Domain-Specific Block Selection and Paired-View Pseudo-Labeling for Online Test-Time Adaptation

Yeonguk Yu, Sungho Shin, Seunghyeok Back, Minhwan Ko, Sangjun Noh, and Kyoobin Lee
Gwangju Institute of Science and Technology
{yeon_guk, hogili89, shback, mhko1998, sangjun7}@gm.gist.ac.kr, kyoobinlee@gist.ac.kr

Abstract

Test-time adaptation (TTA) aims to adapt a pre-trained model to a new test domain without access to source data after deployment. Existing approaches typically rely on self-training with pseudo-labels since ground-truth cannot be obtained from test data. Although the quality of pseudo labels is important for stable and accurate long-term adaptation, it has not been previously addressed. In this work, we propose DPLOT, a simple yet effective TTA framework that consists of two components: (1) domain-specific block selection and (2) pseudo-label generation using paired-view images. Specifically, we select blocks that involve domain-specific feature extraction and train these blocks by entropy minimization. After blocks are adjusted for current test domain, we generate pseudo-labels by averaging given test images and corresponding flipped counterparts. By simply using flip augmentation, we prevent a decrease in the quality of the pseudo-labels, which can be caused by the domain gap resulting from strong augmentation. Our experimental results demonstrate that DPLOT outperforms previous TTA methods in CIFAR10-C, CIFAR100-C, and ImageNet-C benchmarks, reducing error by up to 5.4%, 9.1%, and 2.9%, respectively. Also, we provide an extensive analysis to demonstrate effectiveness of our framework. Code is available at https://github.com/gist-ailab/domain-specific-block-selection-and-paired-view-pseudo-labeling-for-online-TTA.

1. Introduction

Deep neural networks achieve remarkable performance when the training and target data originate from the same domain [39, 43]. In contrast, deployed models perform poorly if domain shifts exists between source training data and target test data [14, 38]. For example, a pre-trained image classification model may suffer this phenomenon for the given corrupted images due to sensor degradation, weather change, and other reasons [17, 41]. Various studies have addressed the domain shift issue [21, 31, 60]. Recently, test-time adaptation (TTA), which aims to improve the model performance on target domain without access to the source data during the inference stage, has received attention because of its practicality and applicability [35, 47, 48].

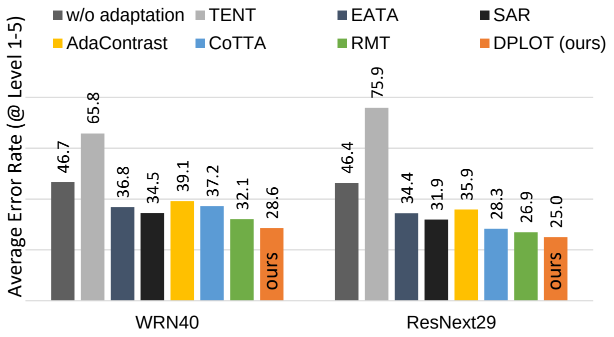

Figure 1. **Results of the proposed framework for online test-time adaptation (Orange).** We evaluate average error rates of the WideResNet40 and ResNext-29 architectures for the CIFAR100-C gradual setting benchmark using competitive test-time adaptation methods. In the gradual setting, the networks should adapt to continually changing corruption domains (135 changes in total).

In online TTA, the objective is to simultaneously make prediction and adaptation using a source pre-trained model for the given test data. Existing TTA methods typically rely on self-training with pseudo labels, such as entropy minimization [15, 46] and consistency regularization [44, 45]. Entropy minimization trains the model by self-generated pseudo-labels, whereas consistency regularization uses pseudo-labels generated by a teacher model. These methods have demonstrated excellent performance with short-term test sequence in the stationary environment [47].

Under the continually changing domain in non-stationary environments [32, 48], self-training with pseudo-labels can lead to error accumulation, gradually degrading the quality of pseudo-labels [48]. To alleviate the issue, stochastic restoration (i.e., reset the model to the source pre-trained weight) and augmentation-averaged pseudo-label (i.e., various hard augmentations including color jitter) was proposed by [48]. In addition, Döbler *et al.* [10] combined symmetric cross-entropy [50] and contrastive loss for stabilizing the pseudo-labeling without restoration. However, previous methods lacks an accurate approach to generate pseudo-labels from the teacher, a critical aspect for stable long-term test-time adaptation.

In this study, we address accurate pseudo-label generation in underlying assumption that *images from source domain and target domain share domain-invariant features for a task, regardless of shifted domain-specific features caused by corruptions*. We consider the domain-invariant feature to be a high-level feature useful for the task and the domain-specific feature to be a low-level feature with no concern for the task as termed in [51–53]. To achieve precise adaptation under this assumption, we propose **D**omain-specific block selection and paired-view **P**seudo-**L**abeling for **O**nline **T**est-Time adaptation (DPLOT), which is composed of two core components: (1) domain-specific block selection and (2) pseudo-label generation using paired-view images. The block selection method identifies blocks involve in domain-specific feature extraction (termed domain-specific block) using augmented source training data. These domain-specific blocks are then fine-tuned through entropy minimization during the test-time phase. Then, we generate the pseudo-label from the teacher by averaging predictions for the given test image and its horizontally flipped counterpart to update all model parameters. This is motivated by the fact that the teacher model's domain-specific blocks are adjusted for the current domain, and hard augmentation may generate another domain gap, which may lead to degraded pseudo-labels. Consequently, our framework provides strong adaptation performance for the model as shown in Figure 1.

In summary, the main contributions are as follows:

- We propose the DPLOT, which consists of domain-specific block selection and paired-view pseudo labeling for long-term online test-time adaptation.
- We compare the proposed method with other online TTA methods in both *continual* and *gradual* settings benchmarks, and our framework outperforms other methods.
- We provide a wide range of analyses that lead to an improved understanding of our framework.

2. Related Work

Unsupervised Domain Adaptation Unsupervised domain adaptation (UDA) aims to adapt a model to a target domain using given labeled source data and unlabeled target data before model deployment.For example, Ganin and Lempitsky [13] proposed gradient reversal layer to force the feature extractor to produce same feature distribution for the given source data and target data . Also, Hoffman *et al.* [20] used image-to-image translation to create labeled target-like source data to train the network. French *et al.* [11] used self-ensembling with the mean teacher to minimize the difference between the student's prediction for the augmented test data and the teacher's prediction for the test data. Kundu *et al.* [27] proposed class-incremental method without using source training data. Recently, Hoyer *et al.* [21] proposed the adaptation framework based on masked image consistency, where the model is forced to produce the same prediction for the given target data and corresponding masked data. Also, Prasanna *et al.* [37] proposed the continual domain adaptation method based on pruning-aided weight modulation to reduce catastrophic forgetting. Since both UDA and TTA aim to performance improvement in target domain, the methods from UDA can be considered. Our method also uses the mean teacher as in [21] to reduce the prediction difference for adapting the network to the target domain but without using source data.

Test-Time Adaptation Test-time adaptation (TTA), which only requires target data for adaptation unlike UDA, has gained increasing attention. As in [29], (online) TTA can be categorized into batch normalization (BN) calibration [33, 58], entropy minimization [5, 35, 36, 47], and consistency regularization [10, 48, 49]. Specifically, Mizra *et al.* [33] proposed dynamic unsupervised adaptation (DUA), which adapts the statistics of the BN layers[22] to remove degradation caused by BN[12, 28]. On the other hand, updating affine parameters of BN layers has demonstrated improved adaptation performance [5, 35, 36, 47]. Niu *et al.* [36] demonstrated that using batch-independent normalization methods like layer norm [1] and group norm [54] instead of using batch-dependent BN is helpful for stable entropy minimization. Also, they proposed sharpness-aware entropy minimization, which filters out unreliable samples by their gradients, based on the observation that large gradient samples lead to model collapse. Similar to our work, Choi *et al.* [5] proposed the framework that updates model parameters by entropy minimization, differently depending on their sensitivity of distribution shift. Simultaneously, consistency regularization-based TTA frameworks have been investigated. Wang *et al.* [48] proposed TTA in a continually changing domain by using augmentation-averaged pseudo-target from the mean teacher. Also, Chen *et al.* [3] proposed to use contrastive learning in TTA for learning better representation in the target domain. Moreover, robust mean teacher (RMT) is proposed [10], which showed state-of-the-art performance by using various techniques such as symmetric cross-entropy and source replay. Previous entropy minimization methods [35, 36, 47] update BN layer of the model which can modify domain-invariant feature extraction. In this work, we propose block selection method to prevent the modification of domain-invariant feature extraction by updating blocks that involve in domain-specific feature extraction. Also, we propose to generate pseudo-labels from the teacher only using simple flip augmentation to improve the quality of labels. Consequently, our DPLOT uses both entropy minimization and consistency regularization for reliable long-term adaptation.

3. Method

We introduces DPLOT, a simple yet effective online TTA framework. First, we describe the overview of our framework in Section 3.1. Second, we describe the block selection method before deployment in Section 3.2. Lastly, we introduce how our framework adapts the model for the given unlabeled test data after deployment in Section 3.3.

3.1. Overview of DPLOT's components

The components within our framework can be categorized based on whether they are used before or after deployment. Domain-specific block selection is conducted before deployment, while our adaptation method as shown in Figure 2, involving entropy minimization on the selected blocks and the use of paired-view pseudo-labels for updating all model parameters, is performed after deployment.

The domain-specific block selection is proposed under the assumption that the domain-invariant feature of test image remains consistent with the source domain, while the domain-specific feature changes. Therefore, the domain-specific feature extraction of the model should be adjusted to bridge the gap between the source domain and target domain caused by different feature statistics [2, 28, 51]. We evaluate each block by measuring cosine similarity between prototype features before and after the entropy minimization using Gaussian noise-added source data. Subsequently, we select the blocks that maintain high similarity, indicating that the blocks do not involve in domain-invariant feature extraction (i.e., involves in domain-specific feature extraction). During test-time, the selected blocks are updated by minimizing entropy for the given test data to adjust the domain-specific feature extraction to the current corruption. By selecting domain-specific blocks, we can adjust domain-specific feature extraction without disrupting domain-invariant feature extraction unlike previous method [47], where all batch normalization [22] layers are updated.

We provide pseudo-labels generated by the exponential moving average (EMA) teacher [45] (i.e., $\theta'_{t+1} = \alpha\theta'_t + (1-\alpha)\theta_{t+1}$) to the model for further adjusting all parameters on the target domain. This is based on insight that the high-level feature in the test image can be affected by domain shifts, and all parameters should be adjusted to increase adaptation performance. In contrast to previous approach [48], where hard augmentations (e.g., color jitter, Gaussian noise, blur, and random pad-crop) are used for

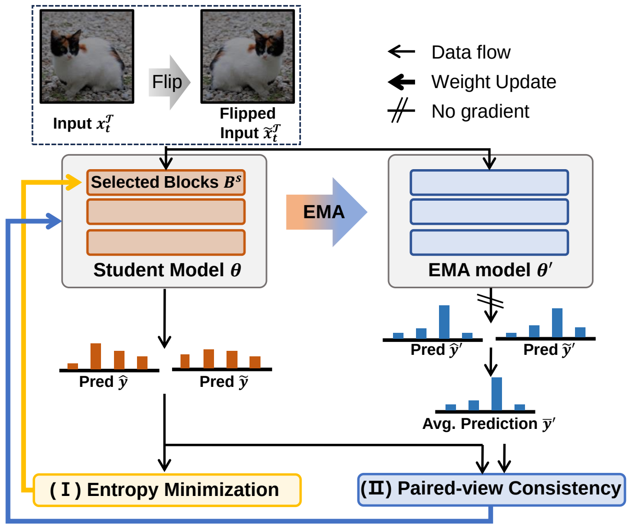

Figure 2. Illustration of our proposed test-time adaptation using entropy minimization and paired-view consistency. During test-time, the current test and corresponding flipped images are given to the student model and EMA teacher model. Entropy minimization is performed to update the parameters of selected blocks (yellow arrow), while all parameters are updated to minimize the difference between student output and the averaged EMA model's prediction (blue arrow).

generating pseudo-labels, we only use horizontal flip augmentation. Since the teacher's domain-specific extraction is adjusted on the target domain by entropy minimization and even a small domain gap between the test and augmented images can reduce the teacher's accuracy, the horizontal flip is well suited for generating pseudo-labels from the teacher.

3.2. Block selection before deployment

We consider a pre-trained neural network θ, comprising a feature extractor and a classifier. We assume that the feature extractor of the network is composed of multiple L blocks (e.g., ResNet18 has 8 residual blocks [16]). For a given a RGB image $x \in \mathbb{R}^{H\times W\times 3}$, a feature vector $f \in \mathbb{R}^d$ of dimension d is extracted. Then, our objective is to select blocks that involve in domain-specific feature extraction. To this end, we first calculate the domain-invariant feature space using prototype vectors. The prototype vector $p_c \in \mathbb{R}^d$ is acquired by averaging all source feature vectors belonging to the class c as follows:

$$p_c = \frac{1}{N_c}\sum_{n=1}^{N_c} f(x_n^c \in \mathcal{X}^S; \theta), \tag{1}$$

where $f(\cdot)$, x_n^c, and N_c refer to the feature extraction, n-th RGB image belonging to the class c, and number of images for class c, respectively. Consequently, prototypes for all class, $P = \{p_1, p_2, ..., p_C\}$, are obtained.

```
Algorithm 1: Domain-specific block selection
Init.   : An empty list $\mathcal{S}$, an empty list $\mathcal{B}^s$;
Input   : Blocks of the model $\mathcal{B} = \{b_1, b_2, ..., b_L\}$,
          source data $(\mathcal{X}^S, \mathcal{Y}^S)$, threshold $\gamma$,
          pre-trained model $\theta$;
Calculate $P$ using Eq. 1
Add noise to $\mathcal{X}^S$
foreach block $b_i \in \mathcal{B}$ do
    Minimize entropy for parameter of $b_i$ using $\mathcal{X}^S$
    Calculate $P'$ using Eq. 1
    Calculate $s_i$ using Eq. 2
    Append $s_i$ in $\mathcal{S}$
    Reset $\theta$;
end
Min-max scale $S$
foreach $(s_i, b_i) \in (\mathcal{S}, \mathcal{B})$ do
    if $s_i > \gamma$ then
        Append $b_i$ in $\mathcal{B}^s$;
    end
end
Output: Selected block list $\mathcal{B}^s$
```

To further select the block for adjusting domain-specific feature extraction during test-time, we measure the similarity s between the original prototypes and modified prototypes after the block's parameters have been updated via entropy minimization with Gaussian noise (zero mean and 0.5 variance) added training images. We use Gaussian noise as it is common corruption and can represent various domain shifts [8, 30]. This similarity is calculated by averaging cosine similarity as follows:

$$s_i = \frac{1}{C}\sum_{c=1}^{C} \frac{p_c \cdot p'_c}{\|p_c\|\|p'_c\|}, \tag{2}$$

where p'_c denotes a single modified prototype vector for the c-th class after the entropy minimization. Also, C refers to the number of classes. After computing the similarity for every block, we apply min-max scaling to normalize the results into the range of [0, 1]. This scaling allows us to choose threshold more generally across various architectures. A high similarity indicates that the block is adapted on Gaussian noise added images without modifying domain-invariant feature space (i.e., high-level feature space). Finally, we select blocks that higher than the threshold γ. The pseudo code of the block selection is described in Algorithm 1. In our experiments, the threshold is set to 0.75, unless otherwise specified.

3.3. Test-time adaptation after deployment

Entropy minimization Entropy minimization is performed for the given target data at current time $x_t^{\mathcal{T}}$ to update

```
Algorithm 2: Adaptation process after deployment
Init.   : Selected Blocks $\mathcal{B}^s$, a pre-trained model $\theta$,
          a teacher model $\theta'$ initialized from $\theta$;
Input   : For each time step $t$, current batch of
          data $x_t^{\mathcal{T}}$;
1: Horizontally flip $x_t^{\mathcal{T}}$ and get pseudo-label from $\theta'_t$
   by Eq. 4 and Eq. 5
2: Update $\mathcal{B}^s$ of $\theta_t$ by entropy minimization loss $\mathcal{L}_e$
   in Eq. 3 using both $x_t^{\mathcal{T}}$ and $\tilde{x}_t^{\mathcal{T}}$
3: Update $\theta_t$ by paired-view consistency loss $\mathcal{L}_pc$ in
   Eq. 6 using both $x_t^{\mathcal{T}}$ and $\tilde{x}_t^{\mathcal{T}}$
4: Update $\theta'_t$ by moving average of $\theta_t$
Output: Ensemble prediction, Updated model $\theta_{t+1}$,
        Updated teacher $\theta'_{t+1}$
```

the selected blocks $\mathcal{B}^s$ of model θ. Following other entropy minimization-based methods [5, 35, 36, 47], we use Shannon Entropy [42] as follows:

$$\mathcal{L}_e = -\sum_c \hat{y}_c \log \hat{y}_c, \tag{3}$$

where $\hat{y}_c$ represents the output probability for the c-th class using the model θ. By minimizing entropy on test batch $x_t^{\mathcal{T}}$, the model is trained to push decision boundaries toward low-density region in prediction space [46]. As a result, the model is forced to acquire discriminative high-level features from current target domain by adjusting domain-specific feature extraction.

Paired-view consistency Domain-specific feature extraction of the model is adjusted to the current corruption using entropy minimization with selected domain-specific blocks. To leverage all parameters, the consistency regularization between the model θ and the moving average teacher θ', which makes the training stable [10, 45, 48], is used. Specifically, we use horizontal flip and moving average teacher [45] to generate pseudo-labels for the given test and corresponding flipped images as follows:

$$\hat{y}' = g_{\theta'}(x_t^{\mathcal{T}}), \tag{4}$$

$$\tilde{y}' = g_{\theta'}(\tilde{x}_t^{\mathcal{T}}), \tag{5}$$

where $g_{\theta'}(\cdot)$ refers to the teacher's prediction of the given input. Also, x and $\tilde{x}$ refer to the original input and horizontally flipped input. The pseudo-label $\bar{y}'$ is calculated by averaging $\hat{y}'$ and $\tilde{y}'$. Subsequently, all parameters of the model θ are updated by the symmetric cross-entropy [10, 50] between the model and teacher predictions:

$$\mathcal{L}_{pc} = \mathcal{L}_{sce}(\hat{y}, \bar{y}') + \mathcal{L}_{sce}(\tilde{y}, \bar{y}'), \tag{6}$$

$$\mathcal{L}_{sce}(a, b) = \frac{1}{2} \cdot (\mathcal{L}_{ce}(a, b) + \mathcal{L}_{ce}(b, a)), \tag{7}$$

where $\mathcal{L}_{sce}$ and $\tilde{y}$ refer to the symmetric cross-entropy and the model's prediction for the flipped input, respectively. Also, L_{ce} refers to the standard cross-entropy loss. We use the symmetric cross-entropy since it is known to be robust to noisy labels [10, 50]. After adaptation, we ensemble the predictions of both models by adding the model's and teacher's logits as [10] for better performance. Our method during test-time is summarized in Algorithm 2.

4. Experiments

Setup We evaluate our framework on CIFAR10-C, CIFAR100-C, and ImageNet-C, designed to benchmark the robustness of classification networks [17]. CIFAR dataset contains 10,000 and ImageNet dataset contains 50,000 test images for each of the 15 corruptions with 5 levels of severity. For the experiments, we use a network pre-trained on the clean training set of CIFAR [26] and ImageNet [9]. For CIFAR10, we use WRN28-10 [57], WRN40-2A [18], ResNet-18A [24]. For CIFAR100, we use WRN40-2A [18] and ResNext-29 [55]. For ImageNet, we use ResNet50 [16] and ResNet50A [19]. Architectures named with 'A' refers to the networks trained to be robust against corruption (e.g., AugMix [18]); specific details are described in supplementary materials.

We evaluate our method in two different settings. First, we consider the continual setting introduced by [48]. Unlike the basic TTA setting, where evaluation is conducted for each corruption individually, the model is adapted to a sequence of test domains in an online fashion under the largest corruption severity level 5 (total 15 shifts). Second, we consider the gradual setting, as introduced by [32] where the corruption severity level changes as follows: $1 \to 2 \to \cdots \to 5 \to \cdots \to 2 \to 1$ (total 135 shifts). This setting is motivated by the fact that domain shift does not occur abruptly but changes rather smoothly.

Also, we follow the implementation setting of RMT [10]. Specifically, the batch size during test-time is set to 200 and 64 for CIFAR and ImageNet-C, respectively. We use an Adam [23] optimizer with a learning rate of 1e-3 and 1e-4 for entropy minimization to domain-specific blocks and paired-view consistency to all blocks, respectively. Warm-up is conducted before deployment, as done in [10], and we use pre-trained weights provided by RobustBench [7].

Baselines We compare our method with various source-free TTA baselines. Also, BN-1 refers to method that recalculates the BN statistics using the test batch. TENT [47], EATA [35], and SAR [36] are entropy minimization-based methods that update batch normalization layer weights to minimize the entropy of current predictions. AdaContrast [3] relies on contrastive learning principles, combining contrastive learning and pseudo-labeling to enable discriminative feature learning for TTA. CoTTA[48] uses

Method	CIFAR100-C		ImageNet-C	
	ResNext-29A	WRN40-2A	ResNet-50	ResNet-50A
Source only	46.4†	45.4	82.0†	67.5
BN-1	35.4†	39.3	68.6†	53.8
TENT-cont.	60.9†	37.5	62.6†	49.6
AdaContrast	33.4†	37.1	65.5†	50.9
CoTTA	32.5†	38.2	62.7†	47.8
EATA	32.3	35.7	**58.8**	46.3
SAR	31.9	35.3	61.9	49.3
RMT	29.0†	34.3	59.8†	46.9
DPLOT (ours)	**27.8**	**31.8**	60.2	**44.6**

Table 1. Averaged classification error rate (%) for the CIFAR100-C and ImageNet-C benchmarks with the continual setting. The error rates are averaged for the given 15 corruption. † indicates that the result is reported by [10].

a teacher's augmentation-averaged pseudo-label for training and stochastic restore to mitigate error accumulation. RMT [10] uses the symmetric cross-entropy, which reduces effect of noisy label, for consistency regularization with pseudo-labels generated by a teacher and contrastive learning to pull the test feature space closer to the source domain. Moreover, RMT utilizes source replays during test-time to keep source knowledge. However, it is worth noting that we do not use source replays as it is not source-free during test-time.

4.1. Continual setting benchmark

First we evaluate our TTA framework on continual setting benchmarks. In Table 1, we provide the averaged classification error rates for CIFAR100-C and ImageNet-C benchmarks in the continual setting, considering 15 different corruptions. Our framework outperforms other methods for CIFAR100-C and ImageNet-C benchmarks except when using ResNet-50. In addition, we provide full comparison result of our framework with other TTA frameworks on CIFAR10-C in Table 2. Our framework achieves state-of-the-art performances, outperforming the best baseline by 5.5%, 6.4%, and 15.5% in mean error rate when using WRN28-10, WRN40-2A, and ResNet-18A, respectively.

4.2. Gradual setting benchmark

In Table 3, we report the average error rate across all severity levels and specfically with respect to level 5 in gradual setting benchmarks. We have following observations. First, TENT suffers from the error accumulation as observed in [25, 35] for gradual setting benchmarks due to more frequent updates. For example, TENT has an increased mean error rate compared to continual setting (e.g., 60.9 $\to$ 74.8 when using ResNext-29A). Second, entropy minimization-based methods (TENT, EATA, and SAR) tend to have degraded performance compared to consistency regularization-based methods (CoTTA and RMT) because the self-generated pseudo-label is more susceptible to error

TIME		t⟶															
Method		*Gaussian*	*shot*	*impulse*	*defocus*	*glass*	*motion*	*zoom*	*zoom*	*frost*	*fog*	*brightness*	*contrast*	*elastic*	*pixelate*	*jpeg*	Mean
WRN28-10	Source only†	72.3	65.7	72.9	46.9	54.3	34.8	42.0	25.1	41.3	26.0	9.3	46.7	26.6	58.5	30.3	43.5
	BN-1†	28.1	26.1	36.3	12.8	35.3	14.2	12.1	17.3	17.4	15.3	8.4	12.6	23.8	19.7	27.3	20.4
	TENT-cont.†	24.8	20.6	28.6	14.4	31.1	16.5	14.1	19.1	18.6	18.6	12.2	20.3	25.7	20.8	24.9	20.7
	AdaContrast†	29.1	22.5	30.0	14.0	32.7	14.1	12.0	16.6	14.9	14.4	8.1	10.0	21.9	17.7	20.0	18.5
	CoTTA†	24.3	21.3	26.6	11.6	27.6	12.2	10.3	14.8	14.1	12.4	7.5	10.6	18.3	13.4	17.3	16.2
	EATA	24.3	19.3	27.6	12.6	28.6	14.4	12.0	15.9	14.6	15.4	9.6	13.3	20.6	16.3	21.8	17.8
	SAR	28.3	26.0	35.8	12.7	34.6	13.9	12.0	17.5	17.6	14.9	8.2	13.0	23.5	19.5	27.2	20.3
	RMT†	21.9	18.6	24.1	10.8	**23.6**	12.0	10.4	13.0	12.4	11.4	8.3	10.1	**15.2**	11.3	14.6	14.5
	DPLOT (ours)	**19.4**	**16.5**	**22.5**	**10.0**	23.7	**11.1**	**9.6**	**12.3**	**11.9**	**10.7**	**7.7**	**9.8**	15.5	**10.8**	**13.9**	**13.7**
WRN40-2A	Source only	28.8	22.9	26.2	9.5	20.6	10.6	9.3	14.2	15.3	17.5	7.6	20.9	14.8	41.3	14.7	18.3
	BN-1	18.4	16.1	22.3	9.0	22.1	10.6	9.7	13.2	13.2	15.3	7.8	12.1	16.3	14.9	17.2	14.5
	TENT-cont.	15.0	12.1	16.8	9.5	18.0	11.7	9.9	11.8	11.4	13.7	9.3	11.4	16.8	13.1	19.8	13.4
	AdaContrast	16.2	12.6	16.9	8.3	18.1	10.0	8.4	10.7	9.8	12.0	**7.1**	8.4	14.2	12.1	13.8	11.9
	CoTTA	15.4	13.5	16.3	9.1	17.8	10.2	8.9	11.9	11.3	14.7	**7.1**	15.0	13.8	10.7	13.3	12.6
	EATA	15.3	11.7	16.6	9.0	17.3	10.8	9.1	11.4	10.6	13.5	8.8	10.6	15.5	11.7	16.4	12.5
	SAR	18.1	15.9	20.5	9.0	20.9	10.6	9.7	13.2	13.3	15.2	7.8	12.1	16.2	14.9	17.1	14.3
	RMT	15.3	12.5	15.4	8.7	15.8	9.6	8.1	**9.7**	9.6	10.4	7.2	9.9	**11.3**	**8.8**	**11.4**	10.9
	DPLOT (ours)	**12.4**	**10.5**	**13.9**	**7.8**	**14.9**	**9.1**	**8.0**	**9.7**	**9.0**	**9.9**	7.2	**8.3**	11.6	**8.8**	11.7	**10.2**
ResNet-18A	Source only	20.2	17.5	29.3	8.8	21.7	10.5	8.7	13.5	13.5	21.6	7.2	34.9	14.3	17.1	11.8	16.7
	BN-1	14.9	13.4	20.1	9.1	22.0	10.6	9.9	13.5	13.7	16.7	8.6	12.8	16.7	12.5	15.1	14.0
	TENT-cont.	13.1	11.4	17.7	9.2	19.8	12.3	10.9	13.5	12.8	16.6	10.4	11.4	16.3	12.2	16.2	13.6
	AdaContrast	13.2	11.2	16.0	8.5	17.9	9.8	8.5	11.2	9.5	13.5	6.9	8.2	13.6	10.1	10.9	11.3
	CoTTA	13.6	11.9	15.7	8.6	17.2	9.3	8.5	11.3	11.2	13.9	7.4	11.0	12.7	9.6	11.1	11.5
	EATA	13.0	10.9	16.1	8.5	17.1	10.0	8.7	10.6	10.2	13.9	7.9	9.5	14.5	10.5	13.3	11.6
	SAR	14.9	13.4	20.0	9.1	21.3	10.6	9.9	13.5	13.7	16.7	8.6	12.8	16.7	12.5	15.1	13.9
	RMT	13.3	10.9	14.9	8.4	15.1	9.5	7.9	9.5	9.6	10.2	7.5	9.0	10.9	8.5	9.7	10.3
	DPLOT (ours)	**10.3**	**9.1**	**13.2**	**7.2**	**14.0**	**7.8**	**6.7**	**8.1**	**7.6**	**8.9**	**5.8**	**6.5**	**9.7**	**7.0**	**8.1**	**8.7**

Table 2. Classification error rate (%) for the continual CIFAR10-C benchmark; the network is trained on clean CIFAR10 and evaluated on continually given corrupted test data. We evaluate our framework with various models: WRN28-10, WRN40-2A, and ResNet18A. The results are averaged over five runs. Also, the best result is indicated in **bold**. † indicates that the result is reported by [10].

	CIFAR10-C			CIFAR100-C		ImageNet-C	
Method	WRN28-10	WRN40-2A	ResNet-18A	ResNext-29A	WRN40-2A	ResNet-50	ResNet-50A
Source only	24.7 / 43.5†	10.4 / 18.3	9.7 / 16.7	33.6 / 46.4†	34.7 / 46.7	58.4 / 82.0†	44.9 / 67.2
BN-1	13.7 / 20.4†	10.5 / 14.5	10.5 / 14.0	29.9 / 35.4†	33.7 / 39.3	48.3 / 68.6†	39.3 / 54.8
TENT-cont.	20.4 / 25.1†	15.0 / 18.2	20.0 / 22.9	74.8 / 75.9†	63.5 / 65.8	46.4 / 58.9†	38.5 / 47.0
AdaContrast	12.1 / 15.8†	8.9 / 10.9	8.2 / 10.0	33.0 / 35.9†	35.9 / 39.1	66.3 / 72.6†	56.7 / 61.5
CoTTA	10.9 / 14.2†	8.6 / 11.3	8.0 / 9.7	26.3 / 28.3†	32.8 / 37.2	38.8 / 43.1†	32.0 / 33.8
EATA	16.0 / 20.6	11.7 / 14.5	11.4 / 13.9	32.0 / 34.4	33.1 / 36.8	40.7 / 49.7	36.0 / 41.2
SAR	13.6 / 20.3	8.7 / 11.4	7.6 / 10.0	28.7 / 31.9	30.7 / 34.5	42.8 / 55.8	36.5 / 45.7
RMT	9.3 / **10.4**†	7.7 / 8.5	11.7 / 12.5	26.4 / 26.9†	31.1 / 32.1	39.3 / 41.5†	33.5 / 34.4
DPLOT (ours)	**8.8 / 10.4**	**7.2 / 8.0**	**6.3 / 7.0**	**23.9 / 25.0**	**27.3 / 28.6**	**37.2 / 40.2**	**30.8 / 31.6**

Table 3. Classification error rate (%) for CIFAR10-C, CIFAR100-C, and ImageNet-C benchmarks with the gradual setting. Error rates are separately reported by averaging over all severity levels and averaging only over the highest severity level 5 (@level 1-5 / @level 5). † indicates that the result is reported by [10].

accumulation due to model collapse and forgetting [35, 36]. Lastly, our framework consistently outperforms other competitive methods across different architectures and datasets, highlighting the benefit of using domain-specific block selection and paired-view consistency for long-time adaptation against corruptions.

5. Discussion

5.1. Component analysis

To understand the effect of each component of our framework, we provide the adaptation performance with various configurations in Table 4. First, we present the per-

Method	CIFAR10-C	CIFAR100-C	ImageNet-C
DPLOT (A)	**8.8 / 10.4**	**23.9 / 25.0**	**30.8 / 31.6**
– Paired-view consistency (B)	11.3 / 14.2	25.9 / 28.8	37.4 / 42.5
– EMA teacher (C)	12.5 / 15.8	25.8 / 28.5	37.9 / 42.8
– Block selection (D)	19.3 / 24.5	66.9 / 68.7	38.5 / 47.0

Table 4. Classification error rate (@level 5/@level 1-5) for the gradual benchmarks with various configurations. We use WRN28-10, ResNext-29A, and ResNet50A for CIFAR10-C, CIFAR100-C, and ImageNet-C datasets, respectively. Note that, we gradually remove each component from DPLOT, and when none of the components are used (D), the method is equivalent to TENT.

formance when using all components: the paired-view consistency, EMA teacher, and the block selection (A). If we remove paired-view consistency using teacher-generated pseudo-labels, the performance significantly drops (B). Furthermore, the performance slightly drops if we do not use ensemble prediction as observed in [10] (C). Finally, it is demonstrated that the performance significantly drops when we update BN layers rather than domain-specific blocks by entropy minimization (D). These results show that all components are necessary for stable long-term adaptation.

5.2. Parameter sensitivity

In Table 5, we empirically demonstrate the influence of the hyperparameter γ for block selection. As γ increases, fewer blocks are selected and vice versa. For instance, only one block is selected for entropy minimization if we set γ to 0.999, while all blocks except the lowest one are selected for γ of 0.0. We find that small γ significantly damages the adaptation performance, but it tends to have robust performance for the range of [0.75, 0.95]. The results demonstrate that as blocks not involve in domain-specific feature extraction are selected to be updated by entropy minimization, the model becomes vulnerable to the model collapse (i.e., predicts all samples to one class [35, 36]). This vulnerability, caused by modified domain-invariant feature space, can be alleviated by our block selection. Also, updating selected blocks using γ in the range of [0.75, 0.95] improves adaptation performance compared to updating the BN weights.

5.3. Effect of block selection

In Figure 3, we present the results of proposed block selection for WRN28-10 and ResNext-29A for CIFAR10 and CIFAR100, respectively. It is demonstrated that the shallower blocks show high similarity between prototypes before and after entropy minimization, while deeper blocks show lower similarity in (a, b). These findings align with observations in [5, 51, 59] that style knowledge (i.e., domain specific feature unrelated to the task) being predominantly captured by shallow blocks. As expected, we find that updating blocks with high similarity does not modify the domain-invariant feature space even after the long-time adaptation (i.e., not forgetting source knowledge (a, b)). Moreover, when applying domain-specific block selection to other methods instead of updating all BN layers (c, d), we find that our block selection methods improves other methods as well. In particular, it reduces the error rate of TENT and EATA by 46.6% and 32.5% in CIFAR10-C, respectively. This can be interpreted as updating domain-specific blocks can alleviate error accumulation caused by model collapse, as addressed by [35, 36]. However, there is no significant improvement in the SAR method. This is because SAR uses a model reset approach, which restores model parameters to their original values; thus, the adaptation performance does not differ significantly.

Threshold γ	CIFAR10-C	CIFAR100-C	ImageNet-C
0.999	9.2 / 11.4	24.8 / 26.1	37.8 / 41.2
0.95	8.8 / 10.7	24.4 / 25.6	37.3 / 40.4
0.9	8.8 / 10.7	24.3 / 25.5	37.1 / 40.2
0.75	8.8 / 10.4	23.9 / 25.0	37.2 / 40.2
0.5	11.1 / 12.5	23.8 / 24.5	39.1 / 41.2
0.25	12.8 / 14.5	92.6 / 93.5	99.1 / 99.8
0.0	80.7 / 81.2	92.7 / 93.5	99.6 / 99.9
BatchNorm	9.6 / 12.8	25.4 / 26.9	37.7 / 40.7

Table 5. Classification error rate (@level 5/@level 1-5) for the gradual benchmarks with various thresholds γ for block selection. For CIFAR10-C, CIFAR100-C, and ImageNet-C, network architecture of WRN28-10, ResNext-29A, and ResNet50 is used, respectively. BatchNorm indicates that block selection is not used and BN weights are updated, as in previous methods [35, 36, 47].

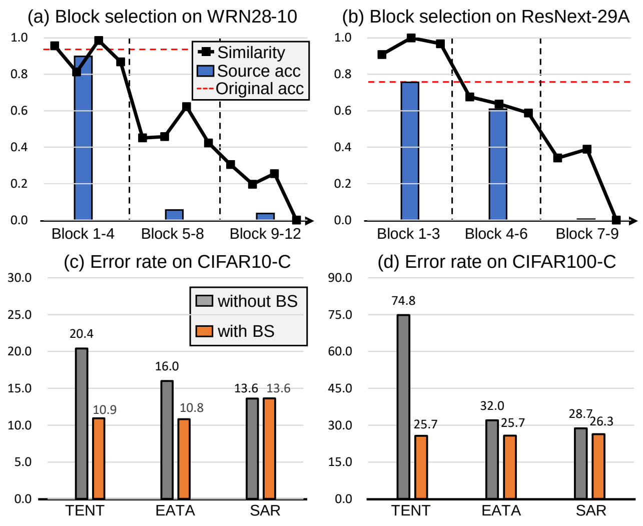

Figure 3. Illustrations of our proposed block selection results (a, b) and classification error rate (@level 1-5) for the gradual setting benchmark using other entropy minimization-based methods with or without block selection (c, d). Additionally, in (a) and (b), the source accuracy after the long-time adaptation (i.e., gradual setting benchmark) with selected blocks is shown in a bar graph.

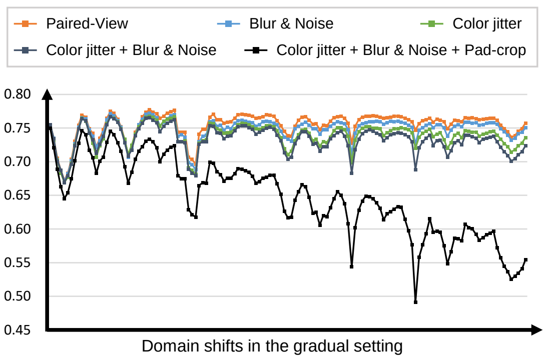

Figure 4. Performance of our framework with various pseudo-label generation setting including our paired-view (orange) and others. We use ResNext-29A for CIFAR100-C.

5.4. Effect of paired-view consistency

In CoTTA [48], pseudo-labels are generated by averaging the teacher's predictions for the given 32 augmented images transformed by random augmentation including random pad-crop, color jitter, random affine transform, Gaussian blur, random horizontal flip, and Gaussian noise. In our framework, we generate pseudo-labels by simply averaging the teacher's predictions for two images: original test and horizontally flipped images. This is based on the insight that the simple flip operation does not create domain gap between the augmented image and the test image, which can reduces the quality of pseudo label.

In Figure 4, we compare the adaptation performance of our framework for the gradual setting with different pseudo-label generation methods. Specifically, while entropy minimization on the domain-specific blocks is conducted, the pseudo label is generated by averaging teacher predictions from (i) 2 images with paired-view (orange; ours), (ii) 32 images with Gaussian noise and Gaussian blur (blue), (iii) 32 images with color jitter (green), (iv) 32 images with noise, blur, and color jitter (grey), and (v) 32 images with random pad-crop, affine transform, noise, blur, and color jitter (black; CoTTA). Note that, random horizontal flip with 0.5 probability is included in (ii-v). As expected, the adaptation with our pseudo-label generation outperforms others. It is worth noting that using random pad-crop, which adds a 16-pixel border to the CIFAR image and subsequently performs random cropping to a 32x32 size, significantly decreases the pseudo-label quality due to the potential for objects to be partially cropped.

5.5. Single sample test-time adaptation

Since the single prediction for a single input is crucial for some real-time systems, we consider single-sample TTA as investigated in [10]. In the single-sample TTA setting, the last b test samples are stored in a memory buffer. After every b steps, the model parameters are updated by test-time adaptation methods with a b-size batch from the memory. Following [10], we decrease the learning rate by *original batch size*$/b$ due to the more frequent updates. In this setting, the challenge of the TTA is that error accumulation also increases due to the frequent updates. Table 6 provides the results for single-sample TTA with various buffer sizes b. We observed that previous methods suffer from a small batch-size, but our method is relatively strong across various buffer sizes. This demonstrates that our method alleviates error accumulation caused by frequent updates through proper pseudo-label generation.

Method	Window size				Mean
	8	16	32	64	
Source only		43.5			43.5
BN-1	26.2	23.1	21.5	20.8	22.9
TENT	23.6	20.2	18.6	18.9	20.3
CoTTA	27.4	37.5	17.1	15.0	24.3
EATA	22.9	19.2	17.6	17.9	19.4
RMT	32.3	21.8	15.8	12.4	20.6
DPLOT (ours)	**16.4**	**13.1**	**11.6**	**11.3**	**13.1**

Table 6. Classification rate (@level 1-5) in the gradual setting of the CIFAR10-C benchmark with WRN28-10, using single-sample TTA, while considering different buffer sizes b.

6. Conclusion

In this work, we propose DPLOT to address proper pseudo-label generation. The proposed framework is based on two components: domain-specific block selection before deployment and paired-view pseudo-labeling. After deployment, we use entropy minimization to update blocks involved in domain-specific feature extraction. Subsequently, we employ paired-view consistency loss, which forces the model to produce the exact prediction of the pseudo-label generated by averaging the teacher's predictions for the test and its corresponding flipped inputs. Extensive experiments demonstrated that our framework outperforms previous competitive methods by a large margin in TTA benchmarks. Also, DPLOT does not modify the network during the training stage, making it easily applicable.

Acknowledgement

This work was partially supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2022-0-00951, Development of Uncertainty Aware Agents Learning by Asking Questions) and by LG Electronics and was collaboratively conducted with the Advanced Robotics Laboratory within the CTO Division of the company.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 2

[2] Philipp Benz, Chaoning Zhang, Adil Karjauv, and In So Kweon. Revisiting batch normalization for improving corruption robustness. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 494–503, 2021. 3

[3] Dian Chen, Dequan Wang, Trevor Darrell, and Sayna Ebrahimi. Contrastive test-time adaptation. In *CVPR*, 2022. 2, 5, 1

[4] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018. 4

[5] Sungha Choi, Seunghan Yang, Seokeon Choi, and Sungrack Yun. Improving test-time adaptation via shift-agnostic weight regularization and nearest source prototypes. In *European Conference on Computer Vision*, pages 440–458. Springer, 2022. 2, 4, 7

[6] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 4

[7] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*, 2020. 5, 1

[8] Sebastian Cygert and Andrzej Czyżewski. Toward robust pedestrian detection with data augmentation. *IEEE Access*, 8:136674–136683, 2020. 4

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 5

[10] Mario Döbler, Robert A Marsden, and Bin Yang. Robust mean teacher for continual and gradual test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7704–7714, 2023. 2, 4, 5, 6, 7, 8, 1

[11] Geoff French, Michal Mackiewicz, and Mark Fisher. Self-ensembling for visual domain adaptation. In *International Conference on Learning Representations*, 2018. 2

[12] Angus Galloway, Anna Golubeva, Thomas Tanay, Medhat Moussa, and Graham W Taylor. Batch normalization is a cause of adversarial vulnerability. *arXiv preprint arXiv:1905.02161*, 2019. 2

[13] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR, 2015. 2

[14] Robert Geirhos, Carlos RM Temme, Jonas Rauber, Heiko H Schütt, Matthias Bethge, and Felix A Wichmann. Generalisation in humans and deep neural networks. *Advances in neural information processing systems*, 31, 2018. 1

[15] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. *Advances in neural information processing systems*, 17, 2004. 1

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3, 5

[17] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019. 1, 5

[18] Dan Hendrycks, Norman Mu, Ekin D. Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. AugMix: A simple data processing method to improve robustness and uncertainty. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020. 5, 1

[19] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. *ICCV*, 2021. 5

[20] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning*, pages 1989–1998. Pmlr, 2018. 2

[21] Lukas Hoyer, Dengxin Dai, Haoran Wang, and Luc Van Gool. Mic: Masked image consistency for context-enhanced domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11721–11732, 2023. 1, 2

[22] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015. 2, 3

[23] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 5

[24] Klim Kireev, Maksym Andriushchenko, and Nicolas Flammarion. On the effectiveness of adversarial training against common corruptions. In *Uncertainty in Artificial Intelligence*, pages 1012–1021. PMLR, 2022. 5, 1

[25] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017. 5

[26] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009. 5

[27] Jogendra Nath Kundu, Rahul Mysore Venkatesh, Naveen Venkat, Ambareesh Revanur, and R Venkatesh Babu. Class-incremental domain adaptation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII 16*, pages 53–69. Springer, 2020. 2

[28] Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting batch normalization for practical domain adaptation, 2017. 2, 3

[29] Jian Liang, Ran He, and Tieniu Tan. A comprehensive survey on test-time adaptation under distribution shifts. *arXiv preprint arXiv:2303.15361*, 2023. 2

[30] Raphael Gontijo Lopes, Dong Yin, Ben Poole, Justin Gilmer, and Ekin D Cubuk. Improving robustness without sacrificing accuracy with patch gaussian augmentation. *arXiv preprint arXiv:1906.02611*, 2019. 4

[31] Yulei Lu, Yawei Luo, Antao Pan, Yangjun Mao, and Jun Xiao. Domain generalization with global sample mixup. In *European Conference on Computer Vision*, pages 518–529. Springer, 2022. 1

[32] Robert A Marsden, Mario Döbler, and Bin Yang. Gradual test-time adaptation by self-training and style transfer. *arXiv preprint arXiv:2208.07736*, 2022. 1, 5

[33] M Jehanzeb Mirza, Jakub Micorek, Horst Possegger, and Horst Bischof. The norm must go on: Dynamic unsupervised domain adaptation by normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14765–14775, 2022. 2

[34] 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. 1

[35] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *International conference on machine learning*, pages 16888–16905. PMLR, 2022. 1, 2, 4, 5, 6, 7

[36] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards stable test-time adaptation in dynamic wild world. In *Internetional Conference on Learning Representations*, 2023. 2, 4, 5, 6, 7, 1, 3

[37] B Prasanna, Sunandini Sanyal, and R Venkatesh Babu. Continual domain adaptation through pruning-aided domain-specific weight modulation. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2457–2463. IEEE, 2023. 2, 4

[38] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International conference on machine learning*, pages 5389–5400. PMLR, 2019. 1

[39] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015. 1

[40] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. ACDC: The adverse conditions dataset with correspondences for semantic driving scene understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 4

[41] Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. *Advances in neural information processing systems*, 33:11539–11551, 2020. 1

[42] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948. 4

[43] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 1

[44] Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in neural information processing systems*, 33:596–608, 2020. 1

[45] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in neural information processing systems*, 30, 2017. 1, 3, 4

[46] Tuan-Hung Vu, Himalaya Jain, Maxime Bucher, Matthieu Cord, and Patrick Pérez. Advent: Adversarial entropy minimization for domain adaptation in semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2517–2526, 2019. 1, 4

[47] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021. 1, 2, 3, 4, 5, 7

[48] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7201–7211, 2022. 1, 2, 3, 4, 5, 8

[49] Xiao Wang and Guo-Jun Qi. Contrastive learning with stronger augmentations. *IEEE transactions on pattern analysis and machine intelligence*, 45(5):5549–5560, 2022. 2

[50] Yisen Wang, Xingjun Ma, Zaiyi Chen, Yuan Luo, Jinfeng Yi, and James Bailey. Symmetric cross entropy for robust learning with noisy labels. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 322–330, 2019. 2, 4, 5

[51] Yu Wang, Rui Zhang, Shuo Zhang, Miao Li, YangYang Xia, XiShan Zhang, and ShaoLi Liu. Domain-specific suppression for adaptive object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9603–9612, 2021. 2, 3, 7

[52] Aming Wu, Yahong Han, Linchao Zhu, and Yi Yang. Instance-invariant domain adaptive object detection via progressive disentanglement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):4178–4193, 2021.

[53] Aming Wu, Rui Liu, Yahong Han, Linchao Zhu, and Yi Yang. Vector-decomposed disentanglement for domain-invariant object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9342–9351, 2021. 2

[54] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 2

[55] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017. 5, 1

[56] Yeonguk Yu, Sungho Shin, Seongju Lee, Changhyun Jun, and Kyoobin Lee. Block selection method for using feature norm in out-of-distribution detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15701–15711, 2023. 4

[57] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016. 5, 1

[58] Bowen Zhao, Chen Chen, and Shu-Tao Xia. DELTA: DEGRADATION-FREE FULLY TEST-TIME ADAPTATION. In *The Eleventh International Conference on Learning Representations*, 2023. 2

[59] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain generalization with mixstyle. In *ICLR*, 2021. 7

[60] Kaiyang Zhou, Ziwei Liu, Yu Qiao, Tao Xiang, and Chen Change Loy. Domain generalization: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 1

Domain-Specific Block Selection and Paired-View Pseudo-Labeling for Online Test-Time Adaptation

Supplementary Material

Architectures

Here, we describe the details of each architecture used in the experiments. All source pre-trained weights are provided by RobustBench[7].

- WRN28-10. We use a wide residual network [57] with a depth of 28 and a width of 10. Additionally, the network has 12 residual blocks.
- WRN40-2A. We use a wide residual network with a depth of 40 and a width of 2. This network is trained using AugMix [18], a data processing technique that improves the model for unseen corruptions. The network has 18 residual blocks.
- ResNet-18A. We use a residual network with 18 layers trained by adversarial training to improve robustness against corruptions [24]. The network has 8 residual blocks.
- ResNext-29A. We use ResNext[55] with 29 layers. ResNext improves ResNet architecture by employing cardinality from group convolution. The network is trained with AugMix [18] augmentation by SGD optimizer using an initial learning rate of 0.1, which decays following a cosine learning rate and is trained for 200 epochs. The network has 9 blocks.
- ResNet50. We use a residual network with 50 layers for the ImageNet benchmark. The network has 16 blocks.
- ResNet50A. We also use a residual network with 50 layers trained using AugMix for the ImageNet benchmark. The network has 16 blocks.

Details of the baselines

Also, we describe the details of each baseline used in the experiments.

- Source only. It is a baseline method that does not use any test-time adaptation method. Thus, it represents the performance of the source pre-trained model directly used in shifted domains.
- BN-1 [34, 41]. BN-1 is a simple method that recalculates the batch normalization statistics using the current test batch.
- TENT-cont. [47]. It is an entropy minimization-based method where weights of the batch normalization layer in the network are updated to minimize the entropy of current prediction as follows.

$$\mathcal{L}_{en} = -\sum_{c} \hat{y}_c \log \hat{y}_c. \tag{8}$$

- AdaContrast [3]. This is a test-time adaptation method based on the contrastive learning. It uses self-training with pseudo-labels along with contrastive learning, enabling discriminative feature learning by pulling positive pairs closer and pushing negative pairs away. The paired images are generated using weak and strong augmentations.
- CoTTA [48]. It is a test-time adaptation framework that works in a continually changing environment based on consistency regularization. This framework uses the teacher's augmentation-averaged pseudo-label, generated by averaging predictions for 32 augmented images, to train the student. Also, it employs stochastic restore, which restores the current weight to the original source pre-trained weight, to reduce error accumulation.
- EATA [35]. It is an improved version of TENT that uses reliable sample selection. Specifically, they investigate the sample that makes model collapse and find that high-entropy samples are reason. Therefore, they only use low-entropy samples to update the weight parameters of batch normalization by entropy minimization during test-time as follows.

$$S^{ent}(x) = \frac{1}{\exp[E(x;\theta) - E_0]} \cdot \mathbb{I}_{\{E(x;\theta)<E_0\}}(x), \tag{9}$$

where $\mathbb{I}_{\{\cdot\}}(\cdot)$ is an indicator function, $E(x;\theta)$ is the entropy of a given sample x. Also, E_0 is a threshold. It excludes high-entropy samples and assigns higher weights to low-entropy samples. Moreover, it utilizes Fisher regularize to alleviate the catastrophic forgetting issue by preventing model parameters, important for source knowledge, from changing too much.
- SAR [36]. It is also based on the entropy minimization-based method. Specifically, it proposes reliable and sharpness-aware entropy minimization, which selects low-entropy samples and encourages the model to go to a flat area of the loss surface to further achieve good generalization ability. Also, it provides a model recovery scheme based on the insight that models after collapse will produce very small entropy loss. This method is based on Group Norm and Layer Norm but can apply to Batch Norm networks as well.
- RMT [10]. RMT is a consistency regularization-based method where the student model is trained by pseudo-labels generated by EMA teacher. The student model has two losses: symmetric cross-entropy between student's prediction and teacher's prediction. It also utilizes contrastive loss that pulls current test features towards source

Architectures	Blocks																	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
WRN28-10	0.89	0.92	1.0	0.98	0.49	0.47	0.54	0.49	0.30	0.20	0.09	0.0	-	-	-	-	-	-
WRN40-2A	0.99	1.0	1.0	0.99	1.0	1.0	0.9	0.76	0.85	0.78	0.80	0.86	0.34	0.59	0.45	0.0	0.43	0.14
ResNet18A	0.93	1.0	0.58	0.53	0.29	0.37	0.26	0.0	-	-	-	-	-	-	-	-	-	-
ResNext-29	0.91	1.0	0.97	0.67	0.67	0.59	0.34	0.39	0.0	-	-	-	-	-	-	-	-	-
WRN40-2A	0.95	0.96	0.97	0.99	1.0	1.0	0.80	0.71	0.72	0.69	0.70	0.69	0.28	0.38	0.31	0.27	0.20	0.0
ResNet50	0.96	1.0	0.95	0.38	0.57	0.52	0.51	0.31	0.58	0.54	0.48	0.46	0.43	0.0	0.26	0.12	-	-
ResNet50A	0.99	1.0	1.0	1.0	0.97	0.99	1.0	0.47	0.54	0.53	0.41	0.48	0.48	0.24	0.23	0.0	-	-

Table 7. Block selection results. The block with grey background color refers to the selected block as it has larger value than 0.75. Architectures in 1-3rd row, 4-5th row, and 6-7th row are for CIFAR10-C, CIFAR100-C, and ImageNet-C, respectively.

feature space. It uses source replays to further use source training data in test-time, but we do not compare our method to this result since it is not source-free.

Domain-specific block selection results

In Table 7, we present the specific results of the block selection. As mentioned in Section 4, we use WRN28-10, WRN40-2A, and ResNet18A for CIFAR10-C. Additionally, we employ ResNext-29A and WRN40-2A for CIFAR100-C. Finally, we utilize ResNet50 and ResNet50A for ImageNet-C.

We observed that shallow blocks tend to exhibit high similarity, whereas deeper blocks tend to show smaller similarity. In the experiment, we select blocks with a similarity higher than 0.75. For instance, we choose 1-4 blocks when using the WRN28-10 architecture, and 1-2 blocks when using ResNet18A.

Computational time

Since our framework updates the parameters twice for a given batch, the computational time can be burdensome for real-time systems. Despite that, our framework uses only one augmentation (i.e., horizontal flip) for generating pseudo-labels, which is much lighter than CoTTA's 32 augmentations. In Table 8, we provide the computational time when using ResNext-29 for CIFAR100-C. All computational times are averaged over 15,000 batches. Our test setting is based on PyTorch 2.0.1, RTX 4090 GPU, and Linux 20.04. Our method has a relatively large computational time compared to other methods, which may be a limitation.

Method	Architectures	
	WRN28-10	ResNext-29A
Source	45.3	29.0
BN-1	46.2	31.3
TENT	96.8	70.9
AdaContrast	315.1	200.7
CoTTA	842.9	806.3
EATA	98.5	73.5
SAR	189.8	137.6
RMT	322.1	203.0
DPLOT (ours)	534.0	336.7

Table 8. Computational time (ms) to predict and adaptation of a single batch with various methods.

Performance at the similar computational time

We present the adaptation performance, utilizing a similar computational time across various methods in Table 9, to demonstrate the effectiveness of each method while considering the computational burden. To ensure all methods have similar computational time, we modify the number of updates for each method during the test-time stage. Specifically, we adopt the setting of single sample TTA as described in Section 5.5, using a buffer size b of 64. Subsequently, we adjust the update frequency k to decrease computational time by reducing adaptation. With k, the model parameters are updated using a batch of size b for every $b \times k$ steps. For example, when using $b = 64$ and $k = 1/4$, the model parameters are updated using the last 64 samples every 254 steps. This reduction in adaptation significantly decreases computational time. We decrease the learning rate by *original batch size*$/(b \times k)$ and set k to achieve a computational time of around 200 ms. We did not modify the entropy minimization-based methods, as they already require computational times below 200 ms.

As shown in Table 9, our method still outperforms other methods with a similar computational cost for the CIFAR100-C gradual benchmark. It demonstrated that our method provides relatively strong adaptation performance, even when we adapt the model using our method only once for every six batches.

Method	Error rates	Update Frequency	Computational Cost
TENT	74.2	1	158
EATA	33.6	1	198
SAR	33.7	1	317
AdaContrast	43.8	1/3	290
CoTTA	44.9	1/10	229
RMT	31.1	1/3	211
DPLOT (ours)	**26.8**	1/6	209

Table 9. Classification error rate (%; @level 1-5) for the CIFAR100-C gradual benchmark with ResNext-29A architecture.

Performance on mixed domain shifts setting

We also present experimental results in a mixed domain setting, as in SAR[36], where all kinds corruptions are given to test images simultaneously. Specifically, test data from 15 corruption-type domains are given to the model concurrently. In this setting, the challenge lies in adapting the TTA framework to a model facing multiple domains which have diverse feature statistics. As depicted in Table 10, our method outperforms other methods across various architectures, except for ResNext-29A. We believe that entropy minimization on domain-specific block is also effective in adjusting domain-specific feature extraction, not only for a single corruption but also for a mixture of given corruptions.

Method	Architectures				
	WRN28-10	WRN40-2A	ResNet18A	ResNext-29A	WRN40-2A
Source	43.5	18.3	16.7	46.5	46.7
BN-1	33.8	20.2	19.4	45.8	47.7
TENT	37.2	16.6	18.5	84.3	56.2
AdaContrast	26.1	15.0	14.0	41.9	42.6
CoTTA	32.3	16.7	15.7	43.1	46.9
EATA	27.9	14.9	14.4	**37.5**	41.0
SAR	33.7	19.9	19.4	45.2	42.3
RMT	26.9	15.2	13.8	38.4	41.9
DPLOT (ours)	**24.3**	**13.0**	**11.9**	38.4	**38.8**

Table 10. Classification error rate (%) for CIFAR10-C and CIFAR100-C with the mixed domain shifts setting. We use WRN28, WRN40-2A, and ResNet18A for CIFAR10-C benchmark, while ResNext-29A and WRN40-2A are used for CIFAR100-C benchmarks.

Various corruption settings for block selection

We present the block selection results with different corruption settings for source training images in Table 11. In the main experiments, we select the block that has a similarity larger than 0.75 after entropy minimization with Gaussian noise (0.0 μ and 0.5 σ) added to source training images. To investigate the effect of corruption on block selection, we vary the Gaussian noise setting. For example, we change the σ to 0.1 or 1.0. Additionally, we calculate the similarity with source training images after changing brightness or contrast (i.e., twice the brightness, twice the contrast).

As shown in Table 11, we observe that Gaussian noise with a small σ, changing brightness, and changing contrast force each block to have high similarity after entropy minimization. We find that there are degraded adaptation performances with these selected blocks (e.g., 10.4 $\rightarrow$ 70.8 error rates for CIFAR10-C gradual benchmark with blocks selected by Gaussian noise with 0.1 σ). We argue that if we use a large enough σ compared to the training images' standard deviation (i.e., about 0.45 for CIFAR10 images), we can select blocks that do not lead to error accumulation, and the model's domain-specific feature extraction can be well-adjusted during test-time. On the contrary, when we use relatively simple corruption such as brightness and contrast, the domain-specific features do not differ much from the original source domain, so our block selection cannot work well for finding blocks that are involved in domain-specific feature extraction.

Corruptions	Blocks											
	1	2	3	4	5	6	7	8	9	10	11	12
GN (0, 0.1)	1.00	1.00	1.00	1.00	0.98	0.95	0.94	0.90	0.22	0.00	0.19	0.22
GN (0, 0.5)	0.89	0.92	1.00	0.98	0.49	0.47	0.54	0.49	0.30	0.20	0.09	0.00
GN (0, 1.0)	0.86	0.82	0.92	1.00	0.60	0.41	0.31	0.49	0.19	0.37	0.00	0.02
Brightness	1.00	1.00	1.00	1.00	0.99	0.99	1.00	1.00	0.22	0.00	0.29	0.10
Contrast	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.92	0.00	0.83	0.28

Table 11. Block selection results with various corruption setting. The block with grey background color is selected. Also, GN (a, b) refers to the Gaussian noise with a mean and b standard deviation. The block selection is conducted for CIFAR10-C with WRN28-10.

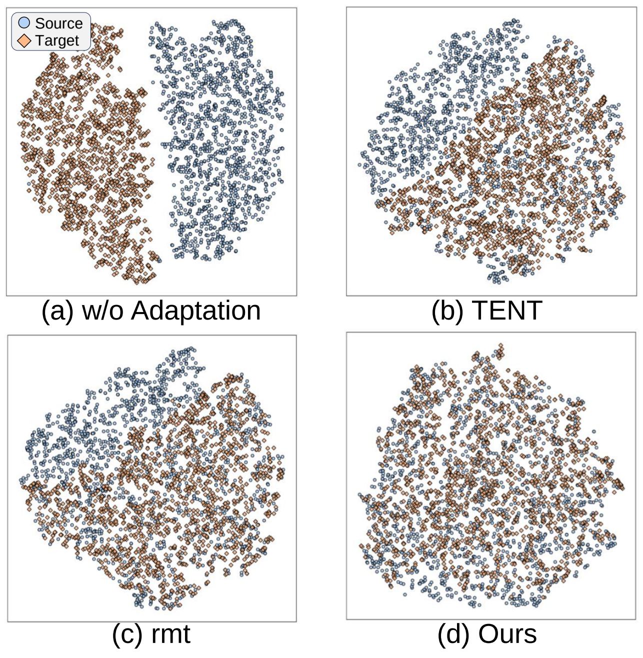

Figure 5. Visualization of features for given original source images (blue) and given Gaussian noise images (orange) produced from the first block of WRN28-10 after adaptation.

Domain-specific feature visualization

We provide feature visualizations of the selected blocks (i.e., block 1-4 of WRN28-10) after adaptation in Figure 5. In the experiment, we use the network checkpoint after adaptation on Gaussian noise domain with TENT (b), RMT (c), and Ours (d). Subsequently, we extract features from the source test data and Gaussian noise-added test data, and visualize them using the t-SNE method. Since we specifically train the blocks that are involved in domain-specific

Arch.	Ours	Ours w/ jigsaw	[56]	[37]
WRN28-10	**13.7**	63.5	16.4	14.3
ResNet18A	**8.7**	9.0	10.6	9.5

Table 12. Mean classification error rate (%; ↓) comparison with the selected block with [56] or [37] on CIFAR10-C continual setting benchmark. The best result is indicated in **bold**.

	Source	TENT	EATA	CoTTA	RMT	Ours
Error rate	88.1	91.2	73.1	73.4	77.9	**72.5**

Table 13. Mean classification error rate (%; ↓) comparison with VGG11-BN on ImageNet-C continual setting benchmark.

feature extraction, source features and target features appear to be well mixed, which can be interpreted as the domain-specific feature extraction being well adjusted. Thus, it demonstrates that our method can outperform other methods when the given test data is corrupted by noise, as shown in Table 2.

Comparison with other block selection

Table 12 provides performance comparison results with selected blocks using three methods: (1) ours w/ jigsaw, using our block selection with jigsaw puzzles as in [56], (2) [56], using block selection for OOD detection of [56] (i.e., penultimate block), and (3) [37], using selected weights by L1-pruning (40% ratio as in [37]). Our block selection outperforms others since [56] and [37] use selection method for other task (OOD detection and Pruning), and cannot conduct domain-specific entropy minimization.

Evaluation on non-residual network

We report the adaptation performance when using VGG11-BN, a non-residual network pre-trained on ImageNet provided by PyTorch, in Table 13. We use VGG11-BN over VGG11 since entropy minimization methods (e.g., TENT, EATA) require BN layers. As VGG does not have blocks, we perform layer-wise selection instead of block-wise selection. Table 13 shows that our method works well for a non-residual network.

Extension to segmentation

Cityscpaes-to-ACDC is a continual semantic segmentation task designed to evaluate the model's robustness against to distribution shifts in the real world. The source model is an off-the-shelf pre-trained segmentation model trained on the Cityscapes dataset [6]. Then, the target domains are four adverse visual conditions, including Fog, Night, Rain, and Snow, from the Adverse Conditions Dataset (ACDC) [40]. The ACDC dataset contains images share the same semantic class with Cityscapes. Following [5, 48], we evaluate the model for the given 10 times-repeated sequence group (i.e., in total 40: fog→night→rain→snow→fog→...→snow) to demonstrate long-term performance and robustness agains to error accumulation.

Time	t⟶								
Round	1				10				Mean
Condition	fog	night	rain	snow	fog	night	rain	snow	
Source	58.1	17.4	42.7	41.7	58.1	17.4	42.7	41.7	40.0
TENT	**61.1**	20.0	38.1	35.5	5.8	1.0	6.6	6.7	10.1
CoTTA	60.8	19.6	47.1	47.1	58.3	20.6	45.0	43.6	43.5
PLOT (ours)	60.5	**20.3**	**47.9**	**50.6**	**59.4**	**25.8**	**50.1**	**49.1**	**46.4**

Table 14. Semantic segmentation results (mIoU in %) on the Cityscapes-to-ACDC online continual test-time adaptation task, where the segmentation model is trained on sunny Cityscapes data and evaluated on the adverse weather ACDC data.

Specifically, we utilize a ResNet50-based DeepLabV3+ [4] model pre-trained on Cityscapes. Since only CoTTA and TENT apply their methods and open their code to the segmentation task, we compare our method with theirs. The multi-scaling (scale factor of [0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0]) input with flip is used for generating CoTTA's pseudo-label, while we only use horizontal flip for our pseudo-label. We use down-sampled resolutions of 769×769 as inputs to the network and evaluated under the original resolution of 1920×1080. Also, We use Adam optimizer with the learning rate 8 times smaller than the default one and batch size of 1. It is worth noting that we use the first block to adjust domain-specific feature extraction without using block selection. Since the image for segmentation has multiple object class, the output feature of the model cannot be directly used for generating prototype vectors. Instead, as the first block is selected across various architectures in classification tasks, we select the first block (which is named as "stem" in Pytorch implementation).

The experimental results are summarized in Table 14. It is demonstrated that our framework with pseudo-label generated by paired-view is also effective for semantic segmentation tasks. Noticeably, TENT method outperforms other methods in Fog domain of first round, but subsequently, suffers from the error accumulation.