Instance-aware Test-Time Adaptation for Domain Generalization

Taki Hasan Rafi
Hanyang University
Seoul, Republic of Korea
takihr@hanyang.ac.kr

Serbeter Karlo
University of Cambridge
Cambridge, United Kingdom
serbetar.karlo@protonmail.com

Amit Agarwal
Oracle Inc.
Seattle, USA
amit.h.agarwal@oracle.com

Hitesh L. Patel
Oracle Inc.
New York, USA
hitesh.laxmichand.patel@oracle.com

Bhargava Kumar
TD Securities
New York, USA
Bhargava.kumar@tdsecurities.com

Dong-Kyu Chae*
Hanyang University
Seoul, Republic of Korea
dongkyu@hanyang.ac.kr

Abstract

Domain generalization (DG) aims to enhance the generalization capability of models to unseen target distributions by leveraging multiple source distributions. This paper focuses on robust test-time adaptation (TTA) based DG that mostly need when the model experiences new unseen domains while testing. We consider two practical challenges under domain shifts: (1) existing DG methods mostly rely on domain-specific information, and not explicitly utilizing class-specific information. Therefore, these approaches ignore mixed features, that are both class and domain-specific, thus resulting in the loss of useful information; (2) while existing TTA methods explicitly require a memory bank of test time samples, which is computationally complex and impractical in many applications. To overcome these limitations, we propose a new framework called IADG that utilizes class-specific information with domain-specific information to ensure robust generalization. Our method exploits disentangled features by pulling class-relevant features to increase diversified negative pairs, facilitating flawless integration of class and domain-specific features. To leverage high-confidence samples during testing, we introduce a novel confidence-guided low-risk instance TTA approach that only considers one unlabeled sample during inference and therefore does not require a dedicated memory bank. Extensive evaluations on five public benchmarks consistently demonstrate the superior performance of our approach over the state-of-the-art. Project code can be found here: https://github.com/takihasan/IADG.

CCS Concepts

• Computing methodologies → Generalization; Transfer learning; Distribution shift; **• Mathematics of computing → Classification**.

*Corresponding Author.

Keywords

domain generalization, domain-invariance, contrastive learning, disentanglement, test-time adaptation

ACM Reference Format:
Taki Hasan Rafi, Serbeter Karlo, Amit Agarwal, Hitesh L. Patel, Bhargava Kumar, and Dong-Kyu Chae. 2025. Instance-aware Test-Time Adaptation for Domain Generalization. In . DASFAA '25, Singapore, Singapore, 13 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

The majority of machine learning models operate under the assumption that the training and testing data adhere to the principle of being independent and identically distributed (i.i.d). Nonetheless, this assumption is rarely observed in the real world. The deployment of models on unseen domains often leads to significant performance degradation, underscoring the challenge posed by the discrepancy between training and testing domains, that also referred to as domain shift. In response to this challenge, domain adaptation (DA) and domain generalization (DG) methods have been proposed [25, 31, 32, 53].

While these methods share several concepts and approaches, the key disparity lies in the fact that DG methods do not necessitate test set data during the training phase. In a typical DG setting, the model is trained on one or multiple source domains and then assessed on a distinct test domain. Notably, the predominant focus of DG approaches is on learning features invariant across different domains [52]. DG models typically assume predictions via domain-specific features or general features [45]. Although, some studies [4, 7] have highlighted the potential of incorporating domain-invariant and domain-specific features in improving generalization performance, but not fully explored class and domain-specific features. The pursuit of disentangled representation learning (DRL) has emerged as a potential solution for resolving DG challenges, but most of the works rely on disentangling class and domain invariant representations.

Even though recent works have shown promising results in practice, their main drawback is to ignore class-specific information that could lead to better generalization and retain more information, especially when the number of domains is increased. Furthermore, in practice, different domains can obtain a strong correlation with class-specific features. For example, let's consider that we want to classify a class "elephant" from two different domains such as "photo" and "painting". Even though an "elephant" image belongs to a class of two distinct domains, both domains (e.g., photo and painting) have different class attributes and texture features to represent the class image, respectively. Thus, we need class-specific information to understand the distinguishing information about the class that is absent in the domain-specific information. However, this information can highly be regarded in generalization performance in target domains due to class-wise information even in the same background. So integrating class-specific features with domain-specific features can exhibit more accurate prediction outcomes because both contain strong correlative information both for domains and their belonging classes. On the other hand, multiple attempts are made to handle domain shifts in inference by adopting an online test model during inference. Although they are equipped to handle test shifts, certain limitations still exist. Several prior approaches [8, 42, 47] require a memory bank of test time samples, which can potentially increase complexity during inference.

To address the limitations above, we propose an approach that utilizes both class and domain-specific features with Test-Time Adaptation method IADG for DG, which strategically merges disentangled representation learning (DRL) and contrastive learning (CL) to learn class and domain-specific (mixed) features. Our stance is that excluding mixed features totally might be too strict of a constraint and can lead to reduced generalization ability. IADG uses a single feature extract to obtain a feature vector containing all features. To disentangle these features into two separate representations, one that contains class-specific features and one that contains class-invariant features, we use two small encoder networks and a combination of orthogonality constraint and adversarial regularization. Moreover, to further augment the process of disentanglement and facilitate the acquisition of generalizable features, we use CL to push away all class-irrelevant features, including those encoded by the class-invariant encoder, while at the same time pulling together class-relevant features extracted from separate samples. This style of CL significantly increases the number of negative pairs and helps the model to learn generalizable features. This fusion of DRL and CL empowers IADG to achieve superior representation learning, consequently enhancing performance in DG tasks.

We adapt model parameters to the test domain in a realistic instance-based method to handle domain shifts during inference. During test time, we have access to a single unlabeled example and no training data. We first obtain the pseudo label by averaging the classification results of weakly augmented views of the original test sample. However, due to the noisy nature of pseudo labels, naively using TTA methods can quickly lead to error accumulation and reduced performance. A large number of the previous TTA methods simply keep only the samples with confidence above some fixed threshold [28, 29, 35]. However, setting the right threshold is not a trivial task, and might be different in different domains and training settings. To address this issue, we propose *to keep all the samples, but weigh the losses* with an additional factor that depends on label confidence, which causes the *low-confident samples to not degrade the model parameters too much.* We theorize that this low confidence might positively affect generalization since they represent difficult examples. Inspired by the success of IADG training, which shows that classification loss and contrastive loss can successfully be combined for better results, we add the contrastive loss term during test time. Because we have no access to training data or other test time samples, to construct negative pairs, we use the class centroids. Notably, our framework is an end-to-end Test-Time DG framework. In a nutshell, we highlight the following contributions of IADG:

- **A New Approach for DG.** We propose IADG, which can exploit both class and specific features that maximize generalization on unseen test domains by ensuring robust representations via contrastive-disentangled learning, alleviating class-wise information scarcity that is lacking in previous works.
- **A Novel TTA Approach.** We introduce a novel confidence-guided instance-aware test-time adaptation (TTA) method that does not require a memory bank to avoid additional complexity during inference. It is designed with a confidence-weighted loss function and thereby shows robustness to error accumulation in out-of-distribution environments.
- **Extensive Experiments.** We comprehensively evaluate consistent protocol [15] to show the effectiveness of IADG on standard benchmarks. The evaluation outcome interprets that IADG can achieve competitive performance and outperform other state-of-the-art methods by a considerable margin.

2 Related Works

Domain Generalization. Domain generalization (DG) aims to utilize multiple source domains to generalize to unseen target domains by minimizing statistical discrepancies. Traditional DG methods emphasize learning domain-invariant representations [1, 24] or employing empirical risk minimization [39]. Advanced strategies include meta-learning [12, 23] and self-supervised learning [20], with augmentation [48] also showing promise. Contrastive learning-based approaches, like PCL [46] and SelfReg [20], align representations across classes and domains. While existing methods largely focus on domain-specific features, our framework combines contrastive learning, disentanglement, and test-time adaptation (TTA), enhancing generalization by incorporating both class- and domain-specific features.

Contrastive Learning. Contrastive learning (CL) has revolutionized SSL (self-supervised learning) by differentiating positive and negative sample pairs in the embedding space. SimCLR [10] and MoCo [16] highlight key CL concepts like data augmentation and momentum encoders. In DG, PCL [46] uses proxy-to-sample relationships to address distribution shifts. Since CL depends on samples that are neither too difficult nor too easy to align, we theorize that aligning disentangled class-specific features, containing no class-invariant domain-specific features, will show to be a better generalization objective. Additionally, we theorize that the inclusion of many class-invariant feature vectors as negative samples, which greatly increases the diversity of negative pairs, will have a positive impact on generalization ability.

Test-Time Adaptation. Test-Time Adaptation (TTA) [41] involves updating a pre-trained model at test time to adapt to new data. Techniques like TENT [41] and SHOT [26] adjust BatchNorm layers using entropy minimization but often require careful hyperparameter tuning to avoid errors. Additionally, many of these methods employ a memory bank of test time samples, which increases complexity. Our proposed TTA method requires no memory bank and uses all test samples while avoiding error accumulation by putting less weight on low-confidence testing samples.

3 Method

3.1 Problem Definition

Let $\mathcal{X} \subset \mathbb{R}^d$ be the input space and let $\mathcal{Y} \subset \mathbb{R}$ be the output space. A domain dataset $D = \{\vec{x}^k \in \mathcal{X}, y^k \in \mathcal{Y}\}_{k=1}^N$ consists of N samples independently drawn from a joint distribution P_{XY} on $\mathcal{X}$ and $\mathcal{Y}$. In domain generalization setting, source dataset $\mathcal{D}_s = \{D_s^i\}_{i=1}^S$ comes from S domains and test dataset $\mathcal{D}_t = \{D_t^i\}_{i=1}^T$ comes from T domains. We denote the joint distribution for the i-th source dataset as $P^i_{XY,s}$ and the joint distribution for the i-th target dataset as $P^i_{XY,t}$. It is important to notice that $P^i_{XY} \neq P^j_{XY}$ if $i \neq j$, *i.e.*, joint distribution from which data is sampled is different in each domain. The aim of DG is to find the hypothesis $h_\theta : \mathcal{X} \rightarrow \mathcal{Y}$ parameterized by θ which minimizes the empirical risk:

$$\theta^* =_\theta \frac{1}{T} \sum_i^T \mathbb{E}_{(\vec{x},y)\sim\mathcal{P}^{i,t}_{XY}} [\mathcal{L}(h_\theta(\vec{x}), y)] \tag{1}$$

where $\mathcal{L}$ is the loss function. In practice, P^t_{XY} is intractable so the optimization objective is approximated as:

$$\theta^* =_\theta \frac{1}{T} \sum_i^T \mathbb{E}_{(\vec{x},y)\in D^i_t} [\mathcal{L}(h(\vec{x}), y)] \tag{2}$$

3.2 Bridging Class Domain-Specific Features

Given an input example $\vec{x}$ from the source dataset, we obtain the feature vector $\vec{z} \in \mathbb{R}^Z$ by feeding the image through the feature extractor $F : \mathcal{X} \rightarrow \mathbb{R}^Z : \vec{z} = F(\vec{x})$. Here, feature vector $\vec{z}$ contains both class-specific and class-invariant features. To promote generalization, IADG disentangles the features in $\vec{z}$ into two representations, $\vec{z}_c$ and $\vec{z}_d$. To obtain separate representations $\vec{z}$ is fed into two encoder networks, $E_c : \mathbb{R}^Z \rightarrow \mathbb{R}^Z$ that encodes the class-specific features, and $E_d : \mathbb{R}^Z \rightarrow \mathbb{R}^Z$ that encodes class-invariant features.

$$\vec{z}_c = E_c(\vec{z}) \tag{3}$$

$$\vec{z}_d = E_d(\vec{z}) \tag{4}$$

To enable the learning of relevant features, we add a class label classification loss from $\vec{z}_c$ and domain label classification loss from $\vec{z}_d$. If $\vec{z}_c$ can be used to reliably classify the class label of the input example, it contains the relevant class-specific features. Similarly, if $\vec{z}_d$ can be used to reliably classify the domain label of the input example, it contains the relevant domain-specific features. Thus, our classification losses for example $\vec{x}$ with class label y and domain label d are:

$$\mathcal{L}^c_{cls}(\vec{x}, y) = H(\sigma(G_c(\vec{z}_c))), y) \tag{5}$$

$$\mathcal{L}^d_{cls}(\vec{x}, d) = H(\sigma(G_d(\vec{z}_d))), d) \tag{6}$$

where $G_c : \mathbb{R}^Z \rightarrow \mathbb{R}^C$ is the class label classifier, with C being the number of classes, $G_d : \mathbb{R}^Z \rightarrow \mathbb{R}^D$ is the domain label classifier, with D being the number of training domains, H is the cross entropy function and σ is the softmax function.

Class-Domain Specific Feature Orthogonality. While these losses force the encoder networks E_c and E_d to learn to encode class and domain-specific features respectively, to disentangle the features, we need to impose additional constraints. Since some features might be both class and domain-specific, we need a way to ensure those features are encoded by E_c only and not by E_d, *i.e.*, that encoded representations contain separate features.

As shown in [44], enforcing orthogonality constraints on feature vectors promotes the learning of disentangled features. We impose this constraint by adding a loss term called orthogonality loss, $\mathcal{L}_{orth}$ to our overall loss objective:

$$\mathcal{L}_{orth}(\vec{x}) = s(\vec{z}_c, \vec{z}_d) \tag{7}$$

where $s(\vec{a}, \vec{b}) = \frac{|\vec{a}\cdot\vec{b}|}{|\vec{a}||\vec{b}|}$ is the absolute cosine similarity. We use the absolute sign since we want to force the dot product to be 0 and not -1. Unlike [44], which uses a single network and disentangles the features geometrically, IADG uses two encoders which enables more powerful disentanglement while adding very slight computational overhead since both encoders are small MLP networks. Minimizing $\mathcal{L}_{orth}$ enables encoders to encode disentangled features. As explored in previous work [50], including domain-invariant and class-specific features can help with generalization. Since these features can be extracted by both encoders. The orthogonality constraint only ensures that the feature does not appear in both representations, but it does not guarantee that the mixed feature will be encoded by E_c. To ensure that, we need an additional constraint that would help with pushing all the mixed features into $\vec{z}_c$ and out of $\vec{z}_d$. Since we do not want any class-specific features to be encoded into $\vec{z}_d$, we need to ensure that the class label classifier G_c can not reliably predict the class label, *i.e.*, we want to maximize the entropy of the logit. Hence we add an additional adversarial loss term $\mathcal{L}_{adv}$:

$$\mathcal{L}_{adv}(\vec{x}) = -I(G_c(\vec{z}_d)) \tag{8}$$

where I is the entropy function. Notice that, unlike previous DANN-based methods [13, 51], we do not include the adversarial loss on the logit obtained by pushing $\vec{z}_c$ through the G_d since that would force E_c to not learn any of domain-specific features, including class-specific features which help with generalization.

3.3 Unified Latent Space Contrastive Learning

Prior works have shown that contrastive learning can help with the generalization performance [20, 46]. To leverage the information contained in sample-sample relationships, we add contrastive loss term $\mathcal{L}_{contr}$. Correctly sampling both correct positive and negative pairs is an important part of optimization in the CL setting [20, 43, 46]. Selecting positive pairs that are too difficult to optimize, such as examples from different domains, results in reduced performance. Furthermore, hard negative samples generally help with performance [33]. To increase the quantity and diversity of negative pairs, we consider not only the contrast between class-specific embeddings belonging to different classes but also between class-specific and class-invariant embeddings.

During training, we are given a batch of B examples $\mathcal{B} = \{\vec{x}^i\}_{i=1}^B$ with class label for sample $\vec{x}^i$ being y^i and domain label being d^i. That batch of examples is encoded into disentangled representations

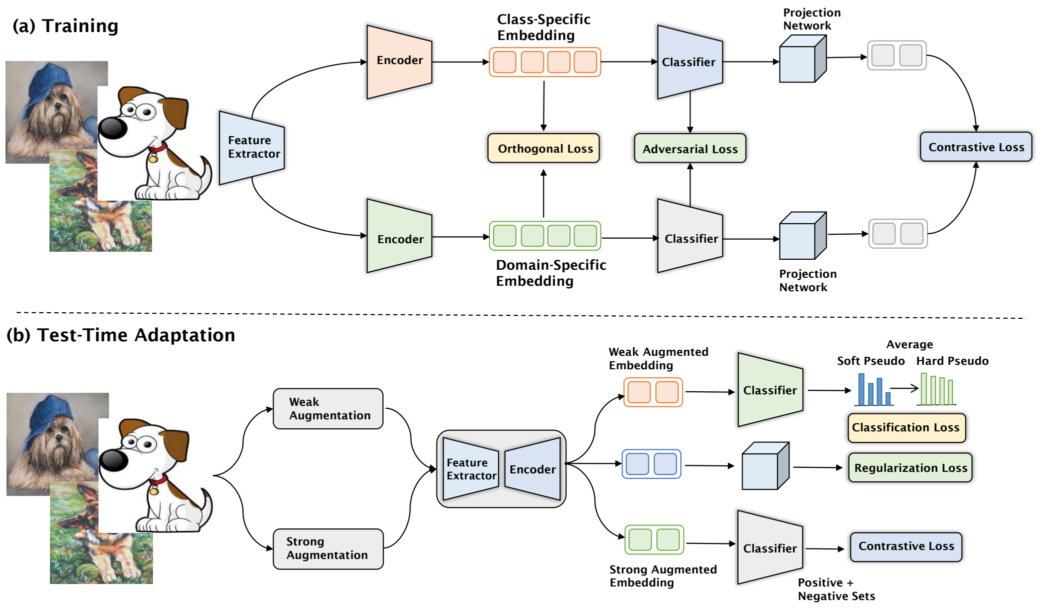

Figure 1: IADG Pipeline. (a) Feature extractor F is used to obtain the feature vector $\vec{z}$ containing class and domain-related information. Class-specific features are encoded by E_c into $\vec{z}_c$ and class-invariant features are encoded by E_d into $\vec{z}_d$. z_c and z_d are forced to be orthogonal to ensure the disentanglement. (b) $\vec{z}_d$ is additionally fed into the class label classifier G_c and trained adversarially. To further improve the generalization performance, feature vectors are projected and trained in a contrastive manner.

to make a batch of class-specific and class-invariant representations. Inspired by SimCLR [10], to prevent representation collapse, we additionally add small projection networks, $P_c : \mathbb{R}^Z \to \mathbb{R}^Z$ for $\vec{z}_c$ and $P_d : \mathbb{R}^Z \to \mathbb{R}^Z$ for $\vec{z}_d$ to obtain the projected feature vectors which we use in the rest of the CL method. Let $\mathcal{B}_c = \{P_c(\vec{z}_c^i)\}_{i=1}^B$ be the set of projected class-specific feature vectors and let $\mathcal{B}_d = \{P_d(\vec{z}_d^i)\}_{i=1}^B$ be the set of projected class-invariant feature vectors. For sample i, we can construct the set of positive pairs as $\mathcal{P}^i = \{p^j | p^j \in \mathcal{B}_c \setminus \vec{z_c^i} \wedge y^i = y^j \wedge d^i = d^j\}$, *i.e.*, set of all class-specific embeddings generated from samples that have both the same class label and domain label. Similarly, we can construct the negative set for the i-th sample as $\mathcal{N}^i = \mathcal{B}_d \cup \{p^j | p^j \in \mathcal{B}_c \wedge y^i \neq y^j\}$, *i.e.*, set of all class-invariant embeddings, and all class-specific embeddings from samples with different class label. Then, the contrastive loss for a given batch is:

$$\mathcal{L}_{contr}(\mathcal{B}) = -\frac{1}{|\mathcal{B}|}\sum_{i=1}^{|\mathcal{B}|}\frac{1}{|\mathcal{P}^i|}\sum_{\vec{p}\in\mathcal{P}^i}\log\frac{\exp(\vec{z}_c^i\cdot\vec{p})}{\sum_{n\in\mathcal{N}^i}\exp(\vec{z}_c^i\cdot\vec{n})} \tag{9}$$

This loss function pushes all negative set embeddings, both those representing classes and those representing domains, away from the anchor class embedding, and pulls closer embeddings of the examples in the same domain with the same class label. We show the effectiveness of including both types of embeddings in negative sets through detailed experimentation in Tab 9. The total loss function is the weighted sum of the classification, disentanglement, and contrastive loss:

$$\begin{aligned}\mathcal{L}(\mathcal{B}) = \frac{1}{|\mathcal{B}|}\sum_{(\vec{x},y,d)\in\mathcal{B}}\big(&\mathcal{L}_{cls}^c(\vec{x},y) + \lambda_d\mathcal{L}_{cls}^d(\vec{x},d)\\ &+\lambda_{orth}\mathcal{L}_{orth}(\vec{x}) + \lambda_{adv}\mathcal{L}_{adv}(\vec{x})\big)\\ &+\lambda_{contr}\mathcal{L}_{contr}(\mathcal{B})\end{aligned} \tag{10}$$

where $\lambda_d, \lambda_{orth}, \lambda_{adv}$ and λ_{contr} are hyperparameters.

3.4 Instance-aware Test Time Adaptation

To further adapt the model to the new domain, we leverage high-confidence samples during test time and use a loss supervised by pseudo label to update the network parameters. Unlike [54], which simply discards low confidence samples, *we propose a novel method*, which can use these hard examples but weigh them by their confidence to reduce error accumulation.

We focus on a more realistic and more difficult TTA setting in which test samples are received one by one (instance TTA). Given an input image $\vec{x}$, we generate a batch by sampling random augmentations and applying them to our input sample. We create two separate batches, the first one by sampling N_w augmentations from a set of weak augmentations $\mathcal{A}_w$ and the second one by sampling N_s augmentations from a set of strong augmentations $\mathcal{A}_s$:

$$\mathcal{B}_w = \{a_w^i(\vec{x})\}_{i=1}^{N_w} \tag{11}$$

$$\mathcal{B}_s = \{a_s^i(\vec{x})\}_{i=1}^{N_s} \tag{12}$$

where $a_w^i \sim \mathcal{A}_w$ is a randomly sampled weak augmentation and $a_s^i \sim \mathcal{A}_s$ is a randomly sampled strong augmentation. The features are obtained by pushing the elements of both batches through the

Algorithm 1 : IADG: Training

```
**Input:** training set $\mathcal{D}_{train}$, batch size $B$, $\lambda_d, \lambda_{orth}, \lambda_{adv}, \lambda_{contr}$, Adam optimizer with learning rate $\mu$.
**Initial:** Parameters $\theta_F$ for feature extractor $F$, $\theta_{E_c}$ for encoder $E_c$, $\theta_{E_d}$ for encoder $E_d$, $\theta_{G_c}$ for label classifier $G_c$, $\theta_{G_d}$ for domain classifier $G_d$, $\theta_{P_c}$ for projector $P_c$, $\theta_{P_d}$ for projector $P_d$.
**repeat**
    $\{\vec{x}^i, y^i, d^i\}_{i=0}^B \leftarrow$ sample from $\mathcal{D}_{train}$
    **for** $i = 1, \cdots, B$ **do**
        $\vec{f}^i = F(\vec{x}^i)$
        $\vec{z}_c^i = E_c(\vec{f}^i)$
        $\vec{z}_d^i = E_d(\vec{f}^i)$
        $L_{cls,c}^i = H(\sigma(G_c(\vec{z}_c^i)), y^i)$
        $L_{cls,d}^i = H(\sigma(G_d(\vec{z}_d^i)), d^i)$
        $L_{orth}^i = s(\vec{z}_c^i, \vec{z}_d^i)$
        $L_{adv}^i = -I(G_c(\vec{z}_d^i))$
        $L_{contr}^i = L_{contr}(\vec{z}_i, \{\vec{z}_c^j\}_j, \{\vec{z}_d^j\}_j)$
    **end for**
    $L = \frac{1}{B}\sum_{i=1}^B (L_{cls,c}^i + \lambda_d L_{cls,d}^i + \lambda_{orth} L_{orth}^i + \lambda_{adv} Ł_{adv}^i + \lambda_{contr} Ł_{contr}^i)$
    $\theta_F, \theta_{E_c}, \theta_{E_d}, \theta_{G_c}, \theta_{G_d}, \theta_{P_c}, \theta_{P_d} \leftarrow$ Adam(Ł)
**until** $\theta_F, \theta_{E_c}, \theta_{E_d}, \theta_{G_c}, \theta_{G_d}, \theta_{P_c}, \theta_{P_d}$ converge
```

feature extractor F and encoder E_c.

$$\mathcal{Z}_w = \{E_c(F(\vec{b})) | \vec{b} \in \mathcal{B}_w\} \tag{13}$$

$$\mathcal{Z}_s = \{E_c(F(\vec{b})) | \vec{b} \in \mathcal{B}_s\} \tag{14}$$

$$\vec{z} = E_c(F(\vec{x})) \tag{15}$$

Then, for each feature vector in $\mathcal{Z}_w$, we obtain the soft pseudo-label by pushing the feature vectors through the class label classifier:

$$\mathcal{Y}_w = \{\sigma(G_c(\vec{z})) | \vec{z} \in \mathcal{Z}_w\}$$

where σ is the softmax function. To obtain the hard pseudo label $\hat{y}$, we average all the soft pseudo labels and take the argument of the maxima.

$$\vec{y} = \frac{1}{|\mathcal{Y}_w|} \sum_{\vec{y}_w \in \mathcal{Y}_w} \vec{y}_w \tag{16}$$

$$\hat{y} =_i y^{(i)} \tag{17}$$

Confidence-Weighted Loss. Given the hard pseudo label, we can calculate the standard cross-entropy classification loss:

$$\mathcal{L}_{tta}^{cls} = \frac{1}{|\mathcal{Y}_w|} \sum_{\vec{y}_w \in \mathcal{Y}_w} H(\vec{y}_w, \hat{y}) \tag{18}$$

However, due to the noisy nature of pseudo-labels obtained this can quickly lead to error accumulation. To mitigate this issue, we propose a new *confidence-weighted* classification loss. The fundamental concept driving this loss is that there exists an inverse relationship between loss and confidence. Samples with lower confidence have a higher loss and thus contribute more to the direction of the gradient. This is not a desirable property in test time settings where obtained labels are noisy.

$$\mathcal{L}_{tta}^{conf} = \frac{o(\vec{y})}{|\mathcal{Y}_w|} \sum_{\vec{y}_w \in \mathcal{Y}_w} H(\vec{y}_w, \hat{y}) \tag{19}$$

where o is the confidence function. In our implementation, we use the normalized logit entropy as the confidence function.

$$o(\vec{x}) = -\frac{1}{\log |\vec{x}|} \sum_{i=1}^{|\vec{x}|} x^{(i)} \log x^{(i)} \tag{20}$$

Contrastive-Proxy TTA. Given a set of features $\mathcal{Z}_s$ generated from strong augmentations of the test time sample $\vec{x}$ we can further improve the parameters of the model by exploiting sample-sample and sample-prototype relationships. Because we work in instance TTA setting and all examples in the batch are generated by the same input example we do not have access to any negative samples. However, because the classifier G_c is a single linear layer network with no bias, we can view the rows in the matrix representation of G_c as class prototypes which we can use as negative samples. Let $\mathbf{G}_c$ be the matrix representation of the classifier G_c and let μ^i be the i-th row of $\mathbf{G}_c$, representing the class prototype for the i-th class, and let $\mathcal{M} = \{\vec{\mu}^i\}_{i=1}^C$ be the set of all class prototypes. Then we can construct the positive set as the set of all projections of encoded strongly augmented samples:

$$\mathcal{P} = \{P_c(\vec{e}) | \vec{e} \in \mathcal{Z}_s\} \tag{21}$$

We can construct the negative set as the set of all projections of class prototypes except the one that is the same as the pseudo-label:

$$\mathcal{N} = \{P_c(\vec{\mu}) | \vec{\mu} \in \mathcal{M} \backslash \vec{\mu}^{\hat{y}}\} \tag{22}$$

Given the positive set and negative set we can write the contrastive loss as:

$$\mathcal{L}_{tta}^{contr} = -\frac{o(\vec{y})}{|\mathcal{P}|} \sum_{\vec{p} \in \mathcal{P}} \log \frac{\exp(\vec{z}_p \cdot \vec{p})}{\sum_{\vec{n} \in \mathcal{N}} \exp(\vec{z}_p \cdot \vec{n})} \tag{23}$$

where $\vec{z}_p = P_c(\vec{z})$ is the projected encoded feature vector for the unaugmented sample. To prevent mode collapse we add regularization term [36]:

$$\mathcal{L}_{tta}^{reg} = \frac{1}{C} \sum_{i=1}^{C} I(G_c(\mu^i)) - I(\frac{1}{C} \sum_{i=1}^{C} G_c(\mu^i)) \tag{24}$$

3.5 Training Procedure

Our total test time loss is the sum of classification loss, contrastive loss and regularization term.

$$\mathcal{L}_{tta} = \lambda_{conf} \mathcal{L}_{tta}^{conf} + \lambda_{contr} \mathcal{L}_{tta}^{contr} + \lambda_{reg} \mathcal{L}_{tta}^{reg} \tag{25}$$

where $\lambda_{conf}, \lambda_{contr}, \lambda_{regul}$ are hyperparameters. During TTA we keep the parameters of the feature extractor F frozen and we update only the encoder E_c, classifier G_c and projection network P_c.

4 Experiments

4.1 Experimental Setup

Dataset. We utilize five multi-domain images classification benchmarks: PACS [24], VLCS [38], Office-Home [40], Terra Incognita [3], and DomainNet [30].

Algorithm 2 : IADG: Test-Time Adaptation

```
**Input:** input example $\vec{x}$, set of weak augmentations $\mathcal{A}^w$, set of strong augmentations $\mathcal{A}^s$, hyperparamters $\lambda_{conf}, \lambda_{contr}, \lambda_{regul}, \lambda_{contr}$, $N^c$, $N^w$, Adam optimizer with learning rate $\mu$.
**Initial:** Parameters $\theta_{E_c}$ for encoder $E_c$, $\theta_{G_c}$ for label classifier $G_c$, $\theta_{P_c}$ for projector $P_c$.
**START**
get batch of weakly augmentations views $\mathcal{B}^w = \{\vec{x}_w^i\}$
get batch of weakly augmentations views $\mathcal{B}^w = \{\vec{x}_s^i\}$
**for** $i = 1, \cdots, C$ **do**
    $\vec{n}^i = E_c(F(\vec{\mu}^i))$
**end for**
**for** $i = 1, \cdots, N^w$ **do**
    $\vec{z}_w^i = E_c(F(\vec{f}_w^i))$
    $\vec{l}_w^i = G_c(\vec{z}_w^i)$
**end for**
**for** $i = 1, \cdots, N^s$ **do**
    $\vec{z}_s^i = E_c(F(\vec{x}_s^i))$
**end for**
$\vec{z} = E_c(F(\vec{x}))$
$y = \frac{1}{N^w} \sum \vec{l}_w^i$
$\hat{y} = \arg\max_i y^{(i)}$
**for** $i = 1, \cdots, N^w$ **do**
    $Ł_{cls}^i = o(\vec{l}^i) H(\sigma(G_c(\vec{l}_w^i)), y^i)$
**end for**
**for** $i = 1, \cdots, N^s$ **do**
    $L_{contr}^i = -\log \frac{\exp(\vec{z} \cdot \vec{z}_s^i)}{\sum_{j \neq i} \exp(\vec{z} \cdot \vec{n}^j)}$
**end for**
$L_{regul} = \frac{1}{C} \sum_{i=1}^{C} I(G_c(\mu^i)) - I(\frac{1}{C} \sum_{i=1}^{C} G_c(\mu^i))$
$L = \frac{1}{N^w} \sum_{i=1}^{N^w} (\lambda_{cls} Ł_{cls}^i) + \frac{1}{N^s} \sum_{i=1}^{N^s} (\lambda_{contr} Ł_{contr}^i) + \lambda_{regul} L_{regul}$

$\theta_{E_c}, \theta_{G_c}, \theta_{P_c} \leftarrow \text{Adam}(L)$
**END**
**Return:** $\hat{y}$
```

PACS [24] includes 4 domains such as $d \in$ {art, cartoons, photos, sketch} with 9,991 samples of dimension (3, 224, 224) and 7 classes; **VLCS** [38] includes domains 4 domains such as $d \in$ {Caltech101, LabelMe, SUN09, VOC2007} with 10,729 samples of dimension (3, 224, 224) and 5 classes such as bird, car, chair, dog, person; **Office-Home** [40] includes 4 domains such as $d \in$ {art, clipart, product, real-world} with 15,588 samples of dimension (3, 224, 224) and 65 classes; **Terra Incognita** [3] contains photographs of wild animals taken by camera traps at 4 locations (*i.e.* domains) such as $d \in$ {L100, L38, L43, L46} with 24,788 samples of dimension (3, 224, 224) and 10 classes; **DomainNet** [30] includes 6 domains such as $d \in$ {clipart, infograph, painting, quickdraw, real, sketch} with 586,575 samples of dimension (3, 224, 224) and 345 classes. The results for each dataset are the average results of the leave-one-domain-out testing setting [15] with the training domain validation set. In this setting, one domain is left out for testing, and other domains are used for training. After training, the model is evaluated on the unseen data from the testing domain. This method is repeated for all domains, and the results are averaged. This setting is more difficult than the Oracle setting, in which the validation set comes from the testing domain.

Implementation Details. We adopt ResNet-50 [17] pre-trained on ImageNet [11] as a general feature extractor. Following [15], we freeze all batch normalization layers before training and insert a dropout layer on top of ResNet. For both the encoder and projection networks we three-layer MLP with ReLU activation and batch normalization layers. Classifier is a single linear layer network with no bias. For optimization, we use the Adam [21] optimizer. For fair comparison, the batch size is set to 32 per domain for all experiments, like in other reference DG methods. The learning rate, weight decay, loss hyperparameters, and number of iterations are set separately for each experiment.

We follow the training procedure as in [15]. We use an 8:2 split for both the training and validation sets. After training, the model with the best performance on the validation set is used for testing unseen data from the testing domain. Each experiment is repeated 10 times with different random splits of data. The final reported score is the average performance of different test runs. In Table 2, we show the learning rate, weight decay, dropout factor, and number of iterations of each dataset during our experiment. In training, we set all λ hyperparameters for balancing the loss terms (Eqs. (11) and (26) of the main paper) to 1.0 for all datasets.

4.2 Quantitative Results

Main Results. As shown in Table 1, we present the main quantitative results on five benchmark datasets. For fair comparison, we select the baseline results of methods that use the ResNet-50 [17] as the backbone. We denote the base version of our algorithm as IADG. Our method can easily be combined with other DG methods such as SWAD [5], MIRO [6] and AugMix [18] for even better results. Algorithm IADG shows the results of combining our algorithm with SWAD-style weight averaging. The results for the model pre-trained by IADG and adapted during test time by our TTA method are denoted in the table under the name IADG + TTA. Combining our method with SWAD [5] performs better than any of the previous state-of-the-art methods on average and on all datasets except TerraIncognita. IADG + TTA achieves the best result in PACS, OfficeHome, DomainNet, and VLCS datasets [38] with an average of 1.1% improvement than D-Net [19] and shows almost no improvement on dataset where classification accuracy is low in TerraIncognita [3]. Only IADG without TTA also outperforms the best method D-Net [19] with an improvement of 0.4% on average. Results also hightlight that our IADG + TTA method is better performing than IADG + TENT by 0.4% improvement. We select TENT [41] for a fair comparison based on Table **??**, as it is one of the best performing method. The results can confirm that the inclusion of class-specific features along with domain-specific features can help with generalization to unseen domains.

Evaluation on Each Dataset. We evaluate our framework IADG on PACS, OfficeHome, DomainNet, TerraInc, and VLCS with ResNet-50 as our backbone. We compare our method with standard Empirical risk minimization (ERM) [39], several domain-specific learning methods (*i.e.* GDRO [14], ARM [49], mDSDI [4]), domain-invariant learning methods (*i.e.* IRM [1], CORAL [37], MMD [25], DANN

Table 1: Performance comparison of our method (IADG) with state-of-the-art DG methods in five different benchmarks. Results are shown in accuracy (%). Best results are in bold.

Methods	PACS	OfficeHome	TerraInc	DomainNet	VLCS	Avg.
			Without TTA			
ERM [39]	85.5 ± 0.1	66.5 ± 0.3	46.1 ± 1.0	41.3 ± 0.3	77.5 ± 0.6	63.4
IRM [1]	83.5 ± 0.2	64.3 ± 0.2	47.6 ± 0.9	28.0 ± 0.1	78.6 ± 0.2	60.4
GDRO [14]	84.4 ± 0.4	66.0 ± 0.2	43.2 ± 2.7	33.4 ± 0.1	76.7 ± 0.2	60.7
CORAL [37]	86.2 ± 0.2	68.7 ± 0.5	47.6 ± 0.8	41.8 ± 0.1	78.8 ± 0.2	64.6
MMD [25]	84.6 ± 0.1	66.3 ± 0.3	42.2 ± 1.3	23.5 ± 0.1	77.5 ± 0.2	58.8
DANN [13]	83.6 ± 0.3	65.9 ± 0.2	46.7 ± 0.3	38.3 ± 0.4	78.6 ± 0.2	62.6
CDANN [25]	82.6 ± 0.2	65.8 ± 0.3	45.8 ± 0.6	38.5 ± 0.4	77.5 ± 0.3	62.0
ARM [49]	85.1 ± 0.2	64.8 ± 0.4	45.5 ± 0.6	36.0 ± 0.4	77.6 ± 0.2	61.8
VREx [22]	84.9 ± 0.2	66.4 ± 0.5	46.4 ± 1.1	33.6 ± 0.7	78.3 ± 0.2	61.9
mDSDI [4]	86.2 ± 0.3	69.2 ± 0.4	48.1 ± 0.7	42.8 ± 0.5	79.0 ± 0.3	65.1
ITTA [9]	83.8 ± 0.3	62.0 ±0.2	43.2 ± 0.5	34.9 ± 0.1	76.9 ± 0.6	60.2
MIRO [6]	85.4 ± 0.4	70.5 ± 0.6	50.4 ± 0.9	46.0 ± 0.3	79.0 ± 0.1	66.5
SWAD [5]	88.1 ± 0.2	70.6 ± 0.3	50.0 ± 0.6	46.5 ± 0.6	79.1 ± 0.2	66.9
PCL [46]	88.7 ± 0.3	71.6 ± 0.3	52.1 ± 0.5	47.7 ± 0.4	76.3 ± 0.4	67.2
EoA [2]	88.6 ± 0.3	72.5 ± 0.4	52.3 ± 0.9	47.4 ± 0.7	79.1 ± 0.2	68.0
D-Net [19]	89.2 ± 0.6	70.4 ± 0.4	**54.5 ± 0.8**	48.1 ± 0.2	81.6 ± 0.5	68.8
IADG (Ours)	**89.4 ± 0.3**	**72.8 ± 0.2**	53.9 ± 0.4	**48.7 ± 0.2**	**82.4 ± 0.3**	**69.2**
			With TTA			
IADG + TENT	89.5 ± 0.2	72.9 ± 0.1	53.9 ± 0.5	49.1 ± 0.1	82.5 ± 0.3	69.5
IADG + TTA (Ours)	**89.7 ± 0.2**	**73.1 ± 0.1**	54.2 ± 0.5	**49.5 ± 0.1**	**83.5 ± 0.3**	**69.9**

Table 2: Implementation details of each dataset in IADG.

Dataset	**Learning Rate**	**Weight Decay**	**Dropout**	**Iterations**
PACS [24]	3e-5	1e-5	0.1	10000
DomainNet [30]	3e-6	3e-6	0.5	30000
OfficeHome [40]	1e-5	1e-6	0.3	10000
VLCS [38]	1e-5	1e-5	0.1	10000
TerraInc [3]	5e-6	3e-7	0.3	20000

[13], CDANN [25]), and other strong methods like SWAD [5], PCL [46], EoA [2], and D-Net [19], which are closely related method with us. As shown in Table 3, our IADG without outperforms D-Net by 0.2% and 0.5% with TTA on avg. in PACS. Here, D-Net is built on SWAD. That proved to be a fair comparison with our IADG method. In PACS, domains like cartoon and sketch, have a white background. We achieve better results than all existing methods including domain-invariant methods (*i.e.* IRM, MMD, CORAL, DANN, CDANN) in these domains because our method is jointly aggregated mixed and domain-invariant features. As shown in Table 4, our IADG outperforms EoA (which is the best method in officehome dataset) by 0.3% and 0.6% with TTA on avg. in Office-Home. As we can see, we achieve higher results in domains like art, product, and the real world than other methods, which contain more background and color information. That means our method achieves better generalization in background contexts as well due to disentangling with domain-specific and class-invariant features. As shown in Table 5, our method downgrades 0.5% in comparison with D-Net in both with and without TTA settings. As shown in Table 6, IADG + TTA dominates five domains as well as achieves higher on avg. result than D-Net, even only IADG also achieves better results. It implies that, if we increase the number of domains, it can improve generalization, while balancing both mixed and domain-invariant features. Furthermore, as shown in Table 7, our

Table 3: Performance comparison on PACS [24]. Results are shown in accuracy (%). Best results are in bold.

Methods	Target domain				Avg.
	Art	Cartoon	Photo	Sketch	
ERM	84.7 ± 0.4	80.8 ± 0.6	97.2 ± 0.3	79.3 ± 1.0	85.5
IRM	84.8 ± 1.3	76.4 ± 1.1	96.7 ± 0.6	76.1 ± 1.0	83.5
GroupDRO	83.5 ± 0.9	79.1 ± 0.6	96.7 ± 0.3	78.3 ± 2.0	84.4
Mixup	86.1 ± 0.5	78.9 ± 0.8	97.6 ± 0.1	75.8 ± 1.8	84.6
MLDG	85.5 ± 1.4	80.1 ± 1.7	97.4 ± 0.3	76.6 ± 1.1	84.9
CORAL	88.3 ± 0.2	80.0 ± 0.5	97.5 ± 0.3	78.8 ± 1.3	86.2
MMD	86.1 ± 1.4	79.4 ± 0.9	96.6 ± 0.2	76.5 ± 0.5	84.6
DANN	86.4 ± 0.8	77.4 ± 0.8	97.3 ± 0.4	73.5 ± 2.3	83.6
CDANN	84.6 ± 1.8	75.5 ± 0.9	96.8 ± 0.3	73.5 ± 0.6	82.6
SWAD	89.3 ± 0.8	83.4 ± 0.5	97.3 ± 0.8	85.5 ± 1.8	88.1
PCL	**90.2 ± 1.0**	83.9 ± 0.6	98.1 ± 0.1	82.6 ± 0.4	88.7
ARM	86.8 ± 0.6	76.8 ± 0.5	97.4 ± 0.3	79.3 ± 1.2	85.1
VREx	86.0 ± 1.6	79.1 ± 0.6	96.9 ± 0.5	77.7 ± 1.7	84.9
D-Net	87.8 ± 0.8	86.5 ± 1.8	96.8 ± 0.3	85.8 ± 1.2	89.2
IADG	89.5 ± 0.8	87.5 ± 0.6	98.2 ± 0.3	86.2 ± 0.9	89.4
IADG + TTA	89.6 ± 0.8	**88.2 ± 0.5**	**98.4 ± 0.2**	**87.2 ± 1.1**	**89.7**

IADG + TTA method outperforms other methods by a considerable margin of 0.9%. It shows that the inclusion of TTA can also achieve results, where this dataset is highly regarded by missing objects in the background, for example, a bird image only having sky. In that case, other methods do not capture domain-specific information, which leads to a poor generalization ability.

Comparison with TTA Methods. As presented in Table **??**, the evaluation spans multiple well-known domain generalization datasets, including PACS, OfficeHome, TerraInc, and VLCS. Results highlight that our approach consistently achieves higher performance metrics compared to other TTA methods such as AdaContrast [8], EATA [28], and TENT [41] . This superiority is reflected in both individual

Table 4: Performance comparison on OfficeHome [40]. Results are shown in accuracy (%). Best results are in bold.

Methods	Target domain				Avg.
	Art	Clipart	Product	Real World	
ERM	61.3 ± 0.7	52.4 ± 0.3	75.8 ± 0.1	76.6 ± 0.3	66.5
IRM	58.9 ± 2.3	52.2 ± 1.6	72.1 ± 2.9	74.0 ± 2.5	64.3
GroupDRO	60.4 ± 0.7	52.7 ± 1.0	75.0 ± 0.7	76.0 ± 0.7	66.0
Mixup	62.4 ± 0.8	54.8 ± 0.6	76.9 ± 0.3	78.3 ± 0.2	68.1
MLDG	61.5 ± 0.9	53.2 ± 0.6	75.0 ± 1.2	77.5 ± 0.4	66.8
CORAL	65.3 ± 0.4	54.4 ± 0.5	76.5 ± 0.1	78.4 ± 0.5	68.7
MMD	60.4 ± 0.2	53.3 ± 0.3	74.3 ± 0.1	77.4 ± 0.6	66.3
DANN	59.9 ± 1.3	53.0 ± 0.3	73.6 ± 0.7	76.9 ± 0.5	65.9
CDANN	61.5 ± 1.4	50.4 ± 2.4	74.4 ± 0.9	76.6 ± 0.8	65.8
SWAD	66.1 ± 0.7	57.7 ± 0.6	78.4 ± 0.4	80.2 ± 0.4	70.6
PCL	67.3 ± 0.2	59.9 ± 0.4	78.7 ± 0.4	80.7 ± 0.3	71.6
ARM	58.9 ± 0.8	51.0 ± 0.5	74.1 ± 0.1	75.2 ± 0.3	64.8
VREx	60.7 ± 0.9	53.0 ± 0.9	75.3 ± 0.1	76.6 ± 0.5	66.4
D-Net	65.7 ± 1.4	58.6 ± 0.3	78.0 ± 1.1	79.7 ± 1.3	70.5
IADG	70.8 ± 0.5	58.9 ± 0.6	79.8 ± 0.1	81.6 ± 0.4	72.8
IADG + TTA	**70.9 ± 0.4**	58.9 ± 0.6	**80.6 ± 0.1**	**82.2 ± 0.5**	**73.1**

Table 5: Performance comparison on TerraInc [3]. Results are shown in accuracy (%). Best results are in bold.

Methods	Target domain				Avg.
	L38	L43	L46	L100	
ERM	49.8 ± 4.4	42.1 ± 1.4	56.9 ± 1.8	35.7 ± 3.9	46.1
IRM	54.6 ± 1.3	39.8 ± 1.9	56.2 ± 1.8	39.6 ± 0.8	47.6
GroupDRO	41.2 ± 0.7	38.6 ± 2.1	56.7 ± 0.9	36.4 ± 2.1	43.2
Mixup	59.6 ± 2.0	42.2 ± 1.4	55.9 ± 0.8	33.9 ± 1.4	47.9
MLDG	54.2 ± 3.0	44.3 ± 1.1	55.6 ± 0.3	36.9 ± 2.2	47.7
CORAL	51.6 ± 2.4	42.2 ± 1.0	57.0 ± 1.0	39.8 ± 2.9	47.6
MMD	41.9 ± 3.0	34.8 ± 1.0	57.0 ± 1.9	35.2 ± 1.8	42.2
DANN	51.1 ± 3.5	40.6 ± 0.6	57.4 ± 0.5	37.7 ± 1.8	46.7
CDANN	47.0 ± 1.9	41.3 ± 4.8	54.9 ± 1.7	39.8 ± 2.3	45.8
SWAD	55.4 ± 1.2	44.9 ± 6.3	59.7 ± 1.1	39.9 ± 0.8	50.0
PCL	58.7 ± 2.9	46.3 ± 2.5	60.0 ± 0.6	43.6 ± 1.3	52.1
ARM	49.3 ± 0.7	38.3 ± 2.4	55.8 ± 0.8	38.7 ± 1.3	45.5
VREx	48.2 ± 4.3	41.7 ± 1.3	56.8 ± 0.8	38.7 ± 3.1	46.4
D-Net	**63.1 ± 2.2**	**48.7 ± 1.4**	60.0 ± 1.4	**46.4 ± 0.6**	**54.5**
IADG	61.2 ± 2.1	48.3 ± 0.7	**61.5 ± 0.9**	45.0 ± 0.9	53.9
IADG + TTA	60.7 ± 1.9	47.1 ± 0.6	60.1 ± 1.1	44.6 ± 1.0	54.2

dataset accuracies and the overall average accuracy. Due to time limitation, we could not perform experiment on DomainNet.

4.3 Ablation Studies

This subsection provides ablation studies and analysis of each component in IADG and test-time adaptation method.

Ablation on Each Component. We investigate how performance changes by adding each component to ERM in Table **??**. Applying adversarial loss to both types of embeddings reduces the performance since it forces the encoder to not encode any domain-specific features into z^c, including the ones that can help with generalization. Adding a small encoder network on top of the feature extractor has almost no effect on performance, despite the network having more parameters. Having adversarial loss applied to only z^c seems to be detrimental. This might be because this forces all mixed features into z^d, which is the opposite of what IADG is trying to achieve.

Ablation of Contrastive Loss. We conduct experiments to investigate the selection criteria for constructing positive and negative sets that are used in contrastive loss. For a given batch of class-specific embeddings $\{z_c^j\}_j$, class-invariant embeddings $\{z_d^j\}_j$, and anchor index i, we can construct a set of positive samples and set of negative examples based on different criteria. We focus on label-based selection criteria, *i.e.*, including examples by comparing different class and domain labels of anchor with class and domain labels of other samples. The best results are achieved when only the examples from the same domain are included in the positive set, as shown in Table 8. Including only results from different domains hurts the performance. This might be due to the fact that even with disentanglement, features from different domains are too difficult to pull close together since we do not eliminate all domain-specific features. Description of different ways to construct the positive set in CL are provided in the supplementary material.

Impact of and Positive and Negative Sets. The diversity of the negative set seems to be positively correlated with the performance. The performance was the worst in versions that only included samples from one domain, as shown in Table 9. Including z_c in the negative set has more effect than including z_d in the negative set but including both types results in better performance than just including one. Description of different ways to construct the negative set in CL are provided in More details on implementation are in supplementary.

Ablation on TTA Module. We investigate the contribution of each component of our TTA method as shown in Table **??**. First, we show that using confidence-weighted cross-entropy loss can improve model performance by tuning model parameters to the test domain. Next, we show that contrastive learning can also be applied during test time. Positive pairs are generated from different augmentations while class prototypes are used as negative pairs. We also show that contrastive learning is prone to mode collapse and reduced performance if it is used without any regularization. Finally, we show that these two approaches can be combined to further improve the model generalization ability.

Ablation on TTA Complexity. As shown in Table 16, our method demonstrates lower parameter usage and runtime while achieving better performance compared to TENT [41] . Specifically, the table details the computational complexity of both TENT and our TTA method in terms of FLOPS, parameters, and execution time. Our approach requires fewer parameters and less computation, leading to more efficient runtime performance. However, note that the adaptation process itself remains computationally expensive, as it involves additional forward and backward steps during the test phase, which increases the overall complexity.

5 Visualizations

To understand our method, we show the visualizations with Grad-CAM [34] for our training objective and TTA objective in Figure 3. Image regions that contribute the most to the objective are similar for both the training loss and TTA loss. According to [27], this fact ensures the increase in the model's ability to classify images. Furthermore, in Figure 4, our training objective and baseline ERM. By visually inspecting the generated heat maps, we can see that our

Table 6: Performance comparison on DomainNet [30]. Results are shown in accuracy (%). Best results are in bold.

Methods	Target domain						Avg.
	Clip.	Info.	Paint.	Quick.	Real	Sketch	
ERM	58.1 ± 0.3	18.8 ± 0.3	46.7 ± 0.3	12.2 ± 0.4	59.6 ± 0.1	49.8 ± 0.4	40.9
IRM	48.5 ± 2.8	15.0 ± 1.5	38.3 ± 4.3	10.9 ± 0.5	48.2 ± 5.2	42.3 ± 3.1	33.9
GroupDRO	47.2 ± 0.5	17.5 ± 0.4	33.8 ± 0.5	9.3 ± 0.3	51.6 ± 0.4	40.1 ± 0.6	33.3
Mixup	55.7 ± 0.3	18.5 ± 0.5	44.3 ± 0.5	12.5 ± 0.4	55.8 ± 0.3	48.2 ± 0.5	39.2
MLDG	59.1 ± 0.2	19.1 ± 0.3	45.8 ± 0.7	13.4 ± 0.3	59.6 ± 0.2	50.2 ± 0.4	41.2
CORAL	59.2 ± 0.1	19.7 ± 0.2	46.6 ± 0.3	13.4 ± 0.4	59.8 ± 0.2	50.1 ± 0.6	41.5
MMD	32.1 ± 13.3	11.0 ± 4.6	26.8 ± 11.3	8.7 ± 2.1	32.7 ± 13.8	28.9 ± 11.9	23.4
DANN	53.1 ± 0.2	18.3 ± 0.1	44.2 ± 0.7	11.8 ± 0.1	55.5 ± 0.4	46.8 ± 0.6	38.3
CDANN	54.6 ± 0.4	17.3 ± 0.1	43.7 ± 0.9	12.1 ± 0.7	56.2 ± 0.4	45.9 ± 0.5	38.3
SWAD	66.0 ± 0.5	22.4 ± 0.4	53.5 ± 0.1	16.1 ± 0.1	65.8 ± 0.3	55.5 ± 0.1	46.5
PCL	67.9 ± 0.3	24.3 ± 0.2	55.3 ± 0.3	15.7 ± 0.5	66.6 ± 0.5	56.4 ± 0.2	47.7
ARM	49.7 ± 0.3	16.3 ± 0.5	40.9 ± 1.1	9.4 ± 0.1	53.4 ± 0.4	43.5 ± 0.4	35.5
VREx	47.3 ± 3.5	16.0 ± 1.5	35.8 ± 4.6	10.9 ± 0.3	49.6 ± 4.9	42.0 ± 3.0	33.6
D-Net	66.5 ± 1.2	24.6 ± 0.5	55.8 ± 0.6	16.6 ± 0.2	67.8 ± 0.7	**57.5 ± 0.9**	48.1
IADG	67.6 ± 0.5	24.8 ± 0.1	56.4 ± 0.3	17.1 ± 0.4	68.5 ± 0.1	57.4 ± 1.1	48.7
IADG + TTA	**68.1 ± 0.5**	**25.3 ± 0.1**	**56.8 ± 0.2**	**17.7 ± 0.6**	**69.1 ± 0.1**	55.2 ± 0.4	**49.5**

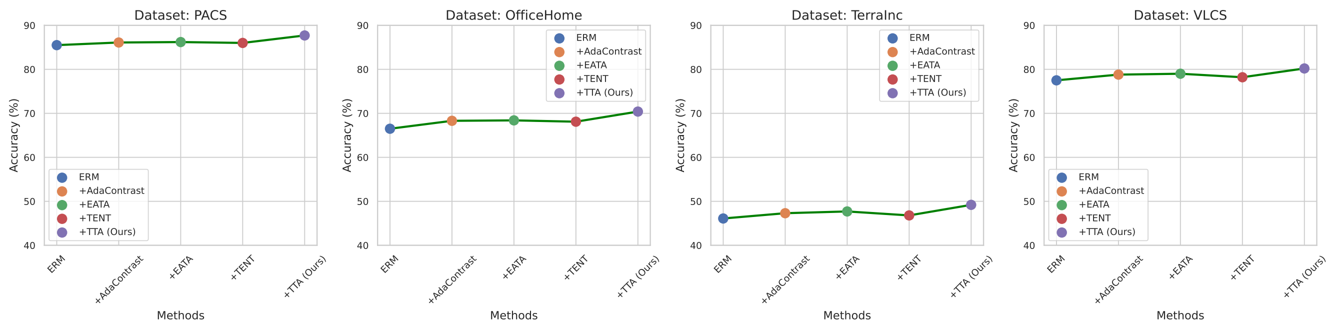

Figure 2: Comparison of TTA methods.

Table 7: Performance comparison on VLCS [38]. Results are shown in accuracy (%). Best results are in bold.

Methods	Target domain				Avg.
	Caltech101	LabelMe	SUN09	VOC2007	
ERM	97.7 ± 0.4	64.3 ± 0.9	73.4 ± 0.5	74.6 ± 1.3	77.5
IRM	98.6 ± 0.1	64.9 ± 0.9	73.4 ± 0.6	77.3 ± 0.9	78.5
GroupDRO	97.3 ± 0.3	63.4 ± 0.9	69.5 ± 0.8	76.7 ± 0.7	76.7
Mixup	98.3 ± 0.6	64.8 ± 1.0	72.1 ± 0.5	74.3 ± 0.8	77.4
MLDG	97.4 ± 0.2	65.2 ± 0.7	71.0 ± 1.4	75.3 ± 1.0	77.2
CORAL	98.3 ± 0.1	66.1 ± 1.2	73.4 ± 0.3	77.5 ± 1.2	78.8
MMD	97.7 ± 0.1	64.0 ± 1.1	72.8 ± 0.2	75.3 ± 3.3	77.5
DANN	99.0 ± 0.3	65.1 ± 1.4	73.1 ± 0.3	77.2 ± 0.6	78.6
CDANN	97.1 ± 0.3	65.1 ± 1.2	70.7 ± 0.8	77.1 ± 1.5	77.5
SWAD	98.8 ± 0.4	63.3 ± 0.3	75.3 ± 0.7	79.2 ± 1.7	79.1
PCL	97.1 ± 0.4	62.1 ± 0.5	71.1 ± 1.3	75.1 ± 0.5	76.3
ARM	98.7 ± 0.2	63.6 ± 0.7	71.3 ± 1.2	76.7 ± 0.6	77.6
VREx	98.4 ± 0.3	64.4 ± 1.4	74.1 ± 0.4	76.2 ± 1.3	78.3
D-Net	99.1 ± 0.1	70.2 ± 0.7	77.2 ± 1.2	80.0 ± 0.8	81.6
IADG	99.2 ± 0.1	70.5 ± 0.6	78.4 ± 0.3	81.4 ± 0.8	82.4
IADG + TTA	**99.4 ± 0.1**	70.7 ± 0.5	**78.6 ± 0.2**	**81.4 ± 0.6**	**83.5**

method is confined to a slightly bigger region, while still only focusing on discriminate features and not the background. Including a larger relevant image region can potentially help with classification performance.

6 Conclusion

We propose IADG, a DG method that explores class and domain specific features to retain more information that are useful for robust generalization. We demonstrate that employing contrastive learning to disentangle representations in a shared latent space enhances robust representations for generalization tasks. During testing, we propose a novel low-risk instance Test-Time Adaptation (TTA) method, leveraging information from high-confidence samples to fine-tune parameters for the testing domain. Our approach mitigates error accumulation by assigning lower weights to less confident samples. Our method outperforms state-of-art methods in multiple datasets to demonstrate advantages over other methods.

Acknowledgment

This work was partly supported by (1) the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT)

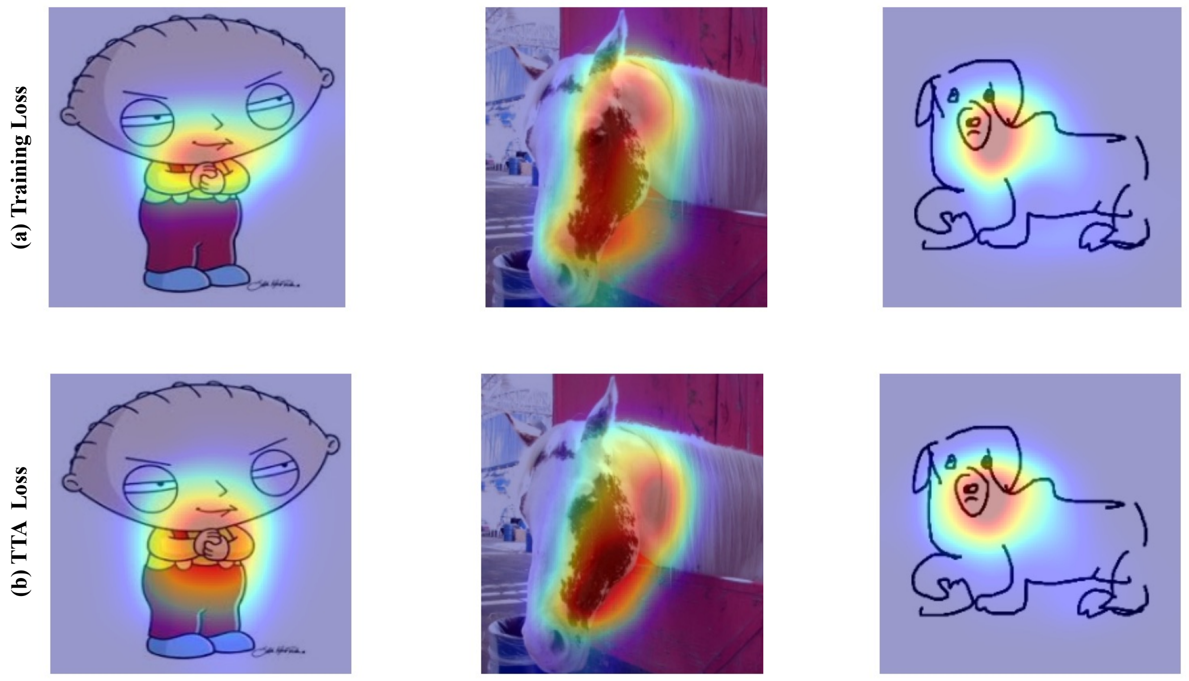

Figure 3: Grad-Cam visualizations. (a) Training Loss, (b) TTA Loss of our method respectively.

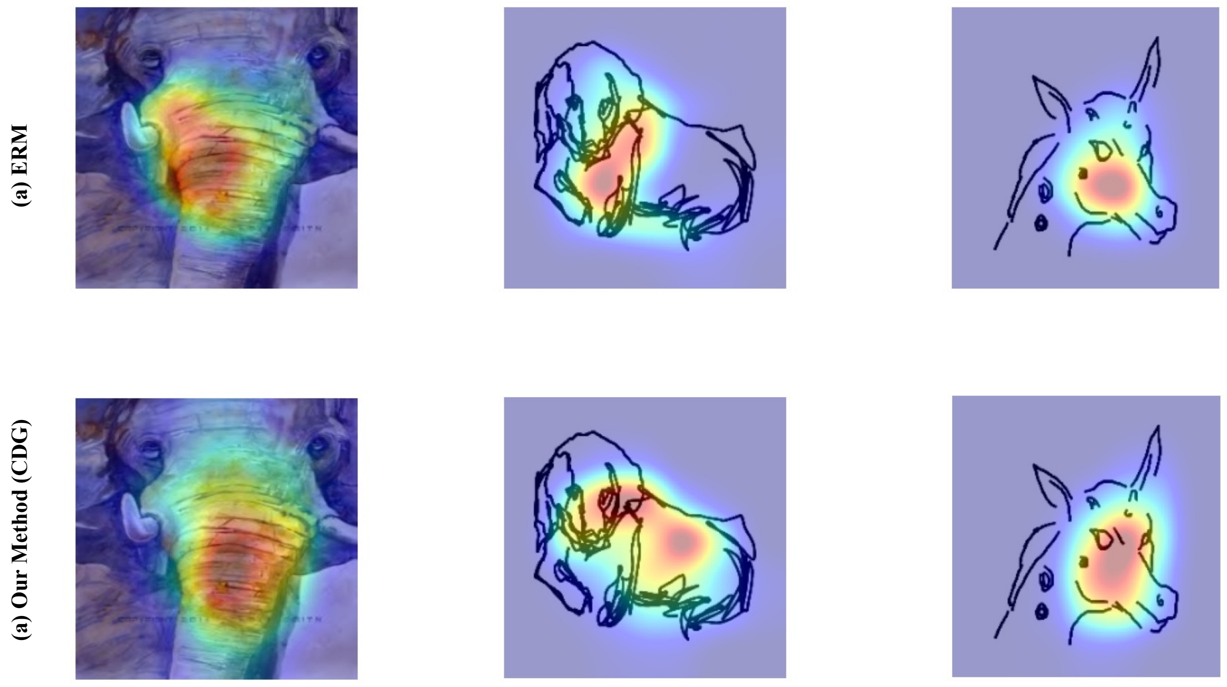

Figure 4: Grad-Cam visualizations. (a) ERM, (b) Our Method (IADG).

Table 8: Description of different ways to construct the positive set in contrastive learning.

Method	Description
$P1$	Positive set includes all examples with the same class label as the anchor. Negative set is empty.
$P2$	Positive set includes only examples with the same class label and same domain label as the anchor. Negative set is empty.
$P3$	Positive set includes only examples with the same class label and different domain label as the anchor. Negative set is empty.

Table 9: Description of different ways to construct the negative set in contrastive learning. Classification accuracy of contrastive learning methods that differ in negative set. Experiment is done on OfficeHome.

Method	Description
$N1$	Positive set is empty. Negative set includes all z_c with a different class label as the anchor.
$N2$	Positive set is empty. Negative set includes only z_c that have same the same domain label but different class label as the anchor.
$N3$	Positive set is empty. Negative set includes all z_d.
$N4$	Positive set is empty. Negative set includes only z_d of examples with the same domain label as the anchor.
$N5$	Positive set is empty. Negative set includes only z_d of examples with the same class label as the anchor.
$N6$	Positive set is empty. Negative set includes only z_d of examples with different domain label as the anchor.
$N7$	Positive set is empty. Negative set includes only z_d of examples with different class label as the anchor.
$N8$	Positive set is empty. Negative set includes all z_d and z_c with a different class label as the anchor.

Table 10: Descriptions of methods used in the ablation study on Office-Home [40].

Method	Description
ERM [39]	Empirical risk minimization
V_{enc}	ERM + encoders
V_{orth}	ERM + encoders + orth
V_{adv-d}	ERM + encoders + orth + domain adversarial
V_{adv-c}	ERM + encoders + orth + class adversarial
V_{adv-cd}	ERM + encoders + orth + domain and class adversarial
IADG	ERM + encoders + orth + domain adversarial + contrastive

Table 11: Performance comparison of different methods on Office-Home [40].

Method	Art	Clipart	Product	Real-World	Average
ERM [39]	61.3	52.4	75.8	76.6	66.5
V_{enc}	61.4	52.5	75.9	76.6	66.4 (⇃ 0.1)
V_{orth}	63.3	53.6	76.4	78.0	67.8 (↾ 1.2)
V_{adv-d}	65.4	54.4	75.7	78.8	68.6 (↾ 2.1)
V_{adv-c}	64.1	53.7	75.3	77.7	67.7 (↾ 1.1)
V_{adv-cd}	63.5	53.8	76.7	78.1	68.0 (↾ 1.5)
IADG	**70.8**	**58.9**	**79.8**	**81.6**	**72.8 (↾ 6.3)**

Table 12: Performance comparison with positive set variations on Office-Home [40]. Results are shown in accuracy (%). Best results are in Bold.

Method	Art	Clipart	Product	Real-World	Avg.
V_{adv-d}	65.4	54.4	75.7	78.8	68.6
$P0$	**66.6**	54.7	76.0	78.9	69.1
$P1$	66.1	**56.4**	**78.7**	**80.3**	**70.4**
$P2$	64.9	54.6	75.8	78.3	68.4

(RS-2024-00345398) and (2) the Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea govern-ment(MSIT)(RS-2020-II201373,Artificial Intelligence Graduate School Program (Hanyang University)).

Table 13: Performance comparison with negative set variations on Office-Home [40]. Results are shown in accuracy (%). Best results are in Bold.

Method	Art	Clipart	Product	Real-World	Avg.
V_{adv-d}	65.4	54.4	75.7	78.8	68.6
$N1$	65.7	52.7	**78.2**	79.0	69.4
$N2$	65.9	54.8	76.1	79.2	69.0
$N3$	65.6	54.4	75.6	78.7	68.6
$N4$	65.6	54.5	75.8	78.8	68.7
$N5$	65.9	54.5	75.3	78.9	68.7
$N6$	66.1	54.6	75.4	79.0	68.8
$N7$	65.4	54.5	76.0	78.8	68.7
$N8$	**66.2**	**56.9**	78.1	**80.2**	**71.7**

Table 14: Ablation study on each component of TTA on VLCS [38]. The method names and related descriptions are provided.

Method	Description
$V1$	Only classification loss.
$V2$	Only contrastive loss without regularization.
$V3$	Only contrastive loss.
$V4$	Full TTA.

Table 15: Performance comparison under different TTA settings on VLCS.

Method	Caltech101	LabelMe	SUN09	VOC2007	Average
IADG	99.3	70.6	78.2	81.1	82.1
$V1$	98.4	**70.9**	77.0	79.6	82.4 (↾ 0.3)
$V2$	89.7	60.3	68.0	71.0	72.3 (⇃ 7.5)
$V3$	98.6	68.1	76.6	80.0	82.4 (↾ 0.3)
$V4$	**99.4**	70.7	**78.6**	**81.4**	**82.5 (↾ 0.4)**

References

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

Table 16: Complexity of TENT vs. TTA (our) settings. Here, incl. and excl. refer to including and excluding TTA method.

Method	FLOPS (G)	Params (M)	Time (s)
ERM	1.82	11.18	0.004
IADG + TENT [41] (incl. / excl.)	6.97 / 1.85	14.33 / 14.28	0.007
IADG + TTA (incl. / excl.)	5.87 / 1.85	13.62 / 13.57	0.005

[2] Devansh Arpit, Huan Wang, Yingbo Zhou, and Caiming Xiong. 2022. Ensemble of averages: Improving model selection and boosting performance in domain generalization. *Advances in Neural Information Processing Systems* 35 (2022), 8265–8277.

[3] Sara Beery, Grant Van Horn, and Pietro Perona. 2018. Recognition in terra incognita. In *Proceedings of the European conference on computer vision (ECCV)*. 456–473.

[4] Manh-Ha Bui, Toan Tran, Anh Tran, and Dinh Phung. 2021. Exploiting domain-specific features to enhance domain generalization. *Advances in Neural Information Processing Systems* 34 (2021), 21189–21201.

[5] Junbum Cha, Sanghyuk Chun, Kyungjae Lee, Han-Cheol Cho, Seunghyun Park, Yunsung Lee, and Sungrae Park. 2021. Swad: Domain generalization by seeking flat minima. *Advances in Neural Information Processing Systems* 34 (2021), 22405–22418.

[6] Junbum Cha, Kyungjae Lee, Sungrae Park, and Sanghyuk Chun. 2022. Domain generalization by mutual-information regularization with pre-trained models. In *European Conference on Computer Vision*. Springer, 440–457.

[7] Prithvijit Chattopadhyay, Yogesh Balaji, and Judy Hoffman. 2020. Learning to balance specificity and invariance for in and out of domain generalization. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX 16*. Springer, 301–318.

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

[9] Liang Chen, Yong Zhang, Yibing Song, Ying Shan, and Lingqiao Liu. 2023. Improved Test-Time Adaptation for Domain Generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 24172–24182.

[10] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR, 1597–1607.

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

[12] Yingjun Du, Jun Xu, Huan Xiong, Qiang Qiu, Xiantong Zhen, Cees GM Snoek, and Ling Shao. 2020. Learning to learn with variational information bottleneck for domain generalization. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16*. Springer, 200–216.

[13] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The journal of machine learning research* 17, 1 (2016), 2096–2030.

[14] Worst-Case Generalization. [n. d.]. Distributionally Robust Neural Networks for Group Shifts: On the Importance of Regularization for. ([n. d.]).

[15] Ishaan Gulrajani and David Lopez-Paz. 2020. In search of lost domain generalization. *arXiv preprint arXiv:2007.01434* (2020).

[16] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 9729–9738.

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

[18] Dan Hendrycks, Norman Mu, Ekin D Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. 2019. Augmix: A simple data processing method to improve robustness and uncertainty. *arXiv preprint arXiv:1912.02781* (2019).

[19] Lanqing Hu, Meina Kan, Shiguang Shan, and Xilin Chen. 2023. DandelionNet: Domain Composition with Instance Adaptive Classification for Domain Generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 19050–19059.

[20] Daehee Kim, Youngjun Yoo, Seunghyun Park, Jinkyu Kim, and Jaekoo Lee. 2021. Selfreg: Self-supervised contrastive regularization for domain generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9619–9628.

[21] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[22] David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Dinghuai Zhang, Remi Le Priol, and Aaron Courville. 2021. Out-of-distribution generalization via risk extrapolation (rex). In *International Conference on Machine Learning*. PMLR, 5815–5826.

[23] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. 2018. Learning to generalize: Meta-learning for domain generalization. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[24] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. 2017. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*. 5542–5550.

[25] Ya Li, Mingming Gong, Xinmei Tian, Tongliang Liu, and Dacheng Tao. 2018. Domain generalization via conditional invariant representations. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

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

[27] Yuejiang Liu, Parth Kothari, Bastien Van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. 2021. Ttt++: When does self-supervised test-time training fail or thrive? *Advances in Neural Information Processing Systems* 34 (2021), 21808–21820.

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

[29] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. 2023. Towards Stable Test-Time Adaptation in Dynamic Wild World. arXiv:2302.12400 [cs.LG]

[30] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. 2019. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision*. 1406–1415.

[31] Fengchun Qiao, Long Zhao, and Xi Peng. 2020. Learning to learn single domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12556–12565.

[32] Alexander Robey, George J Pappas, and Hamed Hassani. 2021. Model-based domain generalization. *Advances in Neural Information Processing Systems* 34 (2021), 20210–20229.

[33] Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. 2020. Contrastive learning with hard negative samples. *arXiv preprint arXiv:2010.04592* (2020).

[34] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*. 618–626.

[35] Junha Song, Jungsoo Lee, In So Kweon, and Sungha Choi. 2023. EcoTTA: Memory-Efficient Continual Test-time Adaptation via Self-distilled Regularization. arXiv:2303.01904 [cs.CV]

[36] Jost Tobias Springenberg. 2015. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390* (2015).

[37] Baochen Sun and Kate Saenko. 2016. Deep coral: Correlation alignment for deep domain adaptation. In *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14*. Springer, 443–450.

[38] Antonio Torralba and Alexei A Efros. 2011. Unbiased look at dataset bias. In *CVPR 2011*. IEEE, 1521–1528.

[39] Vladimir N Vapnik. 1999. An overview of statistical learning theory. *IEEE transactions on neural networks* 10, 5 (1999), 988–999.

[40] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. 2017. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5018–5027.

[41] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. 2020. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726* (2020).

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

[43] Shujun Wang, Lequan Yu, Caizi Li, Chi-Wing Fu, and Pheng-Ann Heng. 2020. Learning from extrinsic and intrinsic supervisions for domain generalization. In *European Conference on Computer Vision*. Springer, 159–176.

[44] Aming Wu, Rui Liu, Yahong Han, Linchao Zhu, and Yi Yang. 2021. Vector-decomposed disentanglement for domain-invariant object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9342–9351.

[45] Huaxiu Yao, Xinyu Yang, Xinyi Pan, Shengchao Liu, Pang Wei Koh, and Chelsea Finn. 2023. Improving Out-of-Domain Generalization with Domain Relations. In *The Twelfth International Conference on Learning Representations*.

[46] Xufeng Yao, Yang Bai, Xinyun Zhang, Yuechen Zhang, Qi Sun, Ran Chen, Ruiyu Li, and Bei Yu. 2022. Pcl: Proxy-based contrastive learning for domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7097–7107.

[47] Jian Zhang, Lei Qi, Yinghuan Shi, and Yang Gao. 2023. Domainadaptor: A novel approach to test-time adaptation. In *Proceedings of the IEEE/CVF International*

Conference on Computer Vision. 18971–18981.

[48] Ling Zhang, Xiaosong Wang, Dong Yang, Thomas Sanford, Stephanie Harmon, Baris Turkbey, Holger Roth, Andriy Myronenko, Daguang Xu, and Ziyue Xu. 2019. When unseen domain generalization is unnecessary? rethinking data augmentation. *arXiv preprint arXiv:1906.03347* (2019).

[49] Marvin Zhang, Henrik Marklund, Nikita Dhawan, Abhishek Gupta, Sergey Levine, and Chelsea Finn. 2021. Adaptive risk minimization: Learning to adapt to domain shift. *Advances in Neural Information Processing Systems* 34 (2021), 23664–23678.

[50] Han Zhao, Remi Tachet Des Combes, Kun Zhang, and Geoffrey Gordon. 2019. On learning invariant representations for domain adaptation. In *International conference on machine learning*. PMLR, 7523–7532.

[51] Shanshan Zhao, Mingming Gong, Tongliang Liu, Huan Fu, and Dacheng Tao. 2020. Domain generalization via entropy regularization. *Advances in Neural Information Processing Systems* 33 (2020), 16096–16107.

[52] Kaiyang Zhou, Ziwei Liu, Yu Qiao, Tao Xiang, and Chen Change Loy. 2022. Domain generalization: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

[53] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. 2021. Domain generalization with mixstyle. *arXiv preprint arXiv:2104.02008* (2021).

[54] Yang Zou, Zhiding Yu, Xiaofeng Liu, B. V. K. Vijaya Kumar, and Jinsong Wang. 2019. Confidence Regularized Self-Training. *CoRR* abs/1908.09822 (2019). arXiv:1908.09822 http://arxiv.org/abs/1908.09822