Instance-Aware Test-Time Adaptation for Domain Generalization

Taki Hasan Rafi[1], Serbeter Karlo[2] Amit Agarwal[3], Hitesh L. Patel[3], Bhargava Kumar[4], and Dong-Kyu Chae[1(✉)]

[1] Hanyang University, South Korea
dongkyu@hanyang.ac.kr
[2] University of Cambridge, United Kingdom
[3] Oracle Cloud Infrastructure, USA
[4] TD Securities, USA

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 is mostly needed 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 do not explicitly utilize class-specific information. Therefore, these approaches ignore mixed features, which 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 along 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.

Keywords: Domain Generalization · Test-Time Adaptation

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 [1], underscoring the challenge posed by the

discrepancy between training and testing domains, that also referred to as domain shift. Domain generalization (DG) aims to reduce domain shifts by leveraging multiple source domains, whereas test-time adaptation (TTA) improves generalization in unseen target data during inference [27,20].

Even though recent DG 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. 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 for domains and their belonging classes.

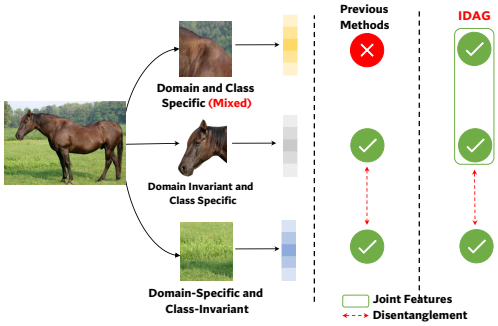

Fig. 1: Most previous disentanglement methods focus on disentangling features into class-invariant and domain-invariant representations, while ignoring features that are both class and domain-specific (mixed). Our disentanglement approach aims to include both mixed and domain-invariant features.

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 [21,24,5] require a memory bank of test time samples, which can potentially increase complexity during inference.

To address the above limitations, we propose **IADG**, a test-time adaptation approach for DG that leverages both class- and domain-specific features. IADG integrates *disentangled representation learning (DRL)* and *contrastive learning (CL)* to extract class and domain-invariant representations. Unlike prior works that strictly separate features, we allow mixed features to improve generalization. Using a single feature extractor, IADG employs two small encoders, orthogonality constraints, and adversarial regularization to disentangle representations. CL further enhances generalization by repelling class-irrelevant features while pulling together class-relevant ones, significantly increasing negative pairs and improving learned representations.

For TTA, IADG updates model parameters per-instance to handle domain shifts. Given a single unlabeled test sample, we estimate a pseudo label by averaging weakly augmented predictions. Unlike prior TTA methods simply keep only the samples with confidence above some fixed threshold [15]. 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.* Inspired by IADG's success in combining classification and contrastive loss, we incorporate contrastive loss at test time. Since no training or other test samples are available, *class centroids* construct negative pairs.

Our contributions can be summarized as follows. Firstly, 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. Secondly, we introduce a novel confidence-guided 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.

2 Related Works

Domain Generalization. Traditional DG methods emphasize learning domain-invariant representations [11]. Contrastive learning-based approaches, like PCL [23] align representations across classes and domains. While existing methods largely focus on domain-specific features, our framework combines contrastive learning, disentanglement, and 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. In DG, PCL [23] 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.

Test-Time Adaptation. TTA involves updating a pre-trained model at test time to adapt to new data. TENT [20] adjusts 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

In DG, we consider $\mathcal{X} \subset \mathbb{R}^d$ be the input space and let $\mathcal{Y} \subset \mathbb{R}$ be the output space. A domain dataset $D = \{\boldsymbol{x}^k \in \mathcal{X}, y^k \in \mathcal{Y}\}_{k=1}^N$ consists of N samples independently drawn from a joint distribution P_{XY} on $\mathcal{X}$ and $\mathcal{Y}$. 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 $P_{XY,s}^i$ and the joint distribution for the i-th target dataset as $P_{XY,t}^i$. It is important

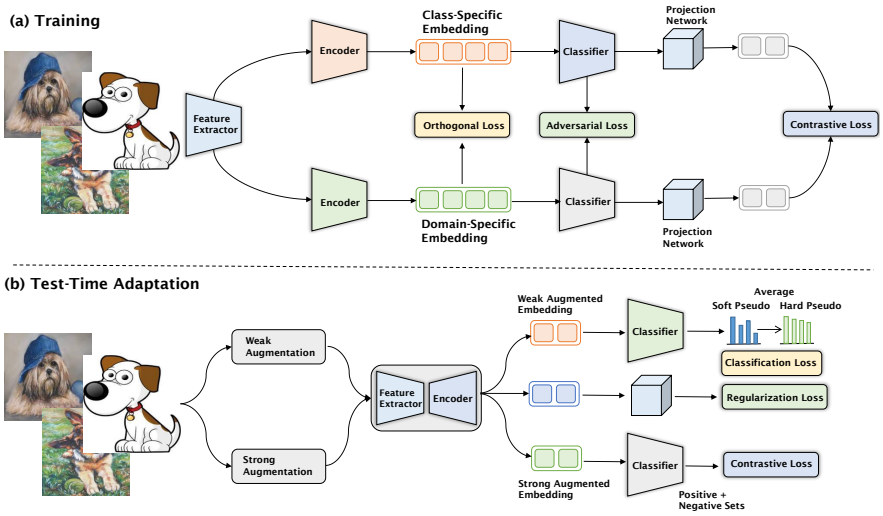

Fig. 2: The overall framework of the proposed IADG model.

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.

3.1 Bridging Class Domain-Specific Features

Given an input $\boldsymbol{x}$, we extract features $\boldsymbol{z} = F(\boldsymbol{x})$ using $F : \mathcal{X} \rightarrow \mathbb{R}^Z$, where $\boldsymbol{z}$ contains both class-specific and class-invariant features. To enhance generalization, IADG disentangles $\boldsymbol{z}$ into two representations via encoders E_c and E_d:

$$\boldsymbol{z}_c = E_c(\boldsymbol{z}), \quad \boldsymbol{z}_d = E_d(\boldsymbol{z}). \tag{1}$$

To ensure $\boldsymbol{z}_c$ and $\boldsymbol{z}_d$ capture relevant features, we impose classification losses for class (y) and domain (d):

$$\mathcal{L}^c_{cls}(\boldsymbol{x}, y) = H(\sigma(G_c(\boldsymbol{z}_c)), y), \tag{2}$$

$$\mathcal{L}^d_{cls}(\boldsymbol{x}, d) = H(\sigma(G_d(\boldsymbol{z}_d)), d), \tag{3}$$

where $G_c : \mathbb{R}^Z \rightarrow \mathbb{R}^C$ and $G_d : \mathbb{R}^Z \rightarrow \mathbb{R}^D$ are classifiers for C classes and D domains, respectively. H is the cross-entropy loss, and σ is softmax.

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 [22], 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:

$$\mathcal{L}_{orth}(\boldsymbol{x}) = s(\boldsymbol{z}_c, \boldsymbol{z}_d) \tag{4}$$

Unlike [22], 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 [25], 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 $\boldsymbol{z}_c$ and out of $\boldsymbol{z}_d$. Since we do not want any class-specific features to be encoded into $\boldsymbol{z}_d$, 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 $\mathcal{L}_{adv}$:

$$\mathcal{L}_{adv}(\boldsymbol{x}) = -I(G_c(\boldsymbol{z}_d)) \tag{5}$$

where I is the entropy function. Notice that, unlike previous DANN-based methods [26,7], we do not include the adversarial loss on the logit obtained by pushing $\boldsymbol{z}_c$ 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.

3.2 Unified Latent Space Contrastive Learning

To leverage the information contained in sample-sample relationships, we add contrastive loss term $\mathcal{L}_{contr}$. Hard negative samples can improve performance [14] because some positive pairs are difficult to optimize. 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 $\mathcal{B} = \{\boldsymbol{x}^i\}_{i=1}^B$ with class label for sample $\boldsymbol{x}^i$ being y^i and domain label being d^i. That batch of examples is encoded into disentangled representations to make a batch of class-specific and class-invariant representations. To prevent representation collapse, we additionally add small projection networks, $P_c : \mathbb{R}^Z \to \mathbb{R}^Z$ for $\boldsymbol{z}_c$ and $P_d : \mathbb{R}^Z \to \mathbb{R}^Z$ for $\boldsymbol{z}_d$ to obtain the projected feature vectors which we use in the rest of the CL method. Let $\mathcal{B}_c = \{P_c(\boldsymbol{z}_c^i)\}_{i=1}^B$ be the set of projected class-specific feature vectors and let $\mathcal{B}_d = \{P_d(\boldsymbol{z}_d^i)\}_{i=1}^B$ be the set of projected class-invariant feature vectors. For sample i, we can construct the set of positive pairs as $\mathcal{P}^i = \{p^j | p^j \in \mathcal{B}_c \setminus \boldsymbol{z_c^i} \wedge y^i = y^j \wedge d^i = d^j\}$, *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:

$$\mathcal{L}_{contr}(\mathcal{B}) = -\frac{1}{|\mathcal{B}|}\sum_{i=1}^{|\mathcal{B}|}\frac{1}{|\mathcal{P}^i|}\sum_{\boldsymbol{p}\in\mathcal{P}^i}\log\frac{\exp(\boldsymbol{z}_c^i\cdot\boldsymbol{p})}{\sum_{n\in\mathcal{N}^i}\exp(\boldsymbol{z}_c^i\cdot\boldsymbol{n})} \tag{6}$$

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.The total loss function is the weighted sum of the classification, disentanglement, and contrastive loss:

$$\begin{aligned}\mathcal{L}(\mathcal{B}) = \frac{1}{|\mathcal{B}|} \sum_{(\boldsymbol{x},y,d)\in\mathcal{B}} \big(\mathcal{L}^c_{cls}(\boldsymbol{x},y) + \lambda_d \mathcal{L}^d_{cls}(\boldsymbol{x},d) \\ +\lambda_{orth}\mathcal{L}_{orth}(\boldsymbol{x}) + \lambda_{adv}\mathcal{L}_{adv}(\boldsymbol{x})\big) \\ +\lambda_{contr}\mathcal{L}_{contr}(\mathcal{B})\end{aligned} \tag{7}$$

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

3.3 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 [28], 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 $\boldsymbol{x}$, 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 $\mathcal{A}_s$: where $a^i_w \sim \mathcal{A}_w$ is a randomly sampled weak augmentation and $a^i_s \sim \mathcal{A}_s$ is a randomly sampled strong augmentation. The features are obtained by pushing the elements of both batches through the feature extractor F and encoder E_c. 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 $\mathcal{Y}_w$. To obtain the hard pseudo label $\hat{y}$, we average all the soft pseudo labels and take the argument of the maxima.

Confidence-Weighted Loss. Given the hard pseudo label, we can calculate the standard cross-entropy classification loss $\mathcal{L}^{cls}_{tta}$. 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.

$$\mathcal{L}^{conf}_{tta} = \frac{o(\boldsymbol{y})}{|\mathcal{Y}_w|} \sum_{\boldsymbol{y}_w\in\mathcal{Y}_w} H(\boldsymbol{y}_w, \hat{y}) \tag{8}$$

where o is the confidence function. In our implementation, we use the normalized logit entropy as the confidence function. Noise sensitivity of our loss is low comparing to other high-quality pseudo-labeling methods, that makes this method computationally less heavy and widely adaptable.

Contrastive-Proxy TTA. Given a set of features $\mathcal{Z}_s$ generated from strong augmentations of the test time sample $\boldsymbol{x}$ 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 $\mathcal{M} = \{\boldsymbol{\mu}^i\}_{i=1}^C$ 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 $\mathcal{P}$ and we can also construct the negative set as the set of all projections of class prototypes except the one that is the same as the pseudo-label $\mathcal{N}$. Given the pos/neg sets, we can write the contrastive loss as:

$$\mathcal{L}_{tta}^{contr} = -\frac{o(\boldsymbol{y})}{|\mathcal{P}|} \sum_{\boldsymbol{p}\in\mathcal{P}} \log \frac{\exp(\boldsymbol{z}_p \cdot \boldsymbol{p})}{\sum_{\boldsymbol{n}\in\mathcal{N}} \exp(\boldsymbol{z}_p \cdot \boldsymbol{n})} \tag{9}$$

where $\boldsymbol{z}_p = P_c(\boldsymbol{z})$ is the projected encoded feature vector for the unaugmented sample. To prevent mode collapse we add regularization term [16]:

$$\mathcal{L}_{tta}^{reg} = \frac{1}{C}\sum_{i=1}^{C} I(G_c(\mu^i)) - I(\frac{1}{C}\sum_{i=1}^{C} G_c(\mu^i)) \tag{10}$$

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

$$\mathcal{L}_{tta} = \lambda_{conf}\mathcal{L}_{tta}^{conf} + \lambda_{contr}\mathcal{L}_{tta}^{contr} + \lambda_{reg}\mathcal{L}_{tta}^{reg} \tag{11}$$

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 Experiment Setting

Dataset. We utilize the following five DG benchmarks: PACS [11], VLCS [17], Office-Home [19], Terra Incognita [2], and DomainNet [13].

Table 1: Summary of DG benchmarks.

Dataset	Domains	Samples	Classes
PACS [11]	{art, cartoons, photos, sketch}	9,991	7
VLCS [17]	{Caltech101, LabelMe, SUN09, VOC2007}	10,729	5
Office-Home [19]	{art, clipart, product, real-world}	15,588	65
Terra Incognita [2]	{L100, L38, L43, L46}	24,788	10
DomainNet [13]	{clipart, infograph, painting, quickdraw, real, sketch}	586,575	345

Implementation Details. We use ResNet-50 [9] as the feature extractor, freezing all batch normalization layers and adding a dropout layer. Both encoder and projection networks are three-layer MLPs with ReLU and batch normalization. We optimize using Adam and set batch size to 32 per domain for

Table 2: Performance comparison of IADG with SOTA DG methods. Results are shown in accuracy (%). Best results are in **bold**.

Methods	PACS	OfficeHome	TerraInc	DomainNet	VLCS	Avg.
Without TTA						
ERM [18]	85.5 ± 0.1	66.5 ± 0.3	46.1 ± 1.0	41.3 ± 0.3	77.5 ± 0.6	63.4
ITTA [6]	83.8 ± 0.3	62.0 ±0.2	43.2 ± 0.5	34.9 ± 0.1	76.9 ± 0.6	60.2
MIRO [4]	85.4 ± 0.4	70.5 ± 0.6	50.4 ± 0.9	46.0 ± 0.3	79.0 ± 0.1	66.5
SWAD [3]	88.1 ± 0.2	70.6 ± 0.3	50.0 ± 0.6	46.5 ± 0.6	79.1 ± 0.2	66.9
PCL [23]	88.7 ± 0.3	71.6 ± 0.3	52.1 ± 0.5	47.7 ± 0.4	76.3 ± 0.4	67.2
D-Net [10]	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**

fair comparison. Hyperparameters (learning rate, weight decay, loss coefficients, and iterations) are tuned per experiment, following [8].

4.2 Performance Evaluation

Main Results. Table 2 presents our main results on five benchmark datasets. For fair comparison, we use ResNet-50 [9] as the backbone and compare against DG baselines. Our base algorithm, IADG, can be combined with SWAD [3] and MIRO [4] for improved results. IADG + SWAD achieves state-of-the-art performance on all datasets except Terra Incognita. IADG + TTA further improves results, achieving the best accuracy on PACS, OfficeHome, DomainNet, and VLCS, with a 1.1% average gain over D-Net [10], though it shows little improvement on Terra Incognita. Even IADG alone outperforms D-Net by 0.4% on average. Additionally, IADG + TTA surpasses IADG + TENT [20] by 0.4%. These results confirm that leveraging class-specific and domain-specific features enhances generalization to unseen domains.

Comparison with TTA Methods. Figure 3 highlights that our approach consistently achieves higher performance metrics compared to other TTA methods such as AdaContrast [5], EATA [12], and TENT [20] . This superiority is reflected in both individual dataset accuracies and the overall average accuracy. Due to the space limitations, we could not include the results on DomainNet, TerraInc and VLCS, but they show very similar trend.

4.3 TTA Efficiency & Complexity Analysis

As shown in Table 3, our method demonstrates lower parameter usage and runtime while achieving better performance compared to TENT [20]. 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.

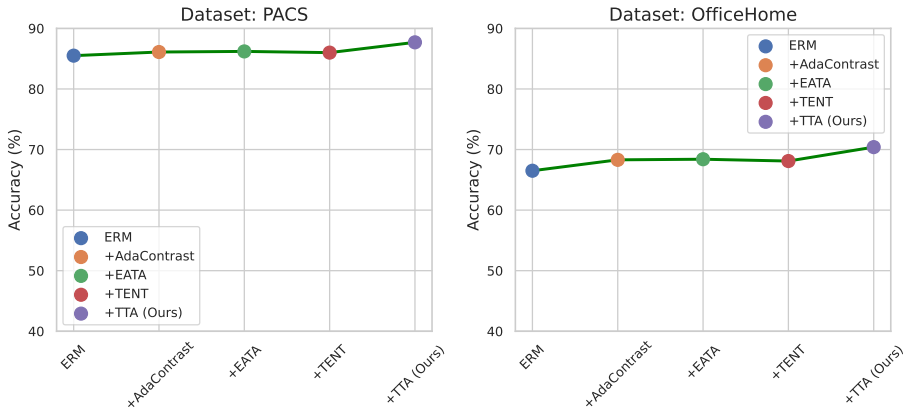

Fig. 3: Comparison of TTA methods.

Table 3: 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 [20] (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

5 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 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.

Acknowledgments. This work was partly supported by (1) the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (RS-2024-00345398) and (2) the Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT)(RS-2020-II201373,Artificial Intelligence Graduate School Program (Hanyang University)).

References

1. Agarwal, A., et al.: Mvtamperbench: Evaluating robustness of vision-language models. arXiv preprint arXiv:2412.19794 (2024)
2. Beery, S., Van Horn, G., Perona, P.: Recognition in terra incognita. In: Proceedings of the European conference on computer vision (ECCV). pp. 456–473 (2018)

3. Cha, J., Chun, S., Lee, K., Cho, H.C., Park, S., Lee, Y., Park, S.: Swad: Domain generalization by seeking flat minima. NeurIPS **34**, 22405–22418 (2021)
4. Cha, J., Lee, K., Park, S., Chun, S.: Domain generalization by mutual-information regularization with pre-trained models. In: ECCV. pp. 440–457. Springer (2022)
5. Chen, D., Wang, D., Darrell, T., Ebrahimi, S.: Contrastive test-time adaptation. In: CVPR. pp. 295–305 (2022)
6. Chen, L., Zhang, Y., Song, Y., Shan, Y., Liu, L.: Improved test-time adaptation for domain generalization. In: CVPR. pp. 24172–24182 (2023)
7. Ganin, Y., et al.: Domain-adversarial training of neural networks. JMLR **17**(1), 2096–2030 (2016)
8. Gulrajani, I., Lopez-Paz, D.: In search of lost domain generalization. arXiv preprint arXiv:2007.01434 (2020)
9. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016)
10. Hu, L., et al.: Dandelionnet: Domain composition with instance adaptive classification for domain generalization. In: CVPR. pp. 19050–19059 (2023)
11. Li, D., Yang, Y., Song, Y.Z., Hospedales, T.M.: Deeper, broader and artier domain generalization. In: ICCV. pp. 5542–5550 (2017)
12. Niu, S., Wu, J., Zhang, Y., Chen, Y., Zheng, S., Zhao, P., Tan, M.: Efficient test-time model adaptation without forgetting. In: ICML. pp. 16888–16905 (2022)
13. Peng, X., Bai, Q., Xia, X., Huang, Z., Saenko, K., Wang, B.: Moment matching for multi-source domain adaptation. In: ICCV. pp. 1406–1415 (2019)
14. Robinson, J., Chuang, C.Y., Sra, S., Jegelka, S.: Contrastive learning with hard negative samples. arXiv preprint arXiv:2010.04592 (2020)
15. Song, J., Lee, J., Kweon, I.S., Choi, S.: Ecotta: Memory-efficient continual test-time adaptation via self-distilled regularization (2023)
16. Springenberg, J.T.: Unsupervised and semi-supervised learning with categorical generative adversarial networks. arXiv preprint arXiv:1511.06390 (2015)
17. Torralba, A., Efros, A.A.: Unbiased look at dataset bias. In: CVPR 2011
18. Vapnik, V.N.: An overview of statistical learning theory. IEEE transactions on neural networks **10**(5), 988–999 (1999)
19. Venkateswara, H., Eusebio, J., Chakraborty, S., Panchanathan, S.: Deep hashing network for unsupervised domain adaptation. In: CVPR. pp. 5018–5027 (2017)
20. Wang, D., Shelhamer, E., Liu, S., Olshausen, B., Darrell, T.: Tent: Fully test-time adaptation by entropy minimization. In: ICLR (2021)
21. Wang, Q., Fink, O., Van Gool, L., Dai, D.: Continual test-time domain adaptation. In: CVPR. pp. 7201–7211 (2022)
22. Wu, A., Liu, R., Han, Y., Zhu, L., Yang, Y.: Vector-decomposed disentanglement for domain-invariant object detection. In: ICCV. pp. 9342–9351 (2021)
23. Yao, X., et al.: Pcl: Proxy-based contrastive learning for domain generalization. In: CVPR. pp. 7097–7107 (2022)
24. Zhang, J., Qi, L., Shi, Y., Gao, Y.: Domainadaptor: A novel approach to test-time adaptation. In: ICCV. pp. 18971–18981 (2023)
25. Zhao, H., Des Combes, R.T., Zhang, K., Gordon, G.: On learning invariant representations for domain adaptation. In: ICML. pp. 7523–7532. PMLR (2019)
26. Zhao, S., Gong, M., Liu, T., Fu, H., Tao, D.: Domain generalization via entropy regularization. NeurIPS **33**, 16096–16107 (2020)
27. Zhou, K., Yang, Y., Qiao, Y., Xiang, T.: Domain generalization with mixstyle. arXiv preprint arXiv:2104.02008 (2021)
28. Zou, Y., Yu, Z., Liu, X., Kumar, B.V.K.V., Wang, J.: Confidence regularized self-training. CoRR **abs/1908.09822** (2019), `http://arxiv.org/abs/1908.09822`