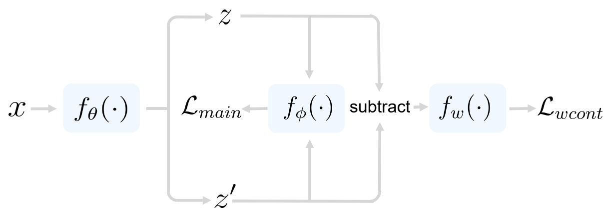


Figure 2. Training process of ITTA. We use x from the source domain as input for the feature extractor $f_\theta(\cdot)$ to obtain the representation z and its augmented version z', where the augmentation skill from [74] is applied. The classifier $f_\phi(\cdot)$ and weight subnetwork $f_w(\cdot)$ are used to compute the main loss $\mathcal{L}_{main}$ and learnable consistency loss $\mathcal{L}_{wcont}$. Please refer to our text for details.

the trained model (see Figure 1).

Meanwhile, [62] suggests that auxiliary loss is not the only factor that affects the performance. Selecting reliable parameters to update is also crucial within the TTT framework. Given the large size of current models, correctly identifying these parameters may require tremendous amounts of effort. To this end, instead of heuristically selecting candidates, we propose to include new adaptive parameters for updating during the test phase. Experimental results show that the proposed method can obtain comparable performances against existing skills.

3. Methodology

In the task of DG, we are often given access to data from S ($S \geq 1$) source domains $\mathcal{D}_s = \{D_1, D_2, ..., D_S\}$ and expect a model to make good prediction on unseen target domains $\mathcal{D}_t = \{D_1, D_2, ..., D_T\}$ ($T \geq 1$). Our method aims to improve the test-time training (TTT) strategy for better DG. The improvements are two-fold. First, we propose a learnable consistency loss for the TTT task, which could be enforced to align with the main objective by tuning its learnable weights. Second, we suggest including additional adaptive parameters and only updating these adaptive parameters during the test phase.

3.1. A Learnable Consistency Loss for TTT

The TTT strategies have shown promising performances when dealing with distribution shift problems [43, 62]. However, their successes are depended on the empirically selected auxiliary TTT tasks, which may deteriorate the performances if chosen improperly. Motivated by the recent successes in multi-view consistency learning [13, 26, 29], we suggest adopting a consistency loss in our TTT task. Note that the naive consistency loss is still not guaranteed to be effective as prior art [45] indicates that only when the auxiliary loss aligns with the main loss, can TTT improves the performance. To this end, we propose to augment the auxiliary loss with learnable parameters that could be adjusted toward a better alignment between the TTT and main tasks. In our case, we make the adopted consistency loss learnable by

Algorithm 1 Pseudo code of the training phase of ITTA in a PyTorch-like style.

```
#  fθ, fφ, fw:  feature extractor, classifier, weight subnetwork
#  α, 0:  weight paramter, all zero tensor

#  training process
for x, y in training_loader:  #  load a minibatch with N samples
    def forward_process(x, y):
        z, z′ = fθ.forward(x)
        #  computing losses
        L_main = CrossEntropyLoss(fφ.forward(z),  y)
        L_main+ = CrossEntropyLoss(fφ.forward(z′),  y)
        L_wcont = MSELoss(fw.forward(z − z′),  0)

        return L_main,  L_wcont

    #  SGD update: feature extractor and classifier
    L_main,  L_wcont = forward_process(x, y)
    ([fθ.params, fφ.params]).zero_grad()
    (L_main + αL_wcont).backward()
    update([fθ.params,  fφ.params])

    #  compute objectives for updating weight subnetwork
    L_main,  L_wcont = forward_process(x, y)
    L_main.backward()
    ĝ_main = fθ.params.grad.clone().normalize()
    fθ.params.zero_grad()
    L_wcont.backward()
    ĝ_wcont = fθ.params.grad.clone().normalize()

    #  SGD update: weight subnetwork
    MSELoss(ĝ_main,  ĝ_wcont).backward()
    fw.params.zero_grad()
    update(fw.params)
```

introducing a weight subnetwork that allows flexible ways to measure the consistency between two views of the same instance.

We first introduce the pipeline of our training framework. Given the D dimensional representation $z \in \mathbb{R}^D$[1] and its corresponding augmented version z' that are obtained from a feature extractor (*i.e.* $\{z, z'\} = f_\theta(x)$, where x is an input image from $\mathcal{D}_s$, and $f_\theta(\cdot)$ is the feature extractor parameterized by θ. In our implementation, we use the existing augmentation method [74] to obtain z' by modifying the intermediate activation in $f_\theta(x)$. We show in our supplementary material that our framework can also thrive with other augmentation strategies), our learnable consistency loss is given by,

$$\mathcal{L}_{wcont} = \|f_w(z - z')\|, \tag{1}$$

where $\|\cdot\|$ denotes the $L2$ norm; $f_w(\cdot)$ is the weight subnetwork parameterized by w. To make the training process more stable and potentially achieve better performance, we apply a dimension-wise nonlinear function to map each dimension of $z - z'$ before calculating the $L2$ norm. That is, $\forall h \in \mathbb{R}^D$, $f_w(h)$ is implemented by stacking layers of a nonlinear function: ReLU$(a * h + b)$, where $a \in \mathbb{R}^D$ and

[1]We omit the batch dimensions of the variables for simplicity.

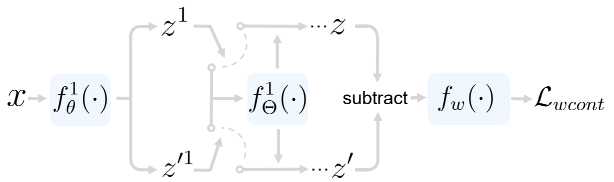


Figure 3. Test adaptation process of ITTA. Different from that in the training stage, we include additional adaptive parameters f_Θ after each block of the feature extractor f_θ. For each test sample x, the intermediate representations z^i and z'^i obtained from f_θ^i are passed to f_Θ^i before going to the next block f_θ^{i+1}. We use the learnable consistency loss $\mathcal{L}_{wcont}$ as the objective to update f_Θ. Please refer to our text for details.

$b \in \mathbb{R}^D$ are the weight and bias from the nonlinear function, and different layers of a, b form the parameter w in f_w. In effect, this creates a piecewise-linear mapping function for h: depending on the value of h, the output could be 0, a constant, or a scaling-and-shifted version of h. More studies about the design of f_w are provided in our supplementary material. Compared to the naive consistency learning without f_w, our $\mathcal{L}_{wcont}$ can be more flexible with an adjustable f_w, which we show in the following is the key for learning an appropriate loss in the improved TTT framework.

Combining $\mathcal{L}_{wcont}$ with the main loss $\mathcal{L}_{main}$ which applies the cross-entropy loss (CE) for both the original and augmented inputs (*i.e.* $\mathcal{L}_{main} = \text{CE}(f_\phi(z), y) + \text{CE}(f_\phi(z'), y)$, where f_ϕ is the classifier parameterized by ϕ, and y is the corresponding label), the objective for the feature extractor and classifier can be formulated into,

$$\min_{\{\theta,\phi\}} \mathcal{L}_{main} + \alpha \mathcal{L}_{wcont}, \tag{2}$$

where α is the weight parameter that balances the contributions from the two terms. A simple illustration of the workflow is shown in Figure 2.

From Eq. (2), the expected gradients for the feature extractor from $\mathcal{L}_{main}$ and $\mathcal{L}_{wcont}$ can be represented as,

$$\begin{cases} \mathbf{g}_{main} = \nabla_\theta(\text{CE}(f_\phi(z), y) + \text{CE}(f_\phi(z'), y)), & (3) \\ \mathbf{g}_{wcont} = \nabla_\theta \| f_w(z - z') \|. & (4) \end{cases}$$

We observe that the direction of $\mathbf{g}_{wcont}$ is also determined by the weight subnetwork $f_w(\cdot)$, which should be close with $\mathbf{g}_{main}$ to ensure alignment between $\mathcal{L}_{main}$ and $\mathcal{L}_{wcont}$ [45, 59]. To this end, we propose a straightforward solution by enforcing equality between the normalized versions of $\mathbf{g}_{main}$ and $\mathbf{g}_{wcont}$, and we use this term as the objective for updating $f_w(\cdot)$, which gives,

$$\min_{w} \mathcal{L}_{align}, \quad \text{s.t. } \mathcal{L}_{align} = \| \hat{\mathbf{g}}_{main} - \hat{\mathbf{g}}_{wcont} \|, \tag{5}$$

where $\hat{\mathbf{g}}_{main} = \frac{\mathbf{g}_{main} - \mathbb{E}_{g_{main}}}{\sigma_{g_{main}}}$, and similar for $\hat{\mathbf{g}}_{wcont}$.

In our implementation, we update $\{\theta, \phi\}$ and w in an alternative manner. Pseudo code of the training process are shown in Algorithm 1.

Algorithm 2 Pseudo code of the test phase of ITTA in a PyTorch-like style.

```
# f_θ, f_φ: feature extractor, classifier
# f_w, f_Θ: weight subnetwork, additional adaptive blocks
# m, 0: total number of blocks in f_θ, all zero tensor

# test process
for x in test_loader:  # load a test batch
    def forward_process(x):
        z^1, z'^1 = f_Θ^1.forward((f_θ^1.forward(x)))  # first blocks
        for i in range(2, m + 1):  # the following m − 1 blocks
            z^i, z'^i = f_θ^i.forward(z^{i−1}), f_θ^i.forward(z'^{i−1})
            z^i, z'^i = f_Θ^i.forward(z^i), f_Θ^i.forward(z'^i)

        return z^i,  z'^i

    # test adaptation phase: SGD update additional adaptive parameters
    z,  z' = forward_process(x)
    L_wcont = MSELoss(f_w.forward(z − z'), 0)
    f_Θ.params.zero_grad()
    L_wcont.backward()
    update(f_Θ.params)

    # final prediction
    z,  _ = forward_process(x)
    result = f_φ.forward(z)
```

3.2. Including Additional Adaptive Parameters

Selecting expressive and reliable parameters to update during the test phase is also essential in the TTT framework [62]. Some strategies decide to update all the parameters from the feature extractor [3, 43], while others use only the parameters from the specific layers for updating [62, 70]. Given the fact that the sizes of current deep models are often very large and still growing, exhaustively trying different combinations among the millions of candidates seems to be an everlasting job. As there are no consensuses on which parameter should be updated, we suggest another easy alternative in this work.

Specifically, assuming there are a total of m blocks in the pretrained feature extractor $f_\theta(\cdot)$, and the i-th block can be denoted as $f_\theta^i(\cdot)$. Then the intermediate representation z^i from $f_\theta^i(\cdot)$ can be formulated as,

$$z^i = f_\theta^i(z^{i-1}), \quad \text{s.t. } z^1 = f_\theta^1(x). \tag{6}$$

We propose to include additional adaptive block f_Θ that is parameterized by Θ after each block of f_θ during the test-time adaptation phase, which reformulates Eq. (6) into,

$$z^i = f_\Theta^i(f_\theta^i(z^{i-1})), \quad \text{s.t. } z^1 = f_\Theta^1(f_\theta^1(x)), \tag{7}$$

where $f_\Theta(\cdot)$ does not change the dimension and sizes of the intermediate representations. In our work, we use a structure similar to f_w to implement f_Θ. Note z^m is simplified as z in this phase, and the same process is applied for obtaining z'.

Then, in the test-time adaptation phase, we suggest only updating the new adaptive parameters via the learned consistency loss. The optimization process can be written as,

Table 1. Multi sources domain generalization. Experiments are conducted on the DomainBed benchmark [27]. All methods are examined for 60 trials in each unseen domain. Top5 accumulates the number of datasets where a method achieves the top 5 performances. The score here accumulates the numbers of the dataset where a specific art obtains larger accuracy than ERM on account of the variance. Best results are colored as red. Among the 22 methods compared, less than a quarter outperforms ERM in most datasets (Score ≥ 3).

	PACS	VLCS	OfficeHome	TerraInc	DomainNet	Avg.	Top5↑	Score↑
MMD [40]	81.3 ± 0.8	74.9 ± 0.5	59.9 ± 0.4	42.0 ± 1.0	7.9 ± 6.2	53.2	1	2
RSC [33]	80.5 ± 0.2	75.4 ± 0.3	58.4 ± 0.6	39.4 ± 1.3	27.9 ± 2.0	56.3	0	1
IRM [1]	80.9 ± 0.5	75.1 ± 0.1	58.0 ± 0.1	38.4 ± 0.9	30.4 ± 1.0	56.6	0	1
ARM [71]	80.6 ± 0.5	75.9 ± 0.3	59.6 ± 0.3	37.4 ± 1.9	29.9 ± 0.1	56.7	0	0
DANN [23]	79.2 ± 0.3	76.3 ± 0.2	59.5 ± 0.5	37.9 ± 0.9	31.5 ± 0.1	56.9	1	1
GroupGRO [54]	80.7 ± 0.4	75.4 ± 1.0	60.6 ± 0.3	41.5 ± 2.0	27.5 ± 0.1	57.1	0	1
CDANN [44]	80.3 ± 0.5	76.0 ± 0.5	59.3 ± 0.4	38.6 ± 2.3	31.8 ± 0.2	57.2	0	0
VREx [36]	80.2 ± 0.5	75.3 ± 0.6	59.5 ± 0.1	43.2 ± 0.3	28.1 ± 1.0	57.3	1	1
CAD [52]	81.9 ± 0.3	75.2 ± 0.6	60.5 ± 0.3	40.5 ± 0.4	31.0 ± 0.8	57.8	1	2
CondCAD [52]	80.8 ± 0.5	76.1 ± 0.3	61.0 ± 0.4	39.7 ± 0.4	31.9 ± 0.7	57.9	0	1
MTL [6]	80.1 ± 0.8	75.2 ± 0.3	59.9 ± 0.5	40.4 ± 1.0	35.0 ± 0.0	58.1	0	0
ERM [60]	79.8 ± 0.4	75.8 ± 0.2	60.6 ± 0.2	38.8 ± 1.0	35.3 ± 0.1	58.1	1	-
MixStyle [74]	82.6 ± 0.4	75.2 ± 0.7	59.6 ± 0.8	40.9 ± 1.1	33.9 ± 0.1	58.4	1	1
MLDG [38]	81.3 ± 0.2	75.2 ± 0.3	60.9 ± 0.2	40.1 ± 0.9	35.4 ± 0.0	58.6	1	1
Mixup [67]	79.2 ± 0.9	76.2 ± 0.3	61.7 ± 0.5	42.1 ± 0.7	34.0 ± 0.0	58.6	2	2
Fishr [51]	81.3 ± 0.3	76.2 ± 0.3	60.9 ± 0.3	42.6 ± 1.0	34.2 ± 0.3	59.0	2	2
SagNet [47]	81.7 ± 0.6	75.4 ± 0.8	62.5 ± 0.3	40.6 ± 1.5	35.3 ± 0.1	59.1	1	2
SelfReg [34]	81.8 ± 0.3	76.4 ± 0.7	62.4 ± 0.1	41.3 ± 0.3	34.7 ± 0.2	59.3	2	3
Fish [57]	82.0 ± 0.3	76.9 ± 0.2	62.0 ± 0.6	40.2 ± 0.6	35.5 ± 0.0	59.3	3	4
CORAL [58]	81.7 ± 0.0	75.5 ± 0.4	62.4 ± 0.4	41.4 ± 1.8	36.1 ± 0.2	59.4	2	3
SD [50]	81.9 ± 0.3	75.5 ± 0.4	62.9 ± 0.2	42.0 ± 1.0	36.3 ± 0.2	59.7	4	4
Ours	83.8 ± 0.3	76.9 ± 0.6	62.0 ± 0.2	43.2 ± 0.5	34.9 ± 0.1	60.2	4	4

$$\min_{\Theta} \|f_w(z - z')\|, \text{ s.t. } \{z, z'\} = f_\Theta(f_\theta(x)). \quad (8)$$

Note that different from the training phase, x in this stage is from the target domain $\mathcal{D}_t$, and we use the online setting in [59] for updating. A simple illustration of the test adaptation pipeline is shown in Figure 3.

For the final step, we use the original representation obtained from the pretrained feature extractor and the adapted adaptive parameters for prediction. Pseudo code of the test stage are shown in Algorithm 2.

4. Experiments

4.1. Settings

Datasets. We evalute ITTA on five benchmark datasets: **PACS** [37] which consists of 9,991 images from 7 categories. This dataset is probably the most widely-used DG benchmark owing to its large distributional shift across 4 domains including art painting, cartoon, photo, and sketch; **VLCS** [18] contains 10,729 images of 5 classes from 4 different datasets (*i.e.* domains) including PASCAL VOC 2007 [17], LabelMe [53], Caltech [19], and Sun [63] where each dataset is considered a domain in DG; **OfficeHome** [61] is composed of 15,588 images from 65 classes in office and home environments, and those images can be categorized into 4 domains (*i.e.* artistic, clipart, product, and real world); **TerraInc** [4] has 24,788 images from 10 classes. Those images are wild animals taken from 4 different locations (*i.e.* domains) including L100, L38, L43, and L46; **DomainNet** [49] which contains 586,575 images from 345 classes, and the images in it can be depicted in 6 styles (*i.e.* clipart, infograph, painting, quickdraw, real, and sketch).

Implementation details. For all the experiments, we use the ImageNet [15] pretrained ResNet18 [30] backbone that with 4 blocks as the feature extractor f_θ, which could enlarge the gaps in DG compared to larger models [69]. Correspondingly, we also include 4 blocks of additional adaptive parameters (*i.e.* f_Θ), and each block is implemented with 5 layers of learnable parameters with weight initialized as all ones and bias initialized as all zeros. For the weight subnetwork f_w, we use 10 layers of learnable parameters with the initialization skill similar to that of f_Θ. The classifier f_ϕ is an MLP layer provided by the Domainbed benchmark [27]. For the weight parameter α in Eq. (2), we set it to be 1 for all experiments (please refer to our supplementary material for analysis). The random seeds, learning rates, batch size, and augmentation skills are all dynamically set for all the compared arts according to [27].

Table 2. Single source domain generalization. Experiments are conducted on the PACS dataset [37]. Here A, C, P, and S are the art, cartoon, photo, and sketch domains in PACS. A→C represents models trained on the art domain and tested on the cartoon domain, and similar for others. All methods are examined for 60 trials in each unseen domain. Best results are colored as red.

	A→C	A→P	A→S	C→A	C→P	C→S	P→A	P→C	P→S	S→A	S→C	S→P	Avg.
RSC	66.3±1.3	88.2±0.6	57.2±3.1	65.8±1.5	82.4±0.6	68.7±2.5	60.5±2.0	41.3±6.0	53.1±2.8	53.8±1.6	65.9±0.7	48.4±1.9	62.6
Fish	67.1±0.5	89.2±1.8	57.0±0.2	66.7±1.0	85.6±0.4	64.5±3.6	55.1±2.1	33.9±2.3	51.2±4.2	59.1±3.2	67.1±0.9	58.4±1.2	62.9
CDANN	66.5±1.7	92.2±0.6	65.0±0.9	70.6±0.1	82.9±1.4	67.7±3.0	60.6±0.3	42.2±6.4	46.9±9.9	51.4±2.3	60.7±1.2	51.9±0.4	63.2
SelfReg	63.9±1.9	90.1±1.0	56.8±2.2	70.2±2.3	85.4±0.3	70.2±2.2	60.9±2.6	38.8±4.0	50.5±3.2	54.5±4.7	66.2±1.2	51.7±4.1	63.3
DANN	67.5±1.6	91.2±1.3	67.5±1.3	70.6±1.0	81.4±0.4	66.6±1.1	54.1±2.3	33.5±2.7	52.8±2.3	53.8±1.7	64.4±0.7	58.9±0.8	63.5
CAD	67.1±1.5	89.6±0.4	60.2±0.2	67.7±3.1	83.7±1.4	70.2±2.6	60.6±2.6	38.3±3.7	53.8±3.2	50.7±1.6	65.8±1.3	54.4±1.7	63.5
GroupGRO	66.5±1.2	90.5±1.5	58.9±2.5	70.8±0.9	85.7±1.2	69.7±1.8	62.3±2.1	41.1±2.7	48.2±4.1	54.8±0.5	65.2±1.6	53.9±1.4	64.0
MTL	67.3±1.0	90.1±1.0	58.9±0.7	70.2±1.8	84.2±2.2	71.9±0.7	58.3±2.7	38.5±2.7	52.8±1.5	55.4±3.1	66.1±1.3	55.2±2.6	64.1
IRM	67.5±1.8	93.0±0.5	62.9±4.7	67.6±1.3	83.8±0.4	68.9±0.8	63.7±1.8	39.9±3.7	49.0±5.4	54.9±1.4	63.1±2.1	54.9±1.4	64.1
ARM	66.0±2.4	91.2±0.7	58.7±6.9	70.6±0.8	84.2±1.0	69.1±0.9	59.2±1.8	42.1±5.6	52.1±3.0	60.0±0.6	62.9±3.3	53.8±2.0	64.2
Mixup	65.5±0.8	87.8±0.3	57.2±1.0	71.4±1.1	83.1±1.8	68.0±3.0	59.6±1.7	37.2±2.7	56.5±3.8	55.0±2.2	66.2±1.5	62.7±4.2	64.2
CORAL	66.8±0.5	90.3±0.7	61.5±1.9	67.9±2.1	85.4±0.3	70.4±1.3	55.9±2.9	40.4±4.9	49.8±8.5	55.8±2.1	67.6±0.9	58.9±3.8	64.2
SD	67.1±1.3	91.7±1.2	63.7±4.1	70.3±0.9	84.4±0.7	69.4±2.3	57.5±2.5	42.6±0.8	47.7±1.7	55.9±2.4	65.7±0.8	55.8±2.1	64.3
MMD	67.1±1.4	88.0±0.8	63.6±1.6	70.0±1.1	83.6±0.2	70.2±1.0	58.8±2.6	40.3±1.0	52.3±2.4	57.4±1.9	68.7±0.9	52.7±3.7	64.4
MLDG	67.3±2.0	90.8±0.5	64.4±0.9	70.8±1.0	84.2±0.3	69.7±1.8	61.6±1.0	41.3±5.1	50.4±0.2	49.9±2.5	66.8±0.4	58.7±3.4	64.7
CondCAD	66.9±1.4	92.3±0.7	60.8±4.5	71.0±0.6	84.7±1.1	72.6±0.5	61.2±1.5	40.7±3.6	55.7±1.6	52.3±1.7	64.2±0.4	55.3±1.2	64.8
ERM	67.3±0.7	91.7±0.9	60.1±4.7	70.4±0.6	82.3±2.7	68.1±0.9	59.6±1.8	44.7±2.8	56.5±2.7	52.8±2.3	68.1±0.7	58.4±0.9	65.0
VREx	67.1±1.5	91.0±1.0	62.6±3.5	71.1±2.4	84.1±0.9	71.7±1.3	62.4±3.1	37.7±3.3	53.6±2.3	60.6±1.6	66.7±0.8	57.5±1.4	65.5
Fishr	67.9±1.9	92.7±0.3	62.4±4.7	71.2±0.5	83.4±0.6	70.2±1.1	60.0±2.3	42.7±3.2	57.1±3.9	55.7±3.7	68.4±1.0	62.0±3.1	66.1
SagNet	67.6±1.4	92.3±0.5	59.5±1.7	71.8±0.3	82.8±0.6	69.9±1.8	62.5±2.5	45.2±2.5	64.1±2.0	55.8±1.1	65.7±1.4	55.9±3.5	66.1
MixStyle	68.5±2.0	91.2±1.6	65.1±0.7	73.2±1.3	85.0±0.8	71.7±1.5	63.6±1.7	46.3±1.1	51.6±3.7	54.2±1.5	67.0±3.4	58.3±1.4	66.3
Ours	68.9±0.6	92.4±0.1	62.5±0.6	75.3±0.4	85.9±0.3	70.2±1.4	66.5±1.1	52.2±2.7	63.8±1.1	57.6±3.7	68.0±1.3	57.9±2.0	68.4

Training and evaluation details. For all the compared methods, we conduct 60 trials on each source domain, and each with 5,000 iteration steps. During the training stage, we split the examples from training domains to 8:2 (train:val) where the training and validation samples are dynamically selected among different training trials. During test, we select the model that performs the best in the validation samples and test it on the target domains. The strategy is referred to as the "training-domain validate set" model selection method in [27]. For each domain in different datasets, the final performance is the average accuracy from the 60 trials.

4.2. Multi-Source Generalization

In these experiments, all five benchmark datasets aforementioned are used for evaluation, and the leave-one-out strategy is adopted for training (*i.e.* with $S = |\mathcal{D}_s \cup \mathcal{D}_t|^2 - 1$, and $T = 1$). Results are shown in Table 1. We note that ERM method obtains favorable performance against existing arts. In fact, as a strong baseline, ERM is superior to half of the methods in the term of average accuracy, and only 5 arts (*i.e.* SelfReg [34], Fish [57], CORAL [58], SD [50], and ours) among the compared 22 methods outperforms ERM in most datasets (*i.e.* with Score ≥ 3). In comparison, the proposed ITTA is more effective than all other models on average. In particular, ITTA achieves the best performances in 3 out of the 5 benchmarks (*i.e.* PACS, VLCS, and TerraInc datasets) and 4 in the top 5. Note that although our method does not obtain the best performances in the OfficeHome and DomainNet benchmarks, it still outperforms more than half of the existing models. The results validate the effectiveness of our method when tested in the multi-source setting. We present results of average accuracy in each domain from different datasets in the supplementary material. Please refer to it for details.

[2]We use $|\cdot|$ to denote the number of domains in the environment.

4.3. Single-Source Generalization

In these experiments, we adopt the widely-used PACS [37] benchmark for evaluation, and the models are trained on one domain while tested on the remaining three (*i.e.* with $S = 1$, and $T = 3$). Although some approaches, such as MLDG [38] and Fishr [51], may require more than one domain information for their trainings, we can simulate multi-domain information using only the source domain, and thus the experimental settings are still feasible for them. Compared to the multi-source generalization task, the single-source generalization is considered more difficult due to the limited domain information during the training phase. Evaluation results are presented in Table 2. We note that the ERM method outperforms most state-of-the-art models, and only 5 models, including VREx [36], Fishr [51], SagNet [47], MixStyle [74], and the proposed ITTA, can obtain better results than ERM in the term of average accuracy. Meanwhile, our method achieves the best performances when trained in 5 out of the 12 source domain, and it obtains the best performance on average, leading more than 2% than the second best (*i.e.* MixStyle [74]) and 3% the ERM method.

In line with the findings in [27], we notice that the naive ERM method [60] can indeed perform favorably against most existing models under rigorous evaluation protocol. As a matter of fact, the proposed method is the only one that consistently outperforms ERM in both the multi-source and single-source settings. These results indicate that DG

Table 3. Evaluations of different TTT-based models in the unseen domain from PACS [37]. The reported accuracies (%) and standard deviations are computed from 60 trials in each target domain.

Model	Target domain				Avg.
	Art	Cartoon	Photo	Sketch	
Baseline	79.9±0.5	75.4±1.1	94.4±0.5	75.8±1.2	81.4±0.5
TTT [59]	81.5±0.8	77.6±0.6	94.3±0.2	78.4±0.7	83.0±0.2
MT3 [3]	82.0±1.0	76.5±1.0	94.1±0.2	77.7±1.3	82.6±0.6
TENT [62]	80.2±0.9	77.2±0.8	94.4±0.2	77.4±0.1	82.3±0.5
Ours	84.7±0.4	78.0±0.4	94.5±0.4	78.2±0.3	83.8±0.3

remains challenging for current efforts that aim to ease the distribution shift only through training data, and using the proposed improved TTT strategy may be a promising direction for solving DG.

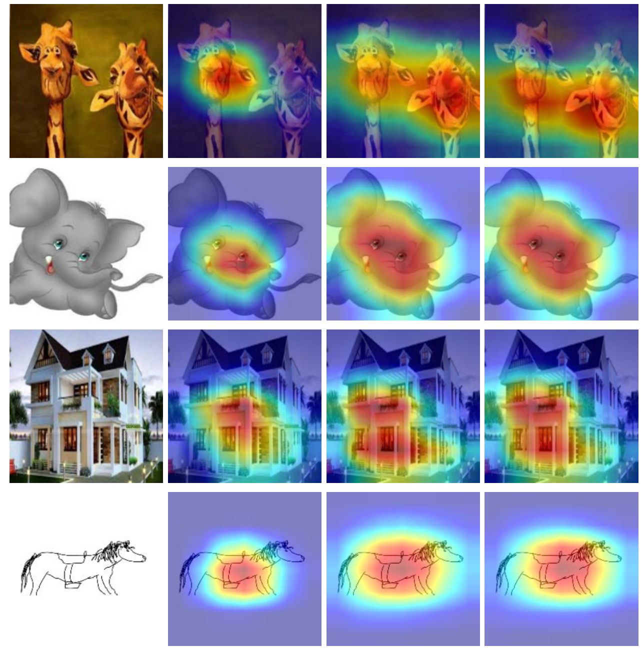
5. Analysis

All experiments in this section are conducted on the widely-used PACS benchmark [37] with the leave-one-out strategy. The experimental settings are the same as that illustrated in Sec. 4.1. Please refer to our supplementary material for more analysis.

5.1. Compared with Other TTT-Based Models

Using test-time adaptation to ease the distribution shift problem has been explored in previous works, such as the original TTT method [59] and MT3 [3]. Their differences lie in that TTT uses a rotation estimation task for the test-time objective, and MT3 adopts a contrastive loss for the task and implements the overall framework using MAML [20]. There is also a recently proposed TENT [62] that aims to minimize the entropy of the final results by tuning the parameters from the batch normalization (BN) layers. To analyze the overall effectiveness of our method, we compare ITTA with these arts using the same baseline (*i.e.* ResNet18 [30] backbone with the existing augmentation skill [74]).

Results are shown in Table 3. We observe that all the compared TTT-based methods can improve the baseline model in almost all target domains except for the "Photo" domain, which might be due to the ImageNet pretraining [66]. This phenomenon demonstrates that the TTT strategy may be a promising effort for easing the distribution shift problem. Meanwhile, we observe that the proposed ITTA is superior to all other approaches in most target domains and leads in the term of average accuracy. The main reason is that compared to the empirically designed TTT tasks adopted in previous works, the proposed learnable consistency loss is enforced to be more aligned with the main loss, thus more suitable for the test-time adaptation task [45]. Meanwhile, compared to the strategies that update the original parameters from the trained model, the adaptation of the newly included parameters is also more effective for the overall TTT framework. In the following, we provide more analysis to support these claims.



(a) Input (b) Ours w/o f_w (c) Ours (d) Main

Figure 4. Grad-CAM [56] visualizations from different loss terms. We use images with varying class labels from the four target domains of PACS [37] as inputs (*i.e.* art, cartoon, photo, and sketch domains from top to bottom). Ours w/o f_w is the naive consistency loss with f_w disabled in Eq. (1). The proposed learnable consistency loss can align well with the main classification task.

5.2. Effectiveness of the Learnable Consistency Loss

To examine the effectiveness of our learnable consistency loss, we conduct ablation studies by comparing our method with the following variants. (1) Ours w/o f_w: we disable f_w when computing the learnable consistency loss in Eq. (1), which uses the naive consistency loss for the auxiliary TTT task. (2) Ours w/ Ent.: after training the model using the baseline settings (*i.e.* ResNet18 with the augmentation strategy [74]), we use the entropy minimization task in [62] for the TTT task. (3) Ours w/ Rot.: we use the rotation estimation task in [59] for the TTT task. To ensure fair comparisons, we use the same baseline settings and include the same additional adaptive parameters for all the variants.

Results are shown in the 4th to 6th rows Table 4. We find that the results from the naive consistency loss (*i.e.* Ours w/o f_w) are slightly better than that from the other two specially-designed objectives (*i.e.* Ours w/ Ent. and Ours w/ Rot.) on average. Besides the possibility of deteriorating the performance [45], our results indicate that empirically selecting a TTT task may also be far from optimal. Meanwhile, we observe that when enabling f_w, the proposed learnable consistency loss is superior to that without f_w in all target domains, and it leads in the term of average accuracy among the variants compared, illustrating its advantage against other

Table 4. Comparison between different TTT tasks and parameter selecting strategies in the unseen domain from the PACS benchmark [37]. Here the "Ent.", "Rot.", and "$\mathcal{L}_{wcont}$" denotes the entropy minimization task in [62], the rotation estimation task in [59], and the proposed learnable consistency objective, the "All", "BN", and "Ada." are the strategies that update all the parameters, parameters from the batch normalization layer, and the proposed strategy that updates only the new additional adaptive parameters. The reported accuracies (%) and standard deviations are computed from 60 trials in each target domain.

Model	TTT tasks			Param selectings			Target domain				Avg.
	Ent.	Rot.	$\mathcal{L}_{wcont}$	All	BN	Ada.	Art	Cartoon	Photo	Sketch	
Ours	–	–	✓	–	–	✓	84.7±0.4	78.0±0.4	94.5±0.4	78.2±0.3	83.8±0.3
Ours w/o f_w	–	–	–	–	–	✓	83.1±0.4	74.6±0.6	94.0±0.5	78.0±0.8	82.5±0.1
Ours w/ Ent.	✓	–	–	–	–	✓	79.9±2.4	77.3±0.3	94.8±0.8	77.6±0.4	82.4±0.8
Ours w/ Rot.	–	✓	–	–	–	✓	81.1±1.0	75.2±0.5	94.9±0.3	77.3±0.6	82.1±0.3
Ours w/o TTT	–	–	✓	–	–	–	83.3±0.5	76.0±0.5	94.4±0.5	76.7±1.4	82.8±0.3
Ours w/ All	–	–	✓	✓	–	–	83.0±0.7	77.0±1.4	94.5±0.7	77.4±0.9	83.0±0.2
Ours w/ BN	–	–	✓	–	✓	–	81.8±0.5	75.6±0.3	94.4±0.3	77.9±1.1	82.4±0.5

adopted TTT tasks. These results are not surprising. By comparing the Grad-CAM [56] visualizations from the main classification task with the learnable and naive consistency losses in Figure 4, we find that the proposed learnable objective can well align with the main loss when f_w is enabled as the hot zones activated by these two tasks are similar, which guarantees the improvement for the test-time adaptation [45, 59]. Please refer to our supplementary material for more visualizations.

5.3. Effectiveness of the Adaptive Parameters

We compare ITTA with three variants to demonstrate the effectiveness of the proposed additional adaptive parameters. (1) Ours w/o TTT: we do not update any parameters during the test phase. This variant is used to verify whether TTT can improve the pretrained model. (2) Ours w/ ALL: similar to the updating strategy in the original TTT method [59], we update all the parameters from the feature extractor during the test phase. (3) Ours w/ BN: following the suggestion from TENT [62], only parameters from the BN layers of the feature extractor are updated. Note the same pretrained model is shared for all variants in these experiments, and the objectives during the test adaptation phase are to minimize the same learned consistency loss.

We list the results in the last three rows in Table 4. We observe that when only updating parameters from the BN layers, the performance is inferior to the strategy without test-time adaptation, and updating all the parameters does not ensure improvements in all target domains. The observations are in line with the findings in [62] that selecting reliable parameters to update is essential in the TTT system and may also interact with the choice of the TTT task. In comparison, when including additional adaptive parameters for updating, the pretrained model can be boosted in all environments. The results validate that our adaptive parameters are more effective than that selected with existing strategies [59, 62] when applied with the proposed learnable test-time objective.

5.4. Limitation

Although the proposed learned loss can bring satisfaction improvements, we are aware that the lunch is not free. When the weight subnetwork f_w is disabled, updating the joint loss in Eq. (2) only costs 1 forward and 1 backward. However, in order to update f_w, we have to compute the second-order derivative in Eq. (5), which will require 1 more forward and 3 more backward processes, bringing extra burden to the system. Our future efforts aim to simplify the overall optimization process and reduce the cost for ITTA.

6. Conclusion

In this paper, we aim to improve the current TTT strategy for alleviating the distribution shift problem in DG. First, given that the auxiliary TTT task plays a vital role in the overall framework, and an empirically selecting one that does not align with the main task may potentially deteriorate instead of improving the performance, we propose a learnable consistency loss that can be enforced to be more aligned with the main loss by adjusting its learnable parameters. This strategy is ensured to improve the model and shows favorable performance against some specially-designed objectives. Second, considering that selecting reliable and effective parameters to update during the test phase is also essential while exhaustively trying different combinations may require tremendous effort, we propose a new alternative by including new additional adaptive parameters for adaptation during the test phase. This alternative is shown to outperform some previous parameter selecting strategies via our experimental findings. By conducting extensive experiments under a rigorous evaluation protocol, we show that our method can achieve superior performance against existing arts in both the multi-source and single-source DG tasks.

Acknowledgements. Liang Chen is supported by the China Scholarship Council (CSC Student ID 202008440331).

References

[1] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019. 5

[2] Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. Metareg: Towards domain generalization using meta-regularization. In *NeurIPS*, 2018. 1, 2

[3] Alexander Bartler, Andre Bühler, Felix Wiewel, Mario Döbler, and Bin Yang. Mt3: Meta test-time training for self-supervised test-time adaption. In *AISTATS*, 2022. 2, 4, 7

[4] Sara Beery, Grant Van Horn, and Pietro Perona. Recognition in terra incognita. In *ECCV*, 2018. 5

[5] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In *NeurIPS*, 2006. 2

[6] Gilles Blanchard, Aniket Anand Deshmukh, Urun Dogan, Gyemin Lee, and Clayton Scott. Domain generalization by marginal transfer learning. *arXiv preprint arXiv:1711.07910*, 2017. 5

[7] Gilles Blanchard, Gyemin Lee, and Clayton Scott. Generalizing from several related classification tasks to a new unlabeled sample. In *NeurIPS*, 2011. 1

[8] Chaoqi Chen, Jiongcheng Li, Xiaoguang Han, Xiaoqing Liu, and Yizhou Yu. Compound domain generalization via meta-knowledge encoding. In *CVPR*, 2022. 1

[9] Chaoqi Chen, Luyao Tang, Feng Liu, Gangming Zhao, Yue Huang, and Yizhou Yu. Mix and reason: Reasoning over semantic topology with data mixing for domain generalization. In *NeurIPS*, 2022. 1

[10] Dian Chen, Dequan Wang, Trevor Darrell, and Sayna Ebrahimi. Contrastive test-time adaptation. In *CVPR*, 2022. 2

[11] Liang Chen, Yong Zhang, Yibing Song, Lingqiao Liu, and Jue Wang. Self-supervised learning of adversarial example: Towards good generalizations for deepfake detection. In *CVPR*, 2022. 2

[12] Liang Chen, Yong Zhang, Yibing Song, Jue Wang, and Lingqiao Liu. Ost: Improving generalization of deepfake detection via one-shot test-time training. In *NeurIPS*, 2022. 2

[13] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. 2, 3

[14] Sungha Choi, Seunghan Yang, Seokeon Choi, and Sungrack Yun. Improving test-time adaptation via shift-agnostic weight regularization and nearest source prototypes. In *ECCV*, 2022. 2

[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 5

[16] Qi Dou, Daniel Coelho de Castro, Konstantinos Kamnitsas, and Ben Glocker. Domain generalization via model-agnostic learning of semantic features. In *NeurIPS*, 2019. 1, 2

[17] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, 2010. 5

[18] Chen Fang, Ye Xu, and Daniel N Rockmore. Unbiased metric learning: On the utilization of multiple datasets and web images for softening bias. In *ICCV*, 2013. 5

[19] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *CVPR worksho*, 2004. 5

[20] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017. 2, 7

[21] Francois Fleuret et al. Uncertainty reduction for model adaptation in semantic segmentation. In *CVPR*, 2021. 2

[22] Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei A Efros. Test-time training with masked autoencoders. In *NeurIPS*, 2022. 2

[23] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *JMLR*, 17(1):2096–2030, 2016. 1, 2, 5

[24] Muhammad Ghifary, David Balduzzi, W Bastiaan Kleijn, and Mengjie Zhang. Scatter component analysis: A unified framework for domain adaptation and domain generalization. *IEEE TPAMI*, 39(7):1414–1430, 2016. 1

[25] Muhammad Ghifary, W Bastiaan Kleijn, Mengjie Zhang, and David Balduzzi. Domain generalization for object recognition with multi-task autoencoders. In *ICCV*, 2015. 2

[26] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. In *NeurIPS*, 2020. 2, 3

[27] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In *ICLR*, 2021. 1, 2, 5, 6

[28] Sivan Harary, Eli Schwartz, Assaf Arbelle, Peter Staar, Shady Abu-Hussein, Elad Amrani, Roei Herzig, Amit Alfassy, Raja Giryes, Hilde Kuehne, et al. Unsupervised domain generalization by learning a bridge across domains. In *CVPR*, 2022. 1

[29] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 2, 3

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

[31] Shoubo Hu, Kun Zhang, Zhitang Chen, and Laiwan Chan. Domain generalization via multidomain discriminant analysis. In *UAI*, 2020. 1

[32] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017. 2

[33] Zeyi Huang, Haohan Wang, Eric P Xing, and Dong Huang. Self-challenging improves cross-domain generalization. In *ECCV*, 2020. 5

[34] Daehee Kim, Youngjun Yoo, Seunghyun Park, Jinkyu Kim, and Jaekoo Lee. Selfreg: Self-supervised contrastive regularization for domain generalization. In *ICCV*, 2021. 2, 5, 6

[35] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *ICML*, 2021. 1

[36] David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Dinghuai Zhang, Remi Le Priol, and Aaron Courville. Out-of-distribution generalization via risk extrapolation (rex). In *ICML*, 2021. 5, 6

[37] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *ICCV*, 2017. 1, 5, 6, 7, 8

[38] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Learning to generalize: Meta-learning for domain generalization. In *AAAI*, 2018. 1, 2, 5, 6

[39] Da Li, Jianshu Zhang, Yongxin Yang, Cong Liu, Yi-Zhe Song, and Timothy M Hospedales. Episodic training for domain generalization. In *ICCV*, 2019. 1, 2

[40] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain generalization with adversarial feature learning. In *CVPR*, 2018. 1, 2, 5

[41] Pan Li, Da Li, Wei Li, Shaogang Gong, Yanwei Fu, and Timothy M Hospedales. A simple feature augmentation for domain generalization. In *ICCV*, 2021. 1, 2

[42] Xiaotong Li, Yongxing Dai, Yixiao Ge, Jun Liu, Ying Shan, and Ling-Yu Duan. Uncertainty modeling for out-of-distribution generalization. In *ICLR*, 2022. 1, 2

[43] Yizhuo Li, Miao Hao, Zonglin Di, Nitesh Bharadwaj Gundavarapu, and Xiaolong Wang. Test-time personalization with a transformer for human pose estimation. In *NeurIPS*, 2021. 2, 3, 4

[44] Ya Li, Xinmei Tian, Mingming Gong, Yajing Liu, Tongliang Liu, Kun Zhang, and Dacheng Tao. Deep domain generalization via conditional invariant adversarial networks. In *ECCV*, 2018. 1, 2, 5

[45] Yuejiang Liu, Parth Kothari, Bastien van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. Ttt++: When does self-supervised test-time training fail or thrive? In *NeurIPS*, 2021. 2, 3, 4, 7, 8

[46] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant feature representation. In *ICML*, 2013. 1, 2

[47] Hyeonseob Nam, HyunJae Lee, Jongchan Park, Wonjun Yoon, and Donggeun Yoo. Reducing domain gap by reducing style bias. In *CVPR*, 2021. 2, 5, 6

[48] Prashant Pandey, Mrigank Raman, Sumanth Varambally, and Prathosh Ap. Generalization on unseen domains via inference-time label-preserving target projections. In *CVPR*, 2021. 1

[49] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *ICCV*, 2019. 5

[50] Mohammad Pezeshki, Oumar Kaba, Yoshua Bengio, Aaron C Courville, Doina Precup, and Guillaume Lajoie. Gradient starvation: A learning proclivity in neural networks. In *NeurIPS*, 2021. 1, 5, 6

[51] Alexandre Rame, Corentin Dancette, and Matthieu Cord. Fishr: Invariant gradient variances for out-of-distribution generalization. In *ICML*, 2022. 1, 2, 5, 6

[52] Yangjun Ruan, Yann Dubois, and Chris J Maddison. Optimal representations for covariate shift. In *ICLR*, 2022. 5

[53] Bryan C Russell, Antonio Torralba, Kevin P Murphy, and William T Freeman. Labelme: a database and web-based tool for image annotation. *IJCV*, 77(1):157–173, 2008. 5

[54] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. In *ICLR*, 2020. 5

[55] Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. In *NeurIPS*, 2020. 2

[56] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, 2017. 7, 8

[57] Yuge Shi, Jeffrey Seely, Philip HS Torr, N Siddharth, Awni Hannun, Nicolas Usunier, and Gabriel Synnaeve. Gradient matching for domain generalization. In *ICLR*, 2021. 1, 2, 5, 6

[58] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *ECCV*, 2016. 5, 6

[59] 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 *ICML*, 2020. 1, 2, 4, 5, 7, 8

[60] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 1999. 1, 5, 6

[61] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *CVPR*, 2017. 5

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

[63] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010. 5

[64] Zehao Xiao, Xiantong Zhen, Ling Shao, and Cees GM Snoek. Learning to generalize across domains on single test samples. In *ICLR*, 2022. 2

[65] Qinwei Xu, Ruipeng Zhang, Ya Zhang, Yanfeng Wang, and Qi Tian. A fourier-based framework for domain generalization. In *CVPR*, 2021. 1, 2

[66] Zhenlin Xu, Deyi Liu, Junlin Yang, Colin Raffel, and Marc Niethammer. Robust and generalizable visual representation learning via random convolutions. In *ICLR*, 2021. 7

[67] Shen Yan, Huan Song, Nanxiang Li, Lincan Zou, and Liu Ren. Improve unsupervised domain adaptation with mixup training. *arXiv preprint arXiv:2001.00677*, 2020. 2, 5

[68] Fu-En Yang, Yuan-Chia Cheng, Zu-Yun Shiau, and Yu-Chiang Frank Wang. Adversarial teacher-student representation learning for domain generalization. In *NeurIPS*, 2021. 1, 2

[69] Nanyang Ye, Kaican Li, Haoyue Bai, Runpeng Yu, Lanqing Hong, Fengwei Zhou, Zhenguo Li, and Jun Zhu. Ood-bench: Quantifying and understanding two dimensions of out-of-distribution generalization. In *CVPR*, 2022. 5

[70] Fuming You, Jingjing Li, and Zhou Zhao. Test-time batch statistics calibration for covariate shift. *arXiv preprint arXiv:2110.04065*, 2021. 4

[71] Marvin Zhang, Henrik Marklund, Nikita Dhawan, Abhishek Gupta, Sergey Levine, and Chelsea Finn. Adaptive risk minimization: A meta-learning approach for tackling group distribution shift. *arXiv preprint arXiv:2007.02931*, 2020. 5

[72] Marvin Zhang, Henrik Marklund, Nikita Dhawan, Abhishek Gupta, Sergey Levine, and Chelsea Finn. Adaptive risk minimization: Learning to adapt to domain shift. *NeurIPS*, 2021. 2

[73] Tao Zhong, Zhixiang Chi, Li Gu, Yang Wang, Yuanhao Yu, and Jin Tang. Meta-dmoe: Adapting to domain shift by meta-distillation from mixture-of-experts. In *NeurIPS*, 2022. 2

[74] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain generalization with mixstyle. In *ICLR*, 2021. 1, 2, 3, 5, 6, 7