Feature Alignment and Uniformity for Test Time Adaptation

Shuai Wang[1] Daoan Zhang[2] Zipei Yan[3] Jianguo Zhang[2] Rui Li[1]
[1]Tsinghua University [2]Southern University of Science and Technology
[3]Hongkong Polytechnic University
s-wang20@mails.tsinghua.edu.cn, dawn_change@126.com, zipei.yan@connect.polyu.hk
zhangjg@sustech.edu.cn, leerui@tsinghua.edu.cn

Abstract

Test time adaptation (TTA) aims to adapt deep neural networks when receiving out of distribution test domain samples. In this setting, the model can only access online unlabeled test samples and pre-trained models on the training domains. We first address TTA as a ***feature revision*** *problem due to the domain gap between source domains and target domains. After that, we follow the two measurements* ***alignment*** *and* ***uniformity*** *to discuss the test time feature revision. For test time feature uniformity, we propose a* ***test time self-distillation*** *strategy to guarantee the consistency of uniformity between representations of the current batch and all the previous batches. For test time feature alignment, we propose a* ***memorized spatial local clustering*** *strategy to align the representations among the neighborhood samples for the upcoming batch. To deal with the common noisy label problem, we propound the entropy and consistency filters to select and drop the possible noisy labels. To prove the scalability and efficacy of our method, we conduct experiments on four domain generalization benchmarks and four medical image segmentation tasks with various backbones. Experiment results show that our method not only improves baseline stably but also outperforms existing state-of-the-art test time adaptation methods. Code is available at https://github.com/SakurajimaMaiii/TSD.*

1. Introduction

Deep learning has achieved great success in computer vision tasks when training and test data are sampled from the same distribution [17, 28, 39]. However, in real-world applications, performance degradation usually occurs when training (source) data and test (target) data are collected from different distributions, *i.e. domain shift*. In practice, test samples may encounter different types of variations or corruptions. Deep learning models will be sensitive to these variations or corruptions, which can cause performance degradation.

To tackle this challenging but practical problem, various works have been proposed to adapt the model at *test* time [5, 8, 22, 38, 45, 57, 66, 67, 76]. Test time training (TTT) [38, 57] adapts model using self-supervised tasks, such as rotation classification [15] during both training and test phases. This paradigm relies on additional model modifications in both training and test phases, which is not feasible and scalable in the real world.

Similar to TTT, test time adaptation [66] (TTA) also adapts the model *i.e.*, updates the parameters in the test phase. But TTA does not require any specific modifications in training and requires only the pre-trained source model and unlabeled target data during the test phase, which is more practical and generalizable. In TTA, the model could be adapted with only online unlabeled data. Hence, the model trained on source data is incompatible with the target data due to the possible domain gap between source data and target data.

To deal with the above-mentioned problem, we address TTA as a *representation revision* problem in this paper. In the test phase of TTA, the accessed model has already learned the feature representations specialized to source domains and may generate inaccurate representations for the target domain due to the large domain gap. It is necessary to rectify the feature representations for the target domain. To achieve better representations for the target domain, we utilize the commonly used measurements for representation quality that can be summarized to feature *alignment* and *uniformity* [68, 73]. Alignment refers that the similar images should have the similar representations, while uniformity means that images of different classes should be distributed as uniform as possible in the latent space. Hence, we propose to address the TTA problem from the above-mentioned properties.

Most of the previous works about TTA can be inducted from the proposed representation revision perspective. Some methods adapt the source model by conducting a feature alignment process, such as feature matching [27, 38] and predictions adjustment [5]. One of the repre-

sentative methods is LAME [5], which encourages neighborhood samples in the feature space to have similar predictions using Laplacian adjusted maximum-likelihood estimation. Moreover, other methods aim to make target feature more uniform in feature space, including entropy minimization [45, 66, 76], prototype adjustment [22], information maximization [36] and batch normalization statistics alignment [35, 44, 53]. One representative method is T3A [22], which adjusts prototypes (class-wise centroids) to have a more uniform representation by building a support set. However, none of the method address the TTA problem from the representation alignment and uniformity simultaneously. In this paper, we identify this limitation and propose a novel method that rectifies feature representation from both two properties. We formulate the two properties in TTA as *test time feature uniformity* and *test time feature alignment*.

Test Time Feature Uniformity. Following the feature uniformity perspective, we hope that representations of test images from different classes should be distributed as uniform as possible. However, only limited test samples can be accessed in an *online* manner in the TTA setting.

To better deal with all of the samples in the target domain, we propose to introduce historical temporal information for every arriving test sample. A memory bank is built to store feature representations and logits for all arriving samples to maintain the useful information from the previous data. We then calculate the pseudo-prototypes for every class by using the logits and features in the memory bank. After that, to guarantee the uniformity for the current batch of samples, the prediction distribution of prototype-based classification and model prediction (outputs of linear classifier) should be similar, *i.e.*, the feature distribution of the current images of one class should be consistent with the feature distribution of all the previous images of the same class. This can reduce the bias of misclassified outlier samples to form a more uniform latent space.

Motivated by this, we minimize the distance between the outputs of linear and prototype-based classifiers. This pattern is similar to self-distillation [25, 75] that transfers knowledge between different layers of the same network architecture. However, unlike typical self-distillation, our method does not require any ground truth supervision. We refer to this method as *Test Time Self-Distillation* (TSD).

Test Time Feature Alignment. Feature alignment encourages the images from the same class to have similar feature representations in the latent space. As for TTA, pseudo labels generated by the source model may be noisy due to the domain gap. Thus, instead of aligning all the positive pairs, we propose a *K-nearest feature alignment* to encourage features from the same class to be closed or features from different classes to be far away from each other. This can reduce the negative impact imposed by the noisy labels and maintain the alignment of images with the same semantics. Specifically, we retrieve K-nearest features in the memory bank for the upcoming images and add consistency regularization between the representations and logits of the images. We refer to this as *Memorized Spatial Local Clustering* (MSLC). The ablation of hyperparameter K is shown in Table 5 and Fig. 3.

Entropy Filter and Consistency Filter. During the adaption, we use stored pseudo features and logits to compute the pseudo-prototypes. However, the noisy label problem cannot be completely alleviated despite the proposed efforts. To further reduce the impact, we adopt both *entropy* and *consistency* filters to filter noisy labels to boost performance. As for the entropy filter, we filter noisy features with high entropy when we compute prototypes because unreliable samples usually produce high entropy.

In addition, the predictions of prototype-based and linear classifiers of the network for reliable samples should be consistent ideally. We use this property to filter unreliable samples and back-propagate the gradient using only reliable samples. We refer to this filter as the consistency filter. The ablation study on the two proposed filters is presented in Table 5.

Finally, we demonstrate the effectiveness of our proposed approach on commonly used domain generalization benchmarks, including PACS [32], VLCS [60], OfficeHome [64] and DomainNet [49]. Furthermore, to prove the efficacy of our method, we conduct more experiments on four cross-domain medical image segmentation benchmarks, including prostate segmentation [1, 37], cardiac structure segmentation [4, 78, 79] and optic cup/disc segmentation [13, 47, 55]. Our method achieves the state-of-the-art performance on the above benchmarks.

We summarize our contributions as follows:

- We propose a new perspective for test time adaptation from the view of feature alignment and uniformity. The proposed test time feature uniformity encourages the representations of the current batch of samples along with the uniformity of all the previous samples. The test time feature alignment manipulates the representation of the test sample according to its neighbors in the latent space to align the representations based on the pseudo label.
- Specifically, to meet the online setting and noisy label problem in TTA, we propose two complementary strategies: unsupervised self-distillation for test time feature uniformity and memorized spatial local clustering for test time feature alignment. We also propound the entropy filter and consistency filter to further mitigate the effect of the noisy labels.
- The experiments demonstrate that our proposed method outperforms existing test time adaptation ap-

proaches on both the domain generalization benchmarks and medical image segmentation benchmarks.

2. Related Work

2.1. Domain Generalization

Domain generalization (DG) aims to learn knowledge from multi-source domains and generalize to the unseen target domain. A primary strategy of DG is domain-invariant feature learning which aims to reduce domain gaps, including aligning distributions among multiple domains with contrastive learning [24, 43, 70, 72], learning useful representations with self supervise learning [6], matching statistics of feature distributions across domains [56,61], domain adversarial learning [14,33] and causality inference [40,41]. Meta-learning based methods [2, 12, 31] simulate domain shift by dividing meta-train and meta-test domains from the original source domains. Data augmentation-based methods effectively solve the DG problem by diversifying training data [19,34,46,65,71,74,77].

Unlike the DG paradigm that focuses on training a generalizable model during the training phase with labeled source data, test time adaptation aims to adapt a pre-trained model in the test phase by using the online unlabeled target data.

2.2. Test Time Adaptation

Our work is mostly related to test time adaptation [5, 8, 22, 45, 66] or test time training [38, 57]. Test Time Training (TTT) [38, 57] adapts the model during the test phase via the self-supervise task, such as rotation classification [15]. However, TTT needs to optimize the source model by jointly training supervised loss and self-supervised loss, which may not be feasible in the real world. Different from this, test time adaptation [66] aims to adapt the model during the test phase without changing the training process. Tent [66] minimizes entropy to update the trainable parameters in Batch Normalization [21] layer. SHOT [36] combines information maximization and pseudo labeling. There are some works [45,67] combing test time adaptation and continual learning to maintain the performance on the source domain. LAME [5] adapts the output of the model rather than parameters with a laplacian adjusted maximum-likelihood estimation. There are also some works using test time Batch Normalization statistics [35, 44, 53], self-training [8] and feature alignment [27,38]. The above methods could be viewed as either feature uniformity or feature alignment as discussed in the Sec. 1, while our method benefits from both properties.

3. Method

Fig. 1 illustrates the overall pipeline of our method. We describe the details of the problem settings and our method in this section.

3.1. Preliminaries

In test time adaptation (TTA), we can only achieve target domain unlabeled images in an online manner and pre-trained model on the source domain. The source model is trained with standard empirical risk minimization on source domains, *e.g.* cross entropy loss for the image classification task. Given the model trained on $\mathcal{D}_s$, we aim to adapt the model using unlabeled target data $\{x_i\} \in \mathcal{D}_t$, $i \in \{1...N\}$, where x_i denotes the ith image of target domain $\mathcal{D}_t$ and N denotes the number of target images, $\mathcal{D}_s$ denotes the source domain. During testing, we initialize the model $g = f \circ h$ with source model parameters trained on source domain $\mathcal{D}_s$, where f denotes backbone and h denotes linear classification head. The output of model g for image x_i is denoted as $p_i = g(x_i) \in \mathbb{R}^C$ where C is the number of class.

3.2. Test Time Self-Distillation

During adaption, given a batch of unlabeled test samples, we can generate image embeddings $z_i = f(x_i)$, logits $p_i = h(z_i)$ and pseudo labels $\hat{y}_i = \arg\max p_i$ via the pre-trained model.

We then maintain a memory bank $\mathcal{B} = \{(z_i, p_i)\}$ to store image embeddings z_i and logits p_i. Following T3A [22], the memory bank is initialized with the weights of the linear classifier. When the target sample x_i comes, for every image, we add the image embedding z_i and logits p_i into the memory bank. To build up the relations between the current samples and all of the previous samples, the pseudo-prototypes shall be generated for each class. The prototype of class k could be formulated as

$$c_k = \frac{\sum_i z_i \mathbb{1}[\hat{y}_i = k]}{\sum_i \mathbb{1}[\hat{y}_i = k]}, \tag{1}$$

where $\mathbb{1}(\cdot)$ is an indicator function, outputting value 1 if the argument is true or 0 otherwise. However, some pseudo labels may be assigned to the wrong class, leading to incorrect prototype computation. We use Shannon entropy [54] filter to filter noisy labels. For prediction p_i, its entropy could be computed as $H(p_i) = -\sum \sigma(p_i) \log \sigma(p_i)$ where σ denotes the softmax operation. We aim to filter unreliable features or predictions with high entropy because lower entropy usually means higher accuracy [66]. Specifically, for every class, image embeddings with the top-M highest entropy in the memory bank would be ignored. After that, we use filtered embeddings to calculate prototypes as Eq. 1 and define prototype-based classification output as the softmax over the feature similarity to prototypes for class k:

$$y_i^k = \frac{\exp\left(\text{sim}(z_i, c_k)\right)}{\sum_{k'=1}^{C} \exp\left(\text{sim}(z_i, c_{k'})\right)}, \tag{2}$$

where $\text{sim}(z_i, c_k)$ denotes cosine similarity between z_i and c_k.

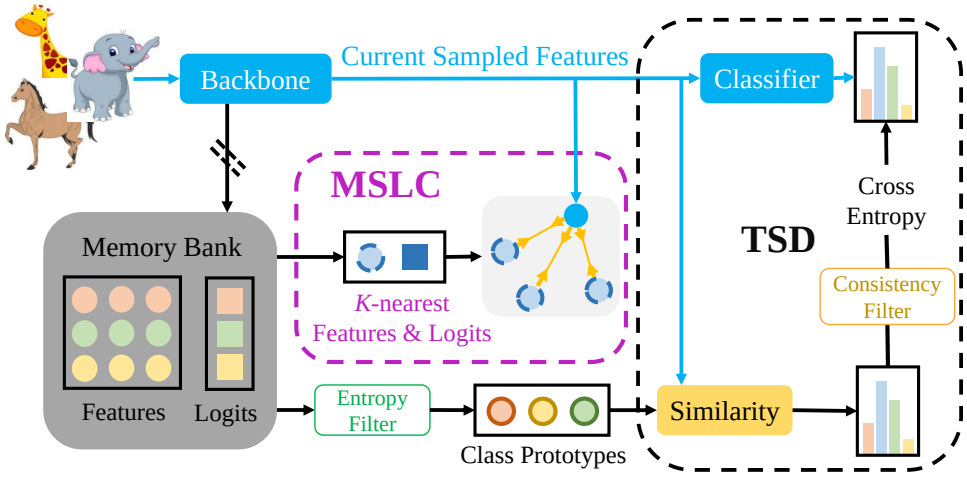


Figure 1. **Overview of our proposed method**. Blue lines denote forward and backward and black lines denote only forward (*i.e.* without gradient backpropagation). Different colours of features, logits and prototypes mean different classes. MSLC: Memorized Spatial Local Clustering. TSD: Test Time Self-Distillation.

The prototype-based classification results y_i and outputs p_i of the network g should share a similar distribution for the same input. Thus, the loss to maintain uniformity is proposed as

$$\mathcal{L}_i(p_i, y_i) = -\sigma(p_i) \log y_i. \tag{3}$$

Note that p_i is a soft pseudo label rather than a hard pseudo label. The reason for using a soft label is that a soft label usually provides more information [20]. By using the proposed test time self-distillation, the network could map the uniformity for the current samples to improve the quality of representations.

Although we use an entropy filter to drop noisy labels when computing prototypes, there are still some mistake predictions inevitable. We propose that, for a reliable sample, the outputs of the linear fully connected layer and prototype-based classifier should be similar. Hence, we adopt the consistency filter to identify the mistake predictions. Specially, if the linear classifier and prototype-based classifier produce the same predictions, *i.e.* the same result after executing $argmax$ to the logits, we assume that this sample is reliable. This strategy could be implemented using a filter mask for image x_i as follows

$$\mathcal{M}_i = \mathbb{1}[\arg\max p_i = \arg\max y_i]. \tag{4}$$

By conducting a consistency filter, we further filter unreliable samples and the unsupervised self-distillation loss could be formulated as follows

$$\mathcal{L}_{tsd} = \frac{\sum_i \mathcal{L}_i * \mathcal{M}_i}{\sum_i \mathcal{M}_i}. \tag{5}$$

3.3. Memorized Spatial Local Clustering

As mentioned before, features belonging to the same class should be aligned in the latent space. However, this situation may differ in TTA due to the domain gap between the target and source domains. We encourage K-nearest neighborhood features instead of all the features to be close to reducing the noisy label impact. A simple strategy is to add consistency regularization within a batch of samples. However, historical temporal information is ignored and the alignment is less effective. Moreover, there is a trivial solution that the model can easily map all images to a certain class if we use only one batch sample for alignment [3].

To handle these problems, we concatenate spatial local clustering with the memory bank. We begin from retrieving K-nearest features in the memory bank for image x. Based on our assumption, the logits of image x should be aligned with the logits of its nearest neighbors in the latent space. To achieve this, we align the two varieties of logits according to the distance between the image embeddings of image x and its neighbors. The formulation is presented as follows

$$\mathcal{L}_{mslc} = \frac{1}{K} \sum_{j=1}^{K} \text{sim}(z, z_j)(\sigma(p) - \sigma(p_j))^2, \tag{6}$$

where $\text{sim}(z, z_j)$ denotes cosine similarity between z and z_j. $\{z_j\}_{j=1}^{K}$ denotes the K-nearest image embeddings of z in the memory bank $\mathcal{B}$ and p_j denotes the corresponding logits. If z_j and z are close in feature space, *i.e.* $\text{sim}(z, z_j)$ is large, this objective function will push p_j and p to get close. We detach the gradient of $\text{sim}(z, z_j)$, *i.e.* $\text{sim}(z, z_j)$ would be viewed as constant, to avoid the trivial solution that the model will output a constant result regardless of different samples.

3.4. Training Objective Function

Combing Eq. 5 and Eq. 6, we formulate the final objective function as

$$\mathcal{L} = \mathcal{L}_{tsd} + \lambda \mathcal{L}_{mslc}, \tag{7}$$

where λ is the trade-off parameter to balance different loss functions. In our implementation, we use cosine similarity as the similarity metric. Specifically, we define $\text{sim}(x, y) = x^\top y/\|x\|\|y\|$.

During testing phase, the adaptation is performed in an *online* manner. Specifically, when receiving image x_T at time point T, the model state is initialized with the parameters update from the last image x_{T-1}. The model produces the prediction $p_T = g(x_T)$ after receiving new samples x_T and updates the model using Eq. 7 with only *one* step gradient descent. It is noticed that the adaption could continue as long as there exists test data.

4. Experiments

4.1. Experimental Setup

Datasets. **PACS** [32] contains 9,991 examples and 7 classes that are collected from 4 domains: art, cartoons, photos, and sketches. **OfficeHome** [64] is consisted of 4 domains: art, clipart, product, and real, which includes 15,588 images and 65 classes. **VLCS** [60] comprises four domains: Caltech101, LabelMe, SUN09 and VOC2007 and includes 10,729 images and 5 classes. **DomainNet** [49], a large-scale dataset has six domains $d \in$\{clipart, infograph, painting, quickdraw, real, sketch\} with 586,575 images and 345 classes.

Models. In the main experiments, we evaluated different methods on ResNet-18/50 [17] equipped with Batch Normalization [21], which is widely used in domain adaptation and generalization literature. Furthermore, we tested our algorithm on different backbones, including Vision Transformer (ViT-B/16) [11], ResNeXt-50 (32×4d) [69], EfficientNet (B4) [58] and MLP-Mixer (Mixer-L16) [59].

Implementation. For source training, we choose one domain as the target domain and the other domains as the source domains. We split all images from the source domains to 80%/20% for training and validation. We use the Adam optimizer [26] with $5e^{-5}$ as the learning rate. All models are initialized with ImageNet-1K [51] pre-trained weights except for ViT-B/16 and MLP-Mixer when we use ImageNet-21K pre-trained weights. We use `torchvision` implementations of all models except for ViT-B/16 and MLP-Mixer; instead, we use the implementations in the `timm` library.

For test time adaptation, we use the Adam optimizer [26] and set the batch size as 128. We empirically set the trade-off parameter $\lambda = 0.1$ (*cf*. Eq. 7). Unlike [45, 66], all the trainable layers are updated and no special selection is required in our method. We use PyTorch [48] for all implementations. We report the accuracy of the whole target domain for evaluation. For all experiments, we report the average of three repetitions with different weight initialization, random seed and data split. Please refer to our supplementary material for more details, including detailed results and error bars.

Table 1. Comparison of our method and existing test time adaptation methods with ResNet18 backbone. We highlight the **best** and the second results.

Method	PACS	OfficeHome	VLCS	DomainNet	Avg.
ERM [63]	82.07	63.12	72.75	38.95	64.22
BN [53]	82.82	62.30	64.31	37.80	61.81
Tent [66]	84.92	63.75	67.36	38.95	63.75
PL [30]	84.64	60.22	68.93	35.23	62.26
SHOT-IM [36]	82.55	63.42	64.90	39.50	62.59
T3A [22]	83.50	64.25	73.03	39.61	65.10
ETA [45]	82.70	62.46	64.35	39.43	62.24
LAME [5]	84.58	62.20	72.88	37.49	64.29
Ours	**87.32**	**64.83**	**73.61**	**40.19**	**66.49**

Table 2. Comparison of our method and existing test time adaptation methods with ResNet50 backbone. We highlight the **best** and the second results.

Method	PACS	OfficeHome	VLCS	DomainNet	Avg.
ERM [63]	84.59	67.37	74.01	45.20	67.74
BN [53]	85.03	66.10	64.78	43.38	64.82
Tent [66]	87.48	67.96	69.20	44.71	67.34
PL [30]	85.23	67.13	68.52	41.18	65.52
SHOT-IM [36]	85.50	67.39	65.23	46.30	66.11
T3A [22]	86.04	68.29	73.98	46.16	68.62
ETA [45]	85.04	66.21	64.79	46.13	65.54
LAME [5]	86.62	66.19	73.94	43.20	67.49
Ours	**89.41**	**68.67**	**74.52**	**47.73**	**70.08**

Hyperparameter search and model selection. We use training domain validation [16] for source model training. We select the model with the highest accuracy on the validation set. For hyperparameter search, we search learning rate lr in $\{1e^{-3}, 1e^{-4}, 1e^{-5}, 1e^{-6}\}$, the hyperparameter for feature filter $M \in \{1, 5, 20, 50, 100, \text{NA}\}$ where NA denotes no entropy filter. We emphasize that all hyperparameters in the TTA setting should be selected before accessing test samples. We do hyperparameter search on *training domain validation set*.

Baselines. We compared our method with Empirical Risk Minimization (ERM) [63], Tent [66], T3A [22], SHOT-IM [36], ETA [45], Test Time Batch Normalization (BN) [53], Laplacian Adjusted Maximumlikelihood Estimation (LAME) [5] and PseudoLabeling (PL) [30]. We use the implementation from the released code of `T3A` library[1], except for LAME and ETA, we use the source code of authors.

4.2. Comparative Study

Comparison with TTA methods. Table 1 and Table 2 present the results on ResNet-18/50 of four different datasets. From Table 1 and 2, we can see that our method

[1] https://github.com/matsuolab/T3A

Table 3. Compare with domain generalization methods on PACS dataset with ResNet50. †: numbers are from the original literature. ‡ denotes our implementation results. We highlight the **best** and the second results in each column.

Method	A	C	P	S	Avg.
ERM [63]	82.5	80.8	94.1	81.0	84.6
DNA† [10]	89.8	83.4	97.7	82.6	88.4
PCL† [70]	90.2	83.9	**98.1**	82.6	88.7
SWAD‡ [7]	89.3	83.2	96.9	83.4	88.2
Ours	87.7	88.8	96.2	85.0	89.4
SWAD + Ours	**92.2**	**89.2**	97.1	**85.6**	**91.0**

generally achieves the state-of-the-art performance. Specifically, the proposed method improves the ERM [63] baseline with 4.8%, 1.3%, 0.5%, 2.53% for each dataset, respectively. Other test time adaptation methods do not improve the baseline as stably as ours.

We also visualized the change in accuracy of different methods throughout the adaption process which is shown in Fig. 2. The "Batch Numbers" in Fig. 2 indicates how many batches of images that the model has been updated with. We can see that our method could adapt the data much faster and achieve higher accuracy on the target domain.

Comparison with DG/SFDA methods. The above experiments mainly focus on test time adaptation that aims to adapt the model during *test* phase. It is natural to ask: *how about our method compared with domain generalization or source free domain adaptation methods?* To answer this question, we first compared our method with some recent domain generalization or source free domain adaptation methods, such as SWAD [7], PCL [70], DNA [10], and F-mix [29][2] on PACS and DomainNet dataset. From Table 3, we can see that our method outperforms the state-of-the-art methods in domain generalization. Furthermore, combining SWAD [7], we achieve the impressive 91% accuracy with ResNet50 backbone.

We also report the result of the challenging DomainNet dataset. The results are listed in Table 4. It can be seen that our method outperforms the state-of-the-art methods of domain generalization, such as SWAD [7] and DNA [10]. Also, our method could significantly improve SWAD [7], which outperforms the current state-of-the-art SFDA method [29]. Note that online test time adaptation is more flexible in the real world because SFDA adapts the test data in an offline manner which requires more training loops and resources than TTA.

To summarize, our model can outperform all the domain generalization and test time adaptation methods on various datasets.

[2]"F-mix" denotes *feature-mixup* in [29].

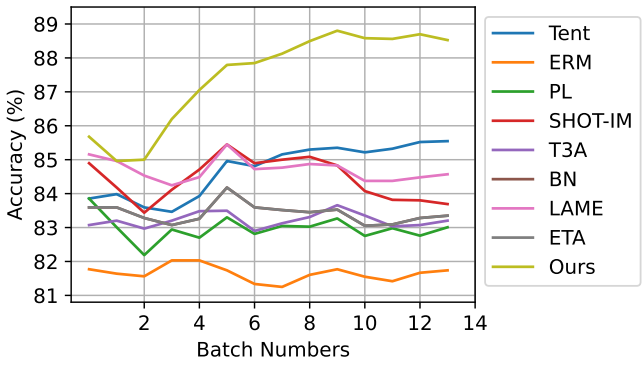


Figure 2. Accuracy visualizations of different methods during the adaptation process on the PACS dataset (target domain: art). The "Batch Numbers" indicates how many batches of images the model has been updated with.

Table 4. Compare with domain generalization methods and source free domain adaptation methods on DomainNet dataset with ResNet50. †: numbers are from the original literature. ‡ denotes our implement results. We highlight the **best** and the second results in each column.

Method	clip	info	paint	quick	real	sketch	Avg.
Domain generalization methods							
ERM [63]	64.8	22.1	51.8	13.8	64.7	54.0	45.2
PCL† [70]	67.9	24.3	55.3	15.7	66.6	56.4	47.7
DNA† [10]	66.1	23.0	54.6	16.7	65.8	56.8	47.2
SWAD‡ [7]	66.1	22.4	53.6	16.3	65.5	56.2	46.7
Source free domain adaptation methods							
F-mix† [29]	**75.4**	24.6	57.8	23.6	65.8	58.5	51.0
Ours	66.1	24.1	52.8	18.2	**68.5**	56.7	47.7
SWAD + Ours	69.2	**28.4**	**58.2**	**26.2**	68.1	**59.6**	**51.6**

4.3. Ablation Study

In this section, we mainly conduct various ablation studies on the PACS dataset with ResNet50 backbone.

Effectiveness of key components. Table 5 shows the contribution of each term mentioned in our method. Compared with the ERM baseline, self-distillation improves the baseline 3.2%. This shows that our method could efficiently capture historical temporal features to maintain uniformity. The entropy filter and consistency filter improve the performance with 0.6% and 0.5%, respectively, which suggests that they mitigate the problem of noise pseudo labels. Finally, our proposed memorized spatial local clustering brings a 0.5% gain which suggests that the module also helps with the revision of representations.

Sensitivity to hyperparameter. Our method has three hyperparameters: top-M features for entropy filtering to compute prototypes, the number of nearest neighbors K in memorized spatial local clustering and the trade-off pa-

Table 5. Ablation Study. SD means self-training using self-distillation (*cf*. Eq 3) without any filter. EF and CF mean entropy filter and consistency filter. MSLC means memorized spatial local clustering loss (*cf*. Eq. 6).

#	SD	EF	CF	MSLC	Acc(% ↑)
0					84.59
1	✓				87.80 (+3.21)
2	✓	✓			88.40 (+3.81)
3	✓	✓	✓		88.92 (+4.33)
4	✓	✓	✓	✓	89.41 (+4.82)

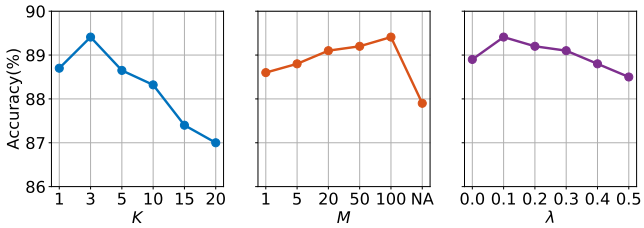


Figure 3. Sensitivity analysis about the number of nearest neighbors K in memorized spatial local clustering, entropy filter hyperparameter M and trade-off parameter λ (*cf*. Eq. 7).

rameter λ in Eq. 7. Fig. 3 presents the ablation study on hyperparameters. First, for the number of nearest neighbors K, when $K \in \{1, 3, 5\}$ achieves good performance. However, though it stills improves the ERM baseline, $K \in \{10, 15, 20\}$ leads to performance degradation. We conjecture that using too many neighbors may increase the probability of introducing misclassified features. Second, for the entropy filter hyperparameter M, we found that the proposed method is robust to the choice of M. The performance improves stably with the M increasing from 1 to 100. If we mute the entropy filter, our method still achieves a competitive performance. Third, for trade-off parameter λ, $\lambda \in [0.1, 0.5]$ achieves good performance and $\lambda = 0.1$ achieves the best performance.

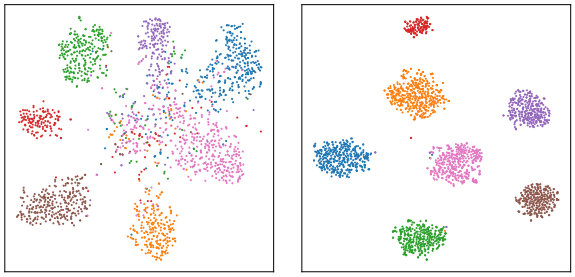
4.4. Analysis

Scalability on different models. We validated our method on various backbones to verify that our method does not rely on any specific network structure design. Table 6 shows that the proposed method generally improves baseline performance regardless of different backbones. For example, for ViT-B/16, the proposed method improves ERM [63] 3.1%, 2.7%, 1.2% for different datasets, respectively. This show that our method is " plug and play " for different deep neural networks which is important for applications in the real world.

Qualitative analysis by tSNE. We provide tSNE [62] visualizations of the ERM baseline and our method, as shown in Fig. 4. The learned features of the ERM baseline on the target domain are not well separated due to the large domain gap, as shown in Fig. 4a. After adaptation to the target domain, our method could generate more uniform and alignment features, as shown in Fig. 4b.

Table 6. Results with different backbones. **Bold** type indicates performance improvement compared with baseline. All baseline models are trained in a standard ERM manner.

Backbones	PACS	OfficeHome	VLCS
ResNet18 [17]	82.07	63.12	72.75
+Ours	**87.32**	**64.83**	**73.61**
ResNet50 [17]	84.59	67.37	74.01
+Ours	**89.41**	**68.67**	**74.52**
ResNeXt-50 [69]	86.67	72.66	78.50
+Ours	**91.33**	**74.18**	**79.38**
ViT-B/16 [11]	87.13	79.06	78.70
+Ours	**90.20**	**81.80**	**79.90**
EfficientNet-B4 [58]	85.11	74.65	77.14
+Ours	**85.41**	72.24	**79.42**
Mixer-L16 [59]	84.59	71.36	76.53
+Ours	**88.47**	**74.82**	**79.75**

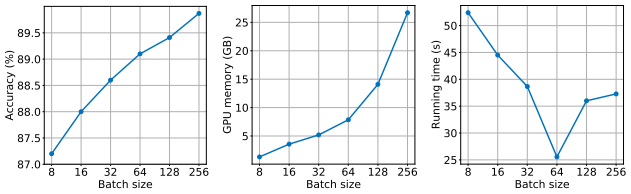


(a) ERM (b) Our method

Figure 4. tSNE [62] visualization of learned feature embeddings. (a) ERM baseline without adaptation. (b) After adaptation with our method. Different colours denote different classes.

Efficiency analysis. Our method adapts a pre-trained model and needs to maintain a memory bank, which may lead to extra computational costs. The usage of GPU memory is mainly influenced by the batch size at test time. We present computational complexity analysis in Fig. 5. The default batch size of our method is 128 and the reported accuracy is 89.4% with an almost 14 GB GPU memory cost. While our method still achieves good performance (89.1% accuracy) when we set the batch size to 64, the GPU memory cost decreases to 7.85 GB and the running time is the fastest for all. When we apply our method to the real world applications, there is a trade-off between accuracy, GPU memory, and running time and we need to balance the choices due to different requirements.

Another strategy for reducing computational complexity



(a) Accuracy (b) GPU memory (c) Running time

Figure 5. Computational complexity with different batch sizes at test time adaptation. From left to right: accuracy, GPU memory and running time on four domains of PACS [32] dataset.

is that we only update affine (*i.e.*, scale and shift) parameters when we adapt the model, like [66]. We conducted the experiments of updating affine parameters in batch normalization layers, and the result is 89% accuracy on the PACS dataset. Compared with 89.4% which is achieved by updating all parameters, the accuracy only decreases by 0.4% and the GPU memory cost drops by 2 GB. This shows that, with our method, only adapting affine parameters in batch normalization layers can achieve competitive performance with a salience GPU memory decrease.

4.5. Scalability to Medical Image Segmentation

Furthermore, we evaluated our algorithm on several medical image segmentation datasets. We conduct the experiments on three tasks: 1) prostate segmentation. We choose Promise12 [37] dataset, including 50 cases as source data and the MSD05 dataset (32 cases) from Medical Segmentation Decathlon [1] as test data. 2) cardiac structure segmentation. We choose ACDC [4] dataset (200 cases) as source data and LGE images (45 cases) from Multi-sequence Cardiac MR Segmentation Challenge [78, 79] as test data. The task is to segment the left ventricular blood pool, right ventricular blood pool, and myocardium. 3) optic disc and cup segmentation. We choose the training set of REFUGE challenge (400 images) [47] as the source domain, and RIM-ONE-r3 (159 images) [13] and Drishti-GS (101 images) [55] as two different target domains. We aim to segment the optic disc and optic cup. For RIM-ONE-r3 and Drishti-GS, we split them into 99/60 and 51/50 for training and test. For other datasets, we randomly split them to 80% for training and 20% for the test.

Implementation. We adopt DeepLabv3+ [9] with MobileNetV2 [52] backbone as segmentation models. In training and test phases, we use Adam [26] optimizer with fixed learning rate as $1e^{-4}$ without specific choice. We enlarge K (*cf*. Eq. 6) from 3 to 64 in the segmentation task. We use Dice Score [42] as the evaluation metric and we reported the average results of all the target structures in Table 7.

Results. We utilize ERM [63] and Tent [66] as the comparison methods. Table 7 shows that our method generally improves the baseline on different tasks. Specifically, we improve ERM baseline 3.9%, 4.44%, 3.57%, and 12.72% for the four tasks. Experiments prove that our method can also work well in image segmentation tasks.

Table 7. Results on medical image segmentation. We highlight the **best** result for each column.

Source	MSD05	ACDC	REFUGE	
Target	Promise12	LGE	DrishtiGS	RIM-ONE-r3
ERM [63]	83.00	82.34	81.05	68.17
Tent [66]	85.64	81.11	83.96	77.87
Ours	**86.90**	**86.78**	**84.62**	**80.89**

5. Limitations

Our work has some limitations. First, even though we have already validated our method on several classification and segmentation tasks, we have not extended our method to low-level tasks, such as image dehazing, denoising, and super-resolution tasks. As for low-level tasks, there is no conception of semantic prototype or entropy which can lead to the failure of our method. The second limitation is *hyperparameter search*. In this paper, we do hyperparameter search on the training domain validation set. However, the best choice on the training domains does not mean the best performance on the test domain [50]. We may need to try more strategies to conduct experiments in the TTA setting like test domain validation.

6. Conclusion

In this work, we focus on the test time adaptation and propose a new perspective that test time adaptation could be viewed as a feature revision problem. Furthermore, feature revision includes two parts: test time feature uniformity and test time feature alignment. As for the feature uniformity perspective, we propose *Test Time Self-Distillation* to make the target feature as uniform as possible in the adaption. To align features from the same class, we propose *Memorized Spatial Local Clustering* to encourage that the distance between feature representations in the latent space should align with the pseudo logits. A mass of experiments proves that our method not only generally improves the ERM baseline but also outperforms existing TTA or SFDA methods on four domain generalization benchmarks. Furthermore, our method could be adopted to different backbones. To broaden our model for real world applications, we then validate our method on four cross-domain medical image segmentation tasks. Experiment results show that our method is effect, flexible and scalable.

Acknowledgment

This work was supported by the National Natural Science Foundation of China, 81971604 and the Grant from the Tsinghua Precision Medicine Foundation, 10001020104.

References

[1] Michela Antonelli, Annika Reinke, Spyridon Bakas, Keyvan Farahani, et al. The medical segmentation decathlon. *Nature Communications*, 13(1), July 2022. 2, 8

[2] Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. Metareg: Towards domain generalization using meta-regularization. In *NeurIPS*, 2018. 3

[3] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, June 2003. 4

[4] Olivier Bernard, Alain Lalande, Clement Zotti, et al. Deep learning techniques for automatic MRI cardiac multi-structures segmentation and diagnosis: Is the problem solved? *IEEE Transactions on Medical Imaging*, 37(11):2514–2525, Nov. 2018. 2, 8

[5] Malik Boudiaf, Romain Müller, Ismail Ben Ayed, and Luca Bertinetto. Parameter-free online test-time adaptation. In *CVPR*, 2022. 1, 2, 3, 5, 12, 13

[6] Fabio Maria Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain generalization by solving jigsaw puzzles. In *CVPR*, 2019. 3

[7] Junbum Cha, Sanghyuk Chun, Kyungjae Lee, Han-Cheol Cho, Seunghyun Park, Yunsung Lee, and Sungrae Park. SWAD: domain generalization by seeking flat minima. In *NeurIPS*, 2021. 6, 12

[8] Dian Chen, Dequan Wang, Trevor Darrell, and Sayna Ebrahimi. Contrastive test-time adaptation. In *CVPR*, 2022. 1, 3, 12

[9] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018. 8

[10] Xu Chu, Yujie Jin, Wenwu Zhu, Yasha Wang, Xin Wang, Shanghang Zhang, and Hong Mei. DNA: domain generalization with diversified neural averaging. In *ICML*, 2022. 6

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 5, 7

[12] Qi Dou, Daniel Coelho de Castro, Konstantinos Kamnitsas, and Ben Glocker. Domain generalization via model-agnostic learning of semantic features. In *NeurIPS*, 2019. 3

[13] F. Fumero, S. Alayon, J. L. Sanchez, J. Sigut, and M. Gonzalez-Hernandez. RIM-ONE: An open retinal image database for optic nerve evaluation. In *2011 24th International Symposium on Computer-Based Medical Systems (CBMS)*. IEEE, June 2011. 2, 8

[14] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35, 2016. 3

[15] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018. 1, 3

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

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 5, 7, 12

[18] Dan Hendrycks and Thomas G. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*, 2019. 12

[19] Dan Hendrycks, Norman Mu, Ekin Dogus Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. In *ICLR*, 2020. 3

[20] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 4

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 3, 5

[22] Yusuke Iwasawa and Yutaka Matsuo. Test-time classifier adjustment module for model-agnostic domain generalization. In *NeurIPS*, 2021. 1, 2, 3, 5, 12, 13

[23] Minguk Jang, Sae-Young Chung, and Hye Won Chung. Test-time adaptation via self-training with nearest neighbor information. In *ICLR*, 2023. 12

[24] Daehee Kim, Youngjun Yoo, Seunghyun Park, Jinkyu Kim, and Jaekoo Lee. Selfreg: Self-supervised contrastive regularization for domain generalization. In *ICCV*, 2021. 3

[25] Kyungyul Kim, Byeongmoon Ji, Doyoung Yoon, and Sangheum Hwang. Self-knowledge distillation with progressive refinement of targets. In *ICCV*, 2021. 2

[26] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 5, 8

[27] Takeshi Kojima, Yutaka Matsuo, and Yusuke Iwasawa. Robustifying vision transformer without retraining from scratch by test-time class-conditional feature alignment. In *IJCAI*, 2022. 1, 3

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012. 1

[29] Jogendra Nath Kundu, Akshay R. Kulkarni, Suvaansh Bhambri, Deepesh Mehta, Shreyas Anand Kulkarni, Varun Jampani, and Venkatesh Babu Radhakrishnan. Balancing discriminability and transferability for source-free domain adaptation. In *ICML*, 2022. 6

[30] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, 2013. 5, 12, 13

[31] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M. Hospedales. Learning to generalize: Meta-learning for domain generalization. In *AAAI*, 2018. 3

[32] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *ICCV*, 2017. 2, 5, 8, 13

[33] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C. Kot. Domain generalization with adversarial feature learning. In *CVPR*, 2018. 3

[34] Xiaotong Li, Yongxing Dai, Yixiao Ge, Jun Liu, Ying Shan, and Lingyu Duan. Uncertainty modeling for out-of-distribution generalization. In *ICLR*, 2022. 3

[35] Yanghao Li, Naiyan Wang, Jianping Shi, Xiaodi Hou, and Jiaying Liu. Adaptive batch normalization for practical domain adaptation. *Pattern Recognition*, 80:109–117, Aug. 2018. 2, 3

[36] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *ICML*, 2020. 2, 3, 5, 12, 13

[37] Geert Litjens, Robert Toth, Wendy van de Ven, Caroline Hoeks, Sjoerd Kerkstra, Bram van Ginneken, Graham Vincent, Gwenael Guillard, Neil Birbeck, Jindang Zhang, et al. Evaluation of prostate segmentation algorithms for mri: the promise12 challenge. *Medical image analysis*, 18(2):359–373, 2014. 2, 8

[38] Yuejiang Liu, Parth Kothari, Bastien van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. TTT++: when does self-supervised test-time training fail or thrive? In *NeurIPS*, 2021. 1, 3

[39] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1

[40] Fangrui Lv, Jian Liang, Shuang Li, Bin Zang, Chi Harold Liu, Ziteng Wang, and Di Liu. Causality inspired representation learning for domain generalization. In *CVPR*, 2022. 3

[41] Divyat Mahajan, Shruti Tople, and Amit Sharma. Domain generalization using causal matching. In *ICML*, 2021. 3

[42] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *3DV*, 2016. 8

[43] Saeid Motiian, Marco Piccirilli, Donald A. Adjeroh, and Gianfranco Doretto. Unified deep supervised domain adaptation and generalization. In *ICCV*, 2017. 3

[44] Zachary Nado, Shreyas Padhy, D Sculley, Alexander D'Amour, Balaji Lakshminarayanan, and Jasper Snoek. Evaluating prediction-time batch normalization for robustness under covariate shift. *arXiv preprint arXiv:2006.10963*, 2020. 2, 3

[45] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *ICML*, 2022. 1, 2, 3, 5, 12, 13

[46] Oren Nuriel, Sagie Benaim, and Lior Wolf. Permuted adain: Reducing the bias towards global statistics in image classification. In *CVPR*, 2021. 3

[47] José Ignacio Orlando, Huazhu Fu, João Barbosa Breda, et al. REFUGE challenge: A unified framework for evaluating automated methods for glaucoma assessment from fundus photographs. *Medical Image Analysis*, 59:101570, Jan. 2020. 2, 8

[48] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 5

[49] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *ICCV*, 2019. 2, 5, 13

[50] Alexandre Rame, Corentin Dancette, and Matthieu Cord. Fishr: Invariant gradient variances for out-of-distribution generalization. In *ICML*, 2022. 8

[51] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, Apr. 2015. 5

[52] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018. 8

[53] Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. In *NeurIPS*, 2020. 2, 3, 5, 12, 13

[54] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948. 3

[55] Jayanthi Sivaswamy, S Krishnadas, Arunava Chakravarty, G Joshi, A Syed Tabish, et al. A comprehensive retinal image dataset for the assessment of glaucoma from the optic nerve head analysis. *JSM Biomedical Imaging Data Papers*, 2(1):1004, 2015. 2, 8

[56] Baochen Sun and Kate Saenko. Deep CORAL: correlation alignment for deep domain adaptation. In *ECCV*, 2016. 3

[57] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *ICML*, 2020. 1, 3

[58] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019. 5, 7

[59] Ilya O. Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. Mlp-mixer: An all-mlp architecture for vision. In *NeurIPS*, 2021. 5, 7

[60] Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *CVPR*, 2011. 2, 5, 13

[61] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014. 3

[62] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. 7

[63] Vladimir Vapnik. *Statistical learning theory*. Wiley, 1998. 5, 6, 7, 8, 12, 13

[64] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *CVPR*, 2017. 2, 5, 13

[65] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *ICML*, 2019. 3

[66] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *ICLR*, 2021. 1, 2, 3, 5, 8, 12, 13

[67] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *CVPR*, 2022. 1, 3

[68] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *ICML*, 2020. 1

[69] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 5, 7

[70] Xufeng Yao, Yang Bai, Xinyun Zhang, Yuechen Zhang, Qi Sun, Ran Chen, Ruiyu Li, and Bei Yu. PCL: proxy-based contrastive learning for domain generalization. In *CVPR*, 2022. 3, 6

[71] Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Seong Joon Oh, Youngjoon Yoo, and Junsuk Choe. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, 2019. 3

[72] Daoan Zhang, Mingkai Chen, Chenming Li, Lingyun Huang, and Jianguo Zhang. Aggregation of disentanglement: Reconsidering domain variations in domain generalization. *arXiv preprint arXiv:2302.02350*, 2023. 3

[73] Daoan Zhang, Chenming Li, Haoquan Li, Wenjian Huang, Lingyun Huang, and Jianguo Zhang. Rethinking alignment and uniformity in unsupervised image semantic segmentation. *arXiv preprint arXiv:2211.14513*, 2022. 1

[74] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. 3

[75] Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *ICCV*, 2019. 2

[76] Marvin Mengxin Zhang, Sergey Levine, and Chelsea Finn. MEMO: Test time robustness via adaptation and augmentation. In *Advances in Neural Information Processing Systems*, 2022. 1, 2

[77] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain generalization with mixstyle. In *ICLR*, 2021. 3

[78] Xiahai Zhuang. Multivariate mixture model for cardiac segmentation from multi-sequence MRI. In *MICCAI*. 2016. 2, 8

[79] Xiahai Zhuang. Multivariate mixture model for myocardial segmentation combining multi-source images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(12):2933–2946, Dec. 2019. 2, 8

This supplementary material provides more details about our implementation (Sec. A), additional experiments on three corruption datasets (Sec. B), some discussion (Sec. C), running time analysis (Sec. D) and detailed results (Sec. E).

A. Other Implement Details

We run our experiments mainly on a single RTX-A6000 or RTX-2080Ti GPU, depending on the need for GPU memory. For source training, we set the batch size as 32 for each source domain and the learning rate as $5e^{-5}$. We set dropout probability and weight decay to zero. We train source model 5k iterations except for DomainNet. We tripled it from 5k to 15k following [7]. All images are resized to 224×224 and data augmentation is used in source domain training, which includes randomly cropping, flipping horizontally, jittering colour, and changing the intensity.

For implementations of different test time adaptation methods, we use publicly released code of T3A [22][3], except for LAME [4] [5] and ETA[5] [45] we use source code of authors. For PL [30], we set confidence as 0.9. We resize all images to 224×224 and no data augmentation is used during the test time adaptation process.

For different backbones, we use `torchvision` implementation [6] except for ViT-B/16 and MLP-mixer, we use implementation from `timm` library[7]. For all experiments, we use three random seeds {0,1,2} and report the average results in the main text.

B. Experiments on Corruption Benchmark

We conduct experiments on three corruption datasets [18], including CIFAR-10/100-C and ImageNet-C. The results are listed in Table 9. From the Table 9, it is noticed that our method still achieves the state-of-the-art performance on three corruption datasets.

C. Discussion

Performance on VLCS. All compared methods shown poor performance on VLCS dataset. We notice that the label distribution is severely different among domains in VLCS compared to other datasets (*e.g.* PACS and OfficeHome), which is probably why fewer methods show performance gain in VLCS compared to other datasets. We calculated the label distribution of VLCS dataset, as shown in Table 8. Since the label distribution is not considered for existing methods, the adaptation may fail.

Related Work. There are some papers considering nearest neighbor information in the test time adaptation setting. TAST [23], a concurrent work published in ICLR 2023, considers nearest neighbor information to refine pseudo-labels. AdaContrast [8] uses a soft voting strategy among the nearest neighbors in the feature space to refine pseudo-labels. Both of them use nearest neighbor information to refine pseudo-labels. Different from them, our *Memorized Spatial Local Clustering* aims to cluster features with the same pseudo-label.

[3] https://github.com/matsuolab/T3A
[4] https://github.com/fiveai/LAME
[5] https://github.com/mr-eggplant/EATA
[6] https://github.com/pytorch/vision
[7] https://github.com/rwightman/pytorch-image-models

Table 8. Label distribution on VLCS dataset.

	0	1	2	3	4
Caltech101	237	123	118	67	870
LabelMe	80	1209	88	42	1237
SUN09	20	932	1036	30	1264
VOC2007	330	699	428	420	1499

Table 9. Accuracy on CIFAR-10/100-C and ImageNet-C. We use ResNet26 for CIFAR-10/100-C, and ResNet50 for ImageNet-C.

	CIFAR-10-C	CIFAR-100-C	ImageNet-C
ERM [63]	70.7	41.4	18.0
BN [53]	77.4	48.0	33.5
Tent [66]	80.7	51.7	42.7
PL [30]	80.2	49.8	38.4
SHOT-IM [36]	80.7	52.1	43.1
T3A [22]	77.4	44.6	36.5
ETA [45]	80.6	<u>52.4</u>	**48.1**
LAME [5]	79.4	50.6	47.6
Ours	**81.7**	**52.6**	<u>48.0</u>

D. Running Time Analysis

We provide the running time of different methods for reference. Running time is tested using RTX-A6000 GPU and AMD-EPYC-7542 32 Core Processor. The results are listed in Table 10 using ResNet50 [17] backbone.

E. Full Results

We provide full results of different methods on ResNet-18/50, including results for each domain and error bars across three random seeds. See Table 11-18.

Table 10. Running time analysis of different methods on four datasets. The units used in the table are seconds. "GD" denotes whether the method requires gradient-based optimization.

	GD	PACS [32]	VLCS [60]	OfficeHome [64]	DomainNet [49]
ERM [63]	✗	17	17	56	736
BN [53]	✗	18	18	56	745
Tent [66]	✓	32	33	67	1094
PL [30]	✓	33	34	69	1339
SHOT-IM [36]	✓	37	39	72	1339
T3A [22]	✗	19	19	57	829
ETA [45]	✓	21	22	66	1094
LAME [5]	✗	18	19	56	750
Ours	✓	35	37	71	1644

Table 11. Full results on PACS with ResNet18.

	A	C	P	S	Avg
ERM	80.56±0.45	77.36±0.85	93.01±0.17	77.35±2.90	82.07±0.49
BN	81.60±0.16	82.00±0.51	92.85±0.24	74.86±1.10	82.82±0.34
Tent	83.43±0.53	83.02±0.74	93.88±0.38	79.35±1.15	84.92±0.32
PL	84.93±1.32	83.27±1.78	92.62±1.37	77.72±3.66	84.64±1.13
SHOT-IM	84.61±1.06	82.36±1.88	93.60±0.38	69.64±3.40	82.55±1.07
T3A	83.00±0.76	79.56±0.44	94.48±0.34	76.95±2.94	83.50±0.67
ETA	81.33±0.30	81.89±0.55	92.82±0.53	74.77±0.91	82.70±0.31
LAME	83.05±0.53	83.06±0.51	94.30±0.29	77.91±0.80	84.58±0.23
Ours	86.50±0.75	86.38±0.82	94.57±0.32	81.84±0.94	87.32±0.39

Table 12. Full results on OfficeHome with ResNet18.

	A	C	P	R	Avg
ERM	55.49±0.61	51.41±0.46	71.92±0.47	73.67±0.18	63.12±0.26
BN	54.36±1.00	51.14±0.25	71.20±0.44	72.52±0.37	62.30±0.25
Tent	55.85±0.91	53.38±0.29	72.50±0.65	73.26±0.37	63.75±0.23
PL	54.64±0.96	48.36±1.58	68.83±1.08	69.05±0.58	60.22±0.37
SHOT-IM	55.45±1.08	52.32±0.99	73.23±0.76	72.67±0.37	63.42±0.52
T3A	56.18±0.48	52.90±0.67	73.44±0.48	74.48±0.19	64.25±0.22
ETA	54.88±0.33	51.05±0.21	71.18±0.44	72.72±0.30	62.46±0.14
LAME	54.84±0.86	50.90±0.26	70.85±0.30	72.19±0.30	62.20±0.21
Ours	57.87±0.81	53.40±0.27	74.20±0.38	73.86±0.22	64.83±0.46

Table 13. Full results on VLCS with ResNet18.

	C	L	S	V	Avg
ERM	92.44±0.78	62.72±0.69	69.26±2.01	66.58±1.43	72.75±0.29
BN	76.23±1.99	57.84±0.80	58.04±0.73	65.12±0.35	64.31±0.47
Tent	82.65±1.34	60.02±1.04	60.49±0.84	66.28±0.46	67.36±0.43
PL	86.64±3.45	61.66±1.47	61.78±2.66	65.64±1.53	68.93±1.07
SHOT-IM	76.65±2.94	57.34±1.26	59.13±0.85	66.46±0.55	64.90±0.70
T3A	96.76±1.22	63.80±0.39	64.98±0.45	66.55±0.38	73.03±0.35
ETA	76.1±1.90	57.89±0.87	58.12±0.73	65.31±0.23	64.35±0.40
LAME	94.7±0.21	62.69±0.25	67.58±0.12	66.55±0.45	72.88±0.13
Ours	97.2±1.15	64.50±0.28	65.42±0.34	67.32±0.38	73.61± 0.42

Table 14. Full results on DomainNet with ResNet18.

	clipart	infograph	painting	quickdraw	real	sketch	Avg
ERM	55.86±0.15	16.85±0.06	44.80±0.20	12.49±0.38	56.74±0.04	46.96±0.12	38.95±0.08
BN	55.90±0.09	12.07±0.15	43.58±0.06	11.63±0.15	56.47±0.09	47.16±0.15	37.80±0.06
Tent	56.67±0.14	13.58±0.19	45.02±0.13	11.52±0.33	57.25±0.05	48.59±0.16	38.77±0.10
PL	55.99±0.12	14.44±0.27	44.29±0.52	4.34±0.46	45.22±1.31	47.09±0.13	35.23±0.32
SHOT-IM	56.73±0.18	14.02±0.22	44.61±0.06	16.13±0.24	57.51±0.13	48.20±0.17	39.53±0.09
T3A	55.82±0.18	16.71±0.20	43.43±0.18	17.86±0.26	57.58±0.06	46.28±0.08	39.61±0.04
ETA	56.46±0.11	14.67±0.12	45.20±0.10	14.13±0.18	57.70±0.05	48.44±0.14	39.43±0.04
LAME	55.42±0.09	12.09±0.16	43.35±0.08	11.52±0.16	55.69±0.09	46.86±0.17	37.49±0.07
Ours	56.79±0.12	18.42±0.14	46.71±0.12	13.45±0.22	57.65±0.12	48.12±0.24	40.19±0.08

Table 15. Full results on PACS with ResNet50.

	A	C	P	S	Avg
ERM	82.50±1.83	80.80±0.33	94.05±0.30	80.99±1.29	84.59±0.40
BN	83.27±0.47	84.91±0.43	94.03±0.31	77.92±1.23	85.03±0.20
Tent	85.28±1.07	86.75±0.92	94.94±0.83	82.96±1.20	87.48±0.52
PL	83.96±1.63	84.15±2.91	93.82±1.74	78.99±2.64	85.23±1.70
SHOT-IM	84.31±0.63	85.74±0.56	94.04±0.67	77.91±0.94	85.50±0.31
T3A	84.07±0.68	82.37±0.92	95.02±0.27	82.72±1.06	86.04±0.24
ETA	83.27±0.47	84.91±0.43	94.03±0.31	77.92±1.24	85.04±0.20
LAME	84.97±0.77	85.50±0.55	95.04±0.23	80.97±1.09	86.62±0.22
Ours	87.68±0.84	88.78±0.63	96.17±0.37	85.01±1.52	89.41±0.51

Table 16. Full results on OfficeHome with ResNet50.

	A	C	P	R	Avg
ERM	60.71±0.88	55.74±0.79	76.18±0.65	76.83±0.41	67.37±0.06
BN	58.23±0.76	55.62±0.68	75.08±0.60	75.47±0.35	66.10±0.20
Tent	60.55±0.93	58.73±0.85	76.48±0.53	76.07±0.59	67.96±0.24
PL	59.14±0.93	57.29±0.62	76.24±0.69	75.85±0.28	67.13±0.17
SHOT-IM	59.20±0.76	57.54±0.58	76.53±0.44	76.27±0.37	67.39±0.16
T3A	61.23±1.00	56.69±1.11	77.95±0.43	77.31±0.22	68.29±0.21
ETA	58.38±0.80	55.78±0.69	75.17±0.56	75.53±0.34	66.21±0.19
LAME	58.67±0.70	55.58±0.53	75.09±0.65	75.40±0.31	66.19±0.20
Ours	62.32±0.52	57.45±0.71	77.48±0.45	77.45±0.38	68.67±0.14

Table 17. Full results on VLCS with ResNet50.

	C	L	S	V	Avg
ERM	94.91±0.32	65.20±2.35	66.52±1.65	69.41±3.38	74.01±1.32
BN	75.26±1.15	56.85±0.53	60.87±0.68	66.16±0.59	64.78±0.34
Tent	84.75±2.25	60.70±1.26	64.94±1.40	66.42±0.89	69.20±0.83
PL	86.75±4.25	61.19±2.22	63.36±3.33	62.77±2.84	68.52±1.79
SHOT-IM	76.54±1.27	55.90±1.20	61.26±0.71	67.24±0.86	65.23±0.32
T3A	97.06±0.30	63.96±0.89	67.14±0.52	67.75±0.31	73.98±0.32
ETA	75.27±1.13	56.85±0.52	60.89±0.67	66.16±0.58	64.79±0.34
LAME	96.25±0.55	61.39±0.40	70.25±0.67	67.89±0.38	73.94±0.14
Ours	97.40±0.41	64.89±0.64	68.05±0.45	68.12±0.43	74.52±0.27

Table 18. Full results on DomainNet with ResNet50.

	clipart	infograph	painting	quickdraw	real	sketch	Avg
ERM	64.76±0.06	22.11±0.11	51.77±0.18	13.84±0.14	64.66±0.21	54.04±0.27	45.20±0.09
BN	64.46±0.09	15.62±0.05	50.64±0.08	11.84±0.05	63.86±0.11	53.86±0.19	43.38±0.03
Tent	65.78±0.08	18.18±0.02	52.96±0.01	10.77±0.11	64.85±0.09	55.71±0.09	44.71±0.03
PL	64.96±0.05	19.00±0.03	50.30±0.27	4.21±0.68	54.40±0.55	54.20±0.15	41.18±0.13
SHOT-IM	65.62±0.05	18.73±0.21	52.41±0.08	19.01±0.24	66.47±0.12	55.54±0.08	46.30±0.07
T3A	64.77±0.05	22.10±0.09	50.89±0.15	19.41±0.18	65.85±0.06	53.96±0.17	46.16±0.03
ETA	65.11±0.09	19.37±0.19	52.69±0.11	18.24±0.33	65.92±0.10	55.48±0.16	46.13±0.08
LAME	64.18±0.12	15.64±0.07	50.54±0.04	11.77±0.05	63.46±0.08	53.65±0.18	43.20±0.03
Ours	66.12±0.08	24.12±0.12	52.82±0.10	18.17±0.08	68.45±0.12	56.72±0.12	47.73±0.05