Appendix: Distribution Alignment for Fully Test-Time Adaptation with Dynamic Online Data Streams

A Algorithm of Domain Shift Detection

Algorithm 1 Domain Shift Detection Mechanism

Require: DA loss for the current batch $\mathcal{L}_{DA}^{B_t}$, short-term window W_s with length p, long-term window W_l with length q, threshold factor τ

if len$(W_l) < q$ **then**
  Append $\mathcal{L}_{DA}^{B_t}$ to W_l
  $W_s = W_s[-(q-1):] + [\mathcal{L}_{DA}^{B_t}]$ {Maintain W_s size by sliding}
else
  Compute average DA loss over W_l as avg_l
  Compute average DA loss over W_s as avg_s
  if $avg\_s > \tau \cdot avg\_l$ **then**
    Reset W_l and W_s to contain only the current $\mathcal{L}_{DA}^{B_t}$
    Reset affine layers to initial parameters
    Reset normalization layers using Eq.6, Eq.7
  else
    $W_l = W_l[1:] + [\mathcal{L}_{DA}^{B_t}]$ {Update W_l by sliding}
    $W_s = W_s[1:] + [\mathcal{L}_{DA}^{B_t}]$ {Update W_s by sliding}
  end if
end if

B More Dataset Details

CIFAR10/100-C, ImageNet-C. CIFAR10-C, CIFAR100-C, and ImageNet-C datasets are from the RobustBench [1] benchmark. These datasets are derived by introducing various corruptions to the images in validation sets of the CIFAR10, CIFAR100, and ImageNet datasets. We summarize the corruption types in Fig. 7, including brightness, frosted glass blur, JPEG compression, contrast, defocus blur, impulse noise, motion blur, snow, zoom blur, frost, pixelation, Gaussian noise, elastic transformation, shot noise, and fog. The aim of these corruptions is to mimic a range of natural environmental conditions that may be encountered during deployment.

Although these 15 corruption domains stem from 4 categories of corruptions (weather, noise, blur, digital), there is a noticeable domain shift from the source domain (clean) to each of the 15 corruption domains. In the continual TTA setting (Tab. 4), to create a sequence of 15 continual domain shifts, we mix the 4 categories of corruptions in the target domain sequence.

ImageNet-D/R. ImageNet-D is re-proposed from DomainNet and includes six image styles: Clipart, Real, Infograph, Painting, Quick-draw, and Sketch.

[1] https://github.com/RobustBench/robustbench

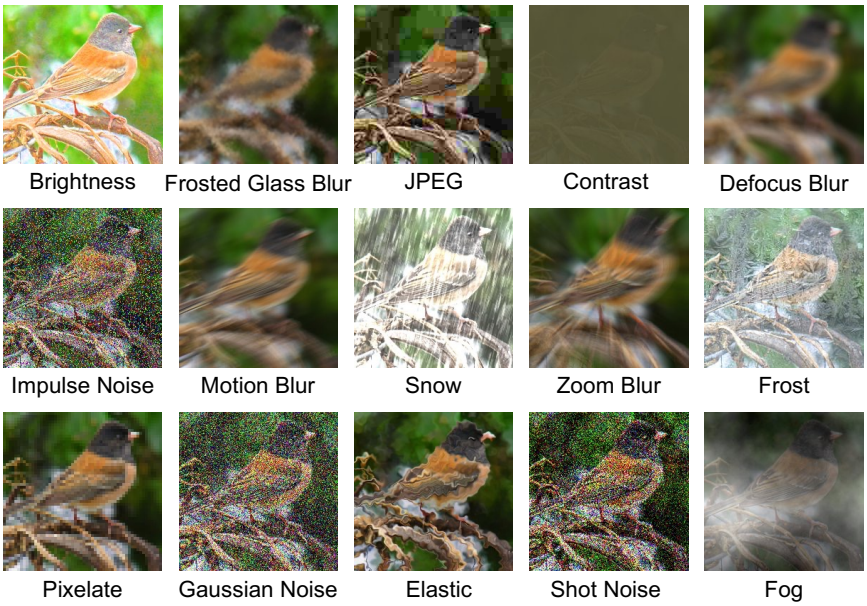


Fig. 7: Examples of corruption types in ImageNet-C.

We do not include Quick-draw as a target domain in the experiment because the source model completely fails in this domain, with an error rate greater than 99% [46], which might be attributed to the noisy labels and classes in this domain[2]. ImageNet-R contains various renditions of 200 ImageNet classes, including art, cartoons, deviantart, graffiti, embroidery, graphics, origami, paintings, patterns, plastic objects, plush objects, sculptures, sketches, tattoos, toys, and video games. Figure 8 exemplifies different renditions of the same class.

C More Implementation Details

C.1 Illustration of Non-I.I.D. Data Streams

As outlined in Sec. 4.2, we use the Dirichlet distribution for generating the non-i.i.d. data streams. We visualize the non-i.i.d. data streams with different Dirichlet parameters δ from the CIFAR10-C dataset in Fig. 9. A lower δ value indicates a higher degree of non-i.i.d. characteristics, manifesting as increased temporal correlation.

[2] Saito, K., Kim, D., Sclaroff, S., Darrell, T., Saenko, K.: Semi-supervised domain adaptation via minimax entropy. In: ICCV. pp. 8050–8058 (2019)

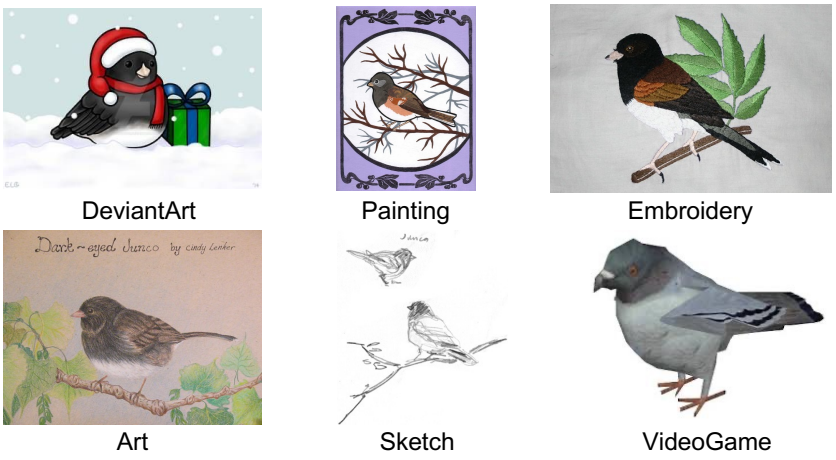


Fig. 8: Different renditions in ImageNet-R.

C.2 Implementation Details on ImageNet-D/R/A

ImageNet-D. The classes in ImageNet-D do not have a one-to-one class mapping relationship with ImageNet. Therefore, we adopt the source model with a 1000-class classifier to directly carry out the TTA process on this dataset. This presents a greater challenge for TTA methods because the model has an additional 537 class options ($1000 - 463 = 537$) that are not present in the target dataset.

ImageNet-R/A. Following [11], we wrap the source model with a 1000-to-200 class mapping mask because ImageNet-R/A comprises 200 classes out of the 1000 from ImageNet, with a one-to-one mapping relationship.

D Hyper-Parameter Details

During the test-time adaptation process, the batch size is consistently set to 64 across all datasets and methods, where applicable. Following MEMO [71], we use the remaining four corruptions, excluding the 15 used for testing in CIFAR10/100-C and ImageNet-C, to select hyper-parameters. We adopt the Adam optimizer and set the learning rate to 1×10^{-4} for WideResNet-28 and ResNeXt-29, and 1×10^{-5} for ResNet-50/18. The parameter α is set to 0.5 for CIFAR10/100-C and 0.75 for ImageNet-C/R/D/A. The confidence threshold θ in EM loss is fixed at 0.8 for all datasets, except for ImageNet-A, where θ is set to 0.4 to account for the significantly decreased confidence in the source model caused by adversarial attacks.

In the continual TTA setting, the hyper-parameters τ, p, and q for the domain shift detection mechanism are fixed at 1.1, 3, and 15, respectively. We set

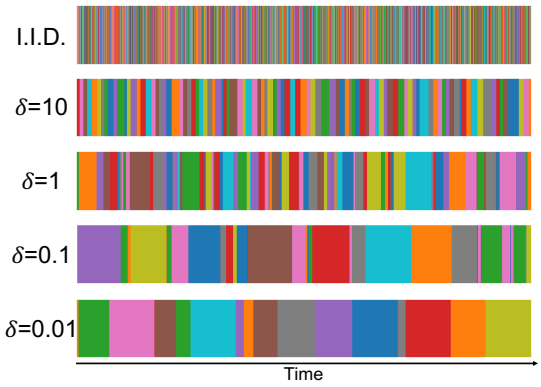
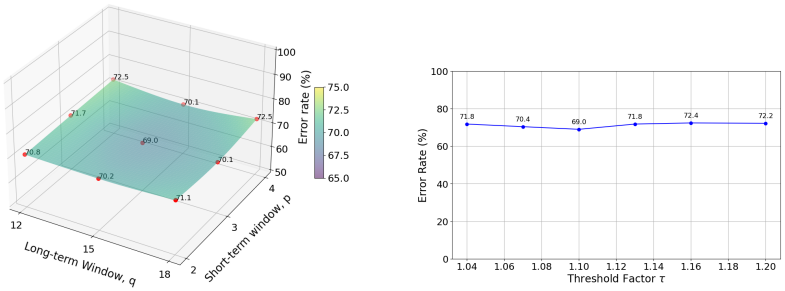


Fig. 9: Illustration of non-independent and identically distributed (non-i.i.d.) data streams with varying Dirichlet parameters δ. Each color represents a different category.



(a) **(b)**

Fig. 10: (a) Influence of varying window sizes (p, q) with the threshold factor fixed at 1.1. (b) Influence of the varying threshold factor τ with window sizes (p, q) fixed at $(3, 15)$.

the batch size to 40 for our method in the experiments. Here, we examine the sensitivity of adaptation performance to the three hyper-parameters (τ, p, q) in the domain shift detection mechanism. Specifically, we investigate the influence of varying window sizes (p, q) with the threshold factor fixed at 1.1 in Fig. 10a. In Fig. 10b, we assess the influence of varying the threshold factor τ with the window sizes (p, q) held constant at $(3, 15)$. The results indicate that the adaptation performance is relatively insensitive to changes in these hyper-parameters of the domain shift detection mechanism within a certain range.

E Source of Experimental Codes

The results of other methods are obtained by running their codes. For NOTE [16] and DELTA [73] methods, we conduct the evaluation experiments based on

their official codes, available at NOTE [3] and DELTA [4] respectively. For other methods, including TTBN [39], MEMO [71], LAME [3], CoTTA [50], EATA [40], RoTTA [69], RMT [11], and SAR [41], we utilize the Online Test-time Adaptation integrated repository,[5] developed by the authors of RMT [11]. Code for our method is available at github.com/WZq975/DA-TTA.

It is worth noting that we use uniform pre-trained source models for our evaluations, which differ from those adopted in the NOTE paper [16], therefore, the reported results vary from those in the NOTE paper. For a faithful comparison, we also evaluate our method using the same pre-trained model (ResNet-18) as used in NOTE, as described in Appendix F.1.

F Additional Results

F.1 Comparison with NOTE and BUFR Using ResNet-18

Table 6: Comparison with NOTE and BUFR using a ResNet-18 pre-trained model. Error rates (in %) are reported. In the table, numbers in red indicate performance degradation after adaptation.

Method	Non-I.I.D			I.I.D		
	CIFAR10-C	CIFAR100-C	ImageNet-C	CIFAR10-C	CIFAR100-C	ImageNet-C
Source	40.6	66.3	85.4	40.6	66.3	85.4
BUFR [12]	76.1	84.2	-	**15.1**	53.0	-
NOTE [16]	**21.1**	47.0	80.6	20.1	46.4	70.3
DA-TTA(ours)	23.1	**46.6**	**72.0**	18.7	**43.8**	**69.3**

We additionally evaluate our method using ResNet-18 as the pre-trained source model, which is also adopted in the NOTE [16] and BUFR [12] papers. Specifically, we utilize ResNet-18 models pre-trained on CIFAR10/100, provided by NOTE, and ResNet-18 pre-trained on ImageNet for evaluations on CIFAR10/100-C and ImageNet-C, respectively. In Tab. 6, the results for NOTE are sourced from its paper, while the results for BUFR are obtained by running its code.[6] BUFR, as an SFDA method, succeeds in adapting to i.i.d. online streams but fails to adapt to non-i.i.d streams. Our method demonstrates better overall performance compared to these two methods.

F.2 Results under I.I.D. Assumption

In this work, we focus on fully test-time adaptation on practical scenarios where the test data stream is non-i.i.d.. Nonetheless, we also provide the evaluation results under i.i.d. assumption in Tab. 7. These results are on the same CIFAR10-C,

[3] https://github.com/TaesikGong/NOTE
[4] https://github.com/bwbwzhao/DELTA
[5] https://github.com/mariodoebler/test-time-adaptation
[6] https://github.com/cianeastwood/bufr

Table 7: Fully test-time adaptation under i.i.d. assumption. We report the average classification error rates (%) for 15 target domains within each of the CIFAR10-C, CIFAR-100-C, and ImageNet-C datasets, respectively.

Method	CIFAR10-C	CIFAR100-C	ImageNet-C
Source	43.5	46.5	82.0
LAME [3]	57.7	73.8	93.4
RoTTA [69]	21.5	42.0	69.3
TTBN [39, 48]	20.8	36.3	68.6
CoTTA [50]	18.5	35.0	68.1
RMT [11]	**17.5**	33.9	63.5
TENT [55]	18.7	32.7	65.3
SAR [41]	20.7	32.5	65.3
EATA [40]	18.2	31.5	**60.4**
Ours	20.0	**31.2**	65.3

CIFAR100-C, and ImageNet-C datasets as those discussed in Tab. 1 for non-i.i.d. streams. The results show that LAME [3] underperforms in i.i.d. streams compared to non-i.i.d. scenarios. Conversely, our method demonstrates consistent accuracy across both i.i.d. and non-i.i.d. streams, ensuring comparable performance under i.i.d. assumption and marked improvements in non-i.i.d. scenarios compared to the state-of-the-art methods.

G Additional Ablation Studies

Table 8: Effects of optimizing the DA loss via updating different layers in the network. We consider four different variants: (A) Optimizing all the convolution blocks before the classifier. (B) Optimizing the last convolution block before the classifier. (C) Optimizing additional affine layers before BN layers. (D) optimizing additional affine layers after BN layers.

Variant	CIFAR10-C	CIFAR100-C	ImageNet-C
A	90.5	98.7	95.4
B	56.7	98.6	99.9
C	24.4	32.1	67.7
D	**24.3**	31.8	65.7
Ours	**24.3**	**31.6**	**64.8**

The objective of the (Distribution Alignment) DA loss is to recalibrate the test-time features, aligning them closer to the source domain. This alignment helps mitigate domain shift, thereby enhancing performance. Specifically, we focus on optimizing the affine layers within Batch Normalization (BN) layers. The rationale is that these affine layers can *linearly* adjust the statistical properties (mean and variance) of the feature distributions. This enables transformation

of the distributions while preserving the representational capacity of the source model's convolution layers. Despite this focus, we also conduct an ablation study exploring the effects of applying DA loss optimization to convolution layers. As shown in Tab. 8, when optimizing all convolution blocks or the last convolution block before the classifier with DA loss (variants A and B), we can observe a detrimental effect on performance, contravening the DA loss's objective.

In our method, optimizing affine layers in BN layers aims to steer the feature distributions closer to those of the source domain. Theoretically, however, these affine layers are not confined to BN layers alone. To verify this, we introduce two additional variants, (C) and (D), incorporating extra affine layers into the source model. In these variants, all source model parameters, including the BN layers, remain frozen. The results, detailed in Tab. 8, reveal that both variants achieve performance comparable to our final version.

H Limitations

Like most existing fully TTA methods, DA-TTA needs to continually optimize the source model with online data streams. This might be impractical for applications on edge devices, given that current foundation models are becoming larger and larger. Furthermore, in certain industry scenarios, source models cannot be modified and are provided only as a black box. In such cases, modifications in intermediate BN layers are not feasible. Other model reprogramming operations should be considered to address this issue.