Maintaining Consistent Inter-Class Topology in Continual Test-Time Adaptation

Chenggong Ni[1*] Fan Lyu[2*] Jiaoyao Tan[1] Fuyuan Hu[1†] Rui Yao[3] Tao Zhou[4]
[1]School of Electronics and Information Engineering, Suzhou University of Science and Technology
[2]New Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences
[3]School of Computer Science and Technology, University of Mining and Technology
[4]School of Computer Science and Technology, North Minzu University, Yinchuan
{cgn@post, jiaoyaotan@post, fuyuanhu@mail}.usts.edu.cn
fan.lyu@cripac.ia.ac.cn, ruiyao.contact@gmail.com, zhoutaonxmu@126.com

Abstract

This paper introduces Topological Consistency Adaptation (TCA), a novel approach to Continual Test-time Adaptation (CTTA) that addresses the challenges of domain shifts and error accumulation in testing scenarios. TCA ensures the stability of inter-class relationships by enforcing a class topological consistency constraint, which minimizes the distortion of class centroids and preserves the topological structure during continuous adaptation. Additionally, we propose an intra-class compactness loss to maintain compactness within classes, indirectly supporting inter-class stability. To further enhance model adaptation, we introduce a batch imbalance topology weighting mechanism that accounts for class distribution imbalances within each batch, optimizing centroid distances and stabilizing the inter-class topology. Experiments show that our method demonstrates improvements in handling continuous domain shifts, ensuring stable feature distributions and boosting predictive performance.

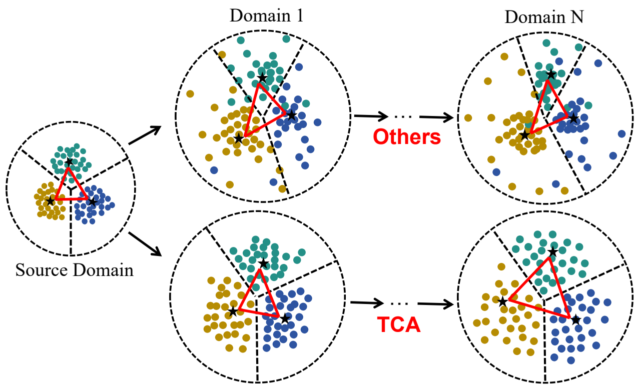

Figure 1. Feature distribution in CTTA. In other methods, as domain shifts occur continuously, the latent inter-class topological relationships learned by the model in the source domain are progressively distorted. Throughout the adaptation process, Topological Consistency Adaptation (TCA) maintains a stable inter-class topological structure.

1. Introduction

Test-time adaptation (TTA) [1, 13, 35] aims to adapt a source pre-trained model by learning from the unlabeled test (target) data during inference time, where the training data cannot be disclosed due to data privacy concerns. Recently, considering that the data distribution may change continuously, Continual Test-Time Adaptation (CTTA) [30] is proposed as an online continuous form of TTA. CTTA aims to continuously adapt models to the evolving data distribution of target domains without relying on source data. A key challenge in the CTTA method within the self-training framework is the accumulation of errors, which arises from pseudo-labels generated by the model's predictions on unlabeled data. To solve the problem, existing methods such as [8, 30, 37] primarily focus on optimizing the self-training framework to prevent error accumulation.

However, the existing methods cannot maintain stable inter-class topological structures in the feature space. As shown in Fig 1, unsupervised self-training can lead to an accumulation of errors due to unstable feature representations. This instability causes the model to produce erroneous pseudo-labels [6, 34], which, in turn, misguide the model into biased updates. This cycle perpetuates, progressively impairing model performance and resulting in error accumulation. There are two primary causes for this phenomenon. On one hand, the balance of inter-class features is disrupted, breaking down the stable topological structures between classes initially learned in the source domain. On the other hand, the uniformity of intra-class feature distribution deteriorates, leading to increasingly indistinct classification boundaries. As this process continues, the accumulation of errors within the model intensifies, further compromising its performance.

To break the vicious cycle, in this paper, we propose to achieve sustained and stable representational capacity by encouraging the output of more stable and uniform feature distributions. Specifically, we propose Topological Consistency Adaptation (TCA). We propose the class topological consistency constraint to maintain inter-class topology stability from both inter-class and intra-class perspectives. For inter-class relationships, we introduce the inter-class uniformity loss, which uses class centroids to prevent topology collapse during continuous domain shifts [3, 10, 23]. For intra-class relationships, we ensure reasonable compactness of intra-class features, indirectly supporting inter-class stability. Additionally, we design a batch imbalance topology weighting that accounts for class distribution imbalances within each batch, further optimizing centroid distances and ensuring inter-class topology stability. Extensive experiments demonstrate that our method effectively reduces error accumulation and improves the generalization performance in the CTTA tasks.

Our contributions are as follows:

- We propose a class topological consistency constraint that ensures stable inter-class topology by maintaining uniformity in class centroids, preventing topology collapse during continual domain shifts.
- We introduce an intra-class compactness loss to indirectly support inter-class stability, ensuring reasonable compactness of features within each class while stabilizing class boundaries.
- We design a batch imbalance topology weighting that accounts for class distribution imbalances in each batch, optimizing centroid distances to preserve inter-class topology stability in the face of imbalanced data.

2. Related Work

2.1. Continual Test-Time Adaptation

The goal of Continual Test-Time Adaptation (CTTA) [14, 16, 27, 30] is to enable a pre-trained source model to continuously adapt to evolving target domains [11, 19] using unlabeled data from these new environments, thereby alleviating the performance degradation that typically arises from domain shifts [17, 26]. CoTTA [30] was the first to introduce continuous domain variation into Test-Time Adaptation (TTA), establishing a weighted-averaged teacher-student framework to address the challenge of error accumulation caused by ongoing domain shifts. Building on this foundation, PETAL [2] further develops a data-driven parameter recovery technique to better align the model with its original source configuration in the parameter space, thereby enhancing adaptation robustness. AdaContrast [5] leverages contrastive learning with online pseudo-label refinement to improve feature representations, reducing the impact of noisy pseudo-labels on adaptation quality. RMT [8] introduces a robust mean teacher framework that addresses the issue of repeated domain shifts by maintaining stability across multiple adaptation phases, while GTTA [18] uses techniques like mixup and style-transfer to artificially create intermediate domains, smoothing the transition between domains. Additionally, CRG [40] restructures the online data buffering and organization mechanisms for CTTA, designing a class relationship graph to preserve intra-class semantic relationships during domain changes.

2.2. Feature Uniformity and Alignment

Unsupervised contrastive representation learning has seen remarkable success in learning representations for image and sequential data [9, 15, 24], with [32] identifying two key properties related to the contrastive loss: alignment and uniformity, the latter of which motivated us to exploit the rich intra-class information preserved by self-supervised learning to enhance generalization performance. [32] demonstrated that directly optimizing these two metrics in unsupervised comparison learning leads to better representational capabilities. SEAT [39] aligns pixel-wise embeddings with semantic embeddings, dynamically aggregates local features of pixels and global semantic features of batches, combines orthogonality constraints with uniformity to achieve a uniform distribution of semantic embeddings. In the TTA scenario, [31] draws on this idea by designing a memory bank to assist the model in aligning the prior distributions, indirectly maintaining a more homogeneous feature distribution by constraining the predictive consistency of both the prototype and linear classifiers. In contrast, our approach directly optimizes the distribution of the feature space by maintaining inter-class feature distances and intra-class compactness, thus maintaining a uniform distribution of features in the space.

3. Methodology

3.1. Problem definition and Overall Architecture

Given a pre-trained model g_θ that parameterized on the source data $(\mathcal{X}^{\mathrm{S}}, \mathcal{Y}^{\mathrm{S}})$, CTTA aims to make the model in response to the continually changing target domain $\mathcal{X}^D$ during the test phase, without accessing any source data, here we simplify $\mathcal{X}^D$ to $\mathcal{X}$. Let the testing model be $g = f \circ h$, where f denotes the backbone and h denotes the linear classification head. With the model, given a batch of unlabeled samples, we can generate the corresponding image embeddings $z = f(x)$, logits $p = h(z)$. Following [30], we build our method on the MT framework and initialize two models, *i.e.*, the teacher model g_t and the student model g_s.

The overall framework of TCA shown in Fig. 2, comprises two main components: the **class topological consis-**

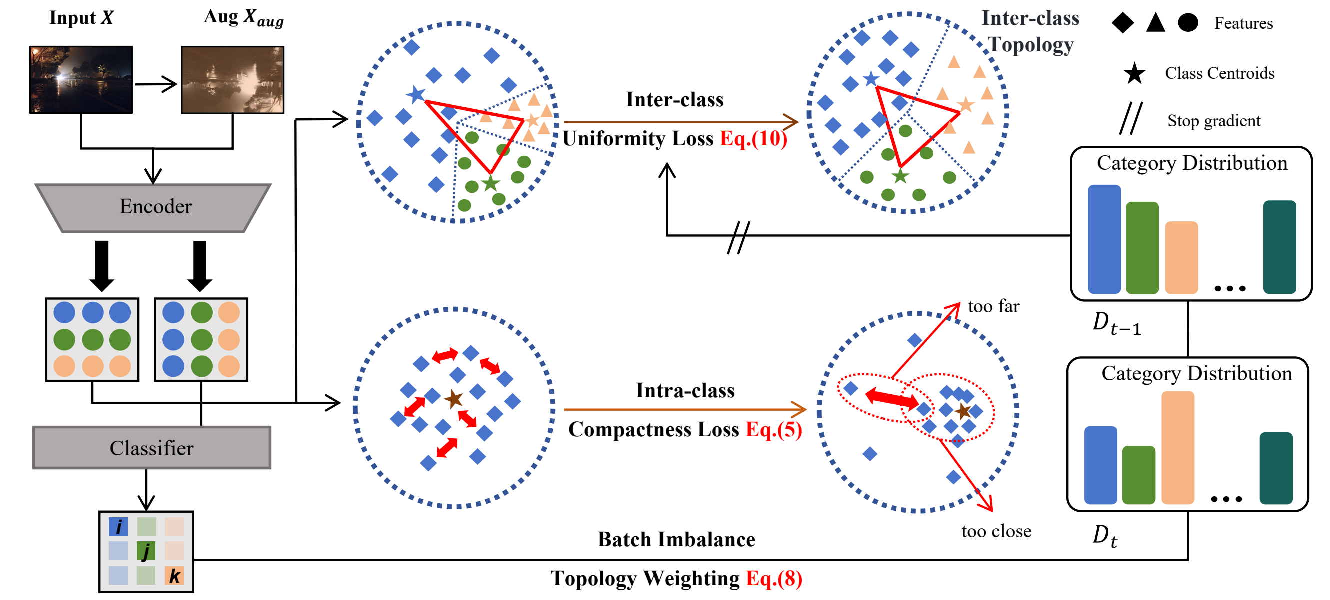

Figure 2. Framework overview. TCA initially focuses on the uniformity of inter-class feature distribution, utilizing enhanced pseudo-label prediction to compute pseudo-centroid proxies, thereby uniformizing the inter-class features. Subsequently, TCA maintains a compact distribution of intra-class features, mitigating the imbalance within the class feature distribution. Finally, TCA continuously maintains dynamic weights of inter-class centroids based on detailed historical prediction distributions, preserving the latent topological relationships between classes.

tency constraint method and the **batch imbalance topology weighting** strategy. The class topological consistency constraint introduces an inter-class uniformity loss and an intra-class compactness loss to maintain a stable and consistent inter-class topology across the evolving target domain. Moreover, since each batch contains a highly imbalanced amount of data for each class, the constructed inter-class topology is unstable, we design a batch imbalance topology weighting method to ensure consistency in inter-class distribution.

3.2. Constructing Inter-Class Topological Graph

Class topological relationships refer to the relative positions and distances between different classes within the feature space. In the CTTA scenario, models tend to accumulate errors, leading to semantic confusion between different classes and resulting in incorrect pseudo-labels. As illustrated in Fig 1, this phenomenon exacerbates with domain shifts. To mitigate this issue, we propose to maintain the stable and uniform inter-class topological relationships learned in the source domain during testing.

Specifically, we utilize class centroids as proxies for the classes and centroid distances as proxies for inter-class topological distances, defining the inter-class topological graph $\mathcal{G} = \langle \mathcal{V}, E \rangle$, where $\mathcal{V} = \{\mathbf{v_1}, \mathbf{v_2}, \dots, \mathbf{v_k}\}$ is the set of centroids and $E = \{(\mathbf{v_i}, \mathbf{v_j}) \mid w_{ij} = \|\mathbf{v_i} - \mathbf{v_j}\|_2, \forall i, j \in \{1, \dots, k\}, i \neq j\}$ represents the edges capturing the neighborhood relationships between these centroids. Each edge e_{ij} is assigned a weight w_{ij}, which corresponds to the L_2 distance between centroids $\mathbf{v}_i$ and $\mathbf{v}_j$. At time t, we use random augmented features [25] of the current batch to improve the reliability of class centroids. This alleviates the instability of class centroids caused by limited samples or distribution shifts, resulting in more stable and reliable pseudo-labels [8]. The class centroid can be computed as:

$$\hat{y}_i = \operatorname{argmax}(g_s(\operatorname{Aug}(x_i))), \tag{1}$$

$$\mathbf{v}_k^{(t)} = \frac{1}{|\mathcal{I}_k|} \sum_{i=1}^{B} \mathbb{I}(\hat{y}_i = k) \cdot f_s(\operatorname{Aug}(x_i)), \tag{2}$$

where $\operatorname{Aug}(\cdot)$ denotes data augmentation, for which we adhere to the unified augmentation procedures outlined. B denotes the batch size. $\mathbb{I}(\cdot)$ is an indicator function. $\mathcal{I}_k$ denotes the set of samples belonging to class k, and $|\mathcal{I}_k|$ represents the number of samples in class k.

In each batch, the inter-class topology includes all classes, and this topology is continuously updated throughout the testing process. Specifically, we initialize the topological graph $\mathcal{G}$ as an empty graph, which will be iteratively populated and updated with class centroids as new batches are processed, and the centroid update relies not only on the current batch but also incorporates the centroid information from the previous batch. The specific calculation method is as follows.

$$\mathbf{v}_k^{(t)} = \alpha \cdot \mathbf{v}_k^{(t-1)} + (1-\alpha) \cdot \mathbf{v}_k, \tag{3}$$

where α is a weighting parameter that controls the influence of previous batch centroids. Eq. (3) allows the model to adapt to the distribution shift in each batch gradually and progressively cover information from all classes. This strategy ensures a stable inter-class topology across varying batch sizes and imbalanced data distributions, leveraging historical information when some classes are absent. For incomplete initial class coverage, the weighted updates gradually build a complete topology. Even in single-sample (batch size $= 1$) online scenarios, we can use the current sample's feature as a proxy for the class centroid update.

At time t, given the $\mathcal{G}_t = \langle \mathcal{V}_t, E_t \rangle$, when the domain changes, the topological structure of $\mathcal{G}_t$ may be distorted, and the distribution in the feature space becomes non-uniform. Thus, inter-class features become less separable, causing decision boundaries to blur, while intra-class distributions become imbalanced, resulting in outliers and feature clustering. These distortions drive the model to generate incorrect pseudo-labels due to unstable feature representations, ultimately leading to an accumulation of errors.

3.3. Class Topological Consistency Constraint

To solve the unstable problem of the topology, we design a class topological consistency constraint method. Our goal is to maintain a uniform topological relationship across all classes. We define the inter-class uniformity loss as the logarithm of the average Gaussian potential of node pairs:

$$\mathcal{L}_{\text{inter}} = \log \left(\frac{2}{|\mathcal{V}|(|\mathcal{V}|-1)} \sum_{\mathbf{v}_i, \mathbf{v}_j \in V,\, i<j} e^{-tw_{ij}^2} \right), \tag{4}$$

where $|\mathcal{V}|$ denotes the number of nodes in the graph $\mathcal{G}$ and t is a fixed parameter set to 2. $\mathcal{L}_{\text{inter}}$ aims to ensure a uniform distribution of centroids by minimizing the average exponential decay of their pairwise distances. It pushes centroids of different categories apart, reduces the propagation of inter-class errors, and maintains stable inter-class distances, thereby preserving a coherent and consistent topological structure.

The stability of inter-class topology depends on both inter-class distribution and intra-class feature concentration. Overly concentrated features can ignore subclass variations and noise, while dispersed features hinder stable intra-class learning, as shown in Fig. 1. Therefore, to further maintain a stable inter-class topological structure, we suggest further compacting and uniformizing intra-class features. Specifically, we apply pairwise uniformization to the intra-class feature pairs within $\mathcal{V}$ to achieve a more compact and uniform inter-class feature distribution. The intra-class compactness loss is defined as:

$$\mathcal{L}_{\text{intra}} = \log \mathbb{E}_{\mathbf{v}_k \in \mathcal{V}} \left[e^{-t\mathbb{E}_{z_i, z_j \sim \mathbf{v}_k} \left(\|z_i - z_j\|^2 \right)} \right], \tag{5}$$

where t is a fixed parameter set to 2. Unlike traditional contrastive learning like [32], which directly applies uniformization to all features, our approach separately considers inter-class and intra-class features, making it more suitable for dynamic domain adaptation scenarios. Without this separation, uniformizing all features together could lead to a distortion of the inter-class topological structure, as inter-class boundaries may become less distinct. This could result in degraded performance, especially in dynamic environments where class distributions shift over time.

Then, by combining Eq. (4) and Eq. (5), we obtain the uniformity loss that maintains a stable inter-class topological structure:

$$\mathcal{L}_{\text{topology}} = \mathcal{L}_{\text{inter}} + \mathcal{L}_{\text{intra}}. \tag{6}$$

Our method encourages uniform feature distributions, minimizing distortion in the inter-class topology. This sharpens class boundaries, stabilizes topological weights, and enhances centroid accuracy, improving the precision of topological vertices. The effectiveness of the intra-class compactness loss relies on a stable inter-class topology. Without Eq. (4), directly optimizing Eq. (5) could further disrupt the inter-class structure.

3.4. Batch Imbalance Topology Weighting

However, the proposed method overlooks the batch imbalance in inter-class distribution present in real testing scenarios. Different batches may contain different classes, and the sample sizes of these classes can vary. Without labels, this imbalance can gradually disrupt the inter-class topology, leading to a feature distribution that appears uniform but is actually highly imbalanced, thereby affecting the model's predictive performance.

To avoid this, we combine the inter-class topological distances with the true class distribution information from the model output to design a batch imbalance topology weighting (BTW) matrix. Specifically, we design a dynamic weighting matrix for the nodes in $\mathcal{G}$, applying the weights to the edges between nodes. These weights are determined by the frequency of each category in the current batch, enabling the edge distances in $\mathcal{G}$ to be adaptively adjusted based on the true category distribution. In the current batch, the frequency μ_k of class k is calculated by normalizing the corresponding number of instances:

$$\mu_k = \frac{\sqrt{|\mathcal{I}_k|}}{\sum_{k=1}^{C} \left(\sqrt{|\mathcal{I}_k|} + \epsilon \right)}, \tag{7}$$

where ϵ is a small constant used to avoid division by zero. We define the weighting term μ_{ij} for the edge $(\mathbf{v}_i, \mathbf{v}_j)$ as:

$$\mu_{ij} = \frac{\mu_i + \mu_j}{2}. \tag{8}$$

To ensure that the weights reflect the model's most recent representational capabilities, we continuously maintain the weighting matrix using EMA [20] to preserve the diversity information of the prior distribution:

$$\mu_{ij}^{(t)} = \beta \mu_{ij}^{(t-1)} + (1-\beta)\mu_{ij}. \tag{9}$$

We apply a gradient stopping operation to μ and integrate the weighting factor μ_{ij} to $\mathcal{L}_{\text{inter}}$ to recalculate inter-class uniformity loss:

$$\mathcal{L}'_{\text{inter}} = \log\left(\frac{\sum_{\mathbf{v}_i,\mathbf{v}_j \in \mathcal{V},\, i<j} \mu_{ij} \cdot e^{-tw_{ij}^2}}{\sum_{\mathbf{v}_i,\mathbf{v}_j \in \mathcal{V},\, i<j} \mu_{ij}}\right). \tag{10}$$

Eq. (10) fully considers the class imbalance in each batch by dynamically adjusting inter-class weights, effectively maintaining the stability of the inter-class topological structure. In imbalanced scenarios, BTW calculates weights based on the frequency of each class in the current batch. On the one hand, this allows the model to more accurately preserve inter-class relative distances when adapting to new data distributions. On the other hand, as previously mentioned, BTW can handle extreme cases, such as when batch size equals 1 or when all samples in the current batch belong to the same class.

3.5. Optimization Objective and the Algorithm

At time step t, the teacher model g_t generates pseudo-labels to help the learning process of the student model g_s. Specifically, we compute the symmetric cross-entropy loss (SCE) [8] between the outputs of the student model and the pseudo-labels. The self-training loss can be formulated as follows:

$$\mathcal{L}_{\text{SCE}} = -\sum_{c=1}^{C} g_t(x)\log g_s(x) - \sum_{c=1}^{C} g_s(x)\log g_t(x). \tag{11}$$

Following [32], we align the feature distributions between positives while maintaining feature uniformity:

$$\mathcal{L}_{\text{align}}(f;\alpha) \triangleq -\mathbb{E}_{(x,y)\sim p_{\text{pos}}}\left[\|f(x)-f(y)\|_2^2\right], \tag{12}$$

where x and y represent features obtained from inputs subjected to different forms of data augmentation. Finally, the total loss function $\mathcal{L}$ can be formulated as follows:

$$\mathcal{L} = \mathcal{L}_{\text{SCE}} + \lambda_1 \cdot \mathcal{L}_{\text{align}} + \lambda_2 \cdot \mathcal{L}_{\text{topology}}. \tag{13}$$

As shown in the Algorithm 1, given a batch of testing data, we first propose the class topological consistency constraint, which maintains the stability of inter-class topology

Algorithm 1 Topological Consistency Adaptation (TCA).

```
Input: Pre-trained model $g = f \circ h$, self-training loss $\mathcal{L}_{\text{SCE}}$, Alignment Loss weight $\lambda_1$, Uniformity Loss weight $\lambda_2$.
 1: for Each Batch do
 2:     Get unsupervised data $x$
 3:     Augment $x$ to construct the class topological graph $\mathcal{G}$ using Eq. (3)
 4:     if Batch $= 1$ then
 5:         Initializing inter-class uniformity $\mathcal{L}_{\text{inter}}$ Eq. (4)
 6:         Compute intra-class compactness $\mathcal{L}_{\text{intra}}$ Eq. (5)
 7:         return $\mathcal{L}_{\text{topology}} = \mathcal{L}_{\text{inter}} + \mathcal{L}_{\text{intra}}$
 8:     else
 9:         Compute category weights $\mu_{ij}$ Eq. (8)
10:         Update $\mathcal{L}_{\text{inter}}$ to $\mathcal{L}'_{\text{inter}}$ Eq. (10)
11:         return $\mathcal{L}_{\text{topology}} = \mathcal{L}'_{\text{inter}} + \mathcal{L}_{\text{intra}}$
12:     end if
13:     Compute $\mathcal{L} = \mathcal{L}_{\text{SCE}} + \lambda_1 \cdot \mathcal{L}_{\text{align}} + \lambda_2 \cdot \mathcal{L}_{\text{topology}}$ Eq. (13)
14:     Model $g$ is updated using $\mathcal{L}$
15: end for
```

from both inter-class and intra-class perspectives. Specifically, for inter-class relationships, we establish the inter-class uniformity loss by calculating class centroids to ensure that the inter-class topology does not collapse during continuous changes in the target domain. For intra-class relationships, we ensure the reasonable compactness of intra-class features to indirectly support the stability of inter-class relationships. Next, we design batch imbalance topology weighting, which comprehensively considers the class distribution imbalance in each batch to further optimize the distances between class centroids, ensuring the stability of the inter-class topology.

4. Experiments

4.1. Set Up

4.1.1. Dataset and Settings

We conduct extensive experiments to demonstrate the effectiveness of our approach. We evaluate TCA on three benchmark tasks for continual test-time adaptation in image processing: CIFAR-10-C, CIFAR-100-C, and ImageNet-C. These tasks are designed to assess the robustness of machine learning models to corruptions and disturbances in the input data.

CIFAR10-C extends the CIFAR-10 dataset, comprising 32 × 32 color images from 10 classes. It includes 15 different corruptions, each at five severity levels, applied to the test images of CIFAR-10 [12], resulting in a total of 10,000 images.

CIFAR100-C extends the CIFAR-100 [12] dataset, con-

	Method	Gau.	shot	imp.	def.	glass	mot.	zoom	snow	fro.	fog	bri.	con.	ela.	pix.	jpeg	Mean
CIFAR10-C	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
	TENT [29]	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
	Ada [5]	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 [30]	24.3	21.3	26.6	11.6	27.6	12.2	10.3	14.8	14.1	12.4	7.6	10.6	18.3	13.4	17.3	16.2
	RMT [8]	24.0	20.4	25.6	12.6	25.4	14.2	12.2	15.4	15.1	14.1	10.3	13.7	17.1	13.5	16.0	16.7
	DSS [33]	24.1	21.3	25.4	11.7	26.9	12.2	10.5	14.5	14.1	12.5	7.8	10.8	18.0	13.1	17.3	16.0
	BeCoTTA [14]	22.9	**19.1**	26.9	**10.2**	27.5	12.7	10.4	14.7	14.3	12.4	**7.2**	**9.4**	20.9	15.2	20.2	16.3
	TCA	**22.4**	19.2	**23.0**	10.8	**23.2**	**11.6**	**9.9**	**13.1**	**13.1**	**11.8**	7.6	10.7	**16.4**	**12.0**	**15.5**	**14.7**
CIFAR100-C	Source only	73.0	68.0	39.4	29.3	54.1	30.8	28.8	39.5	45.8	50.3	29.5	55.1	37.2	74.7	41.2	46.4
	TENT [29]	**37.2**	**35.8**	41.7	37.9	51.2	48.3	48.5	58.4	63.7	71.1	70.4	82.3	88.0	88.5	90.4	60.9
	Ada [5]	42.3	36.8	38.6	27.7	40.1	29.1	27.5	32.9	30.7	38.2	25.9	28.3	33.9	33.3	36.2	33.4
	CoTTA [30]	40.1	37.7	39.7	26.9	38.0	27.9	26.4	32.8	31.8	40.3	24.7	26.9	32.5	28.3	33.5	32.5
	RMT [8]	40.5	36.1	**36.3**	27.7	**33.9**	28.5	26.4	**29.0**	29.0	32.5	25.1	27.4	28.2	**26.3**	**29.3**	30.4
	DSS [33]	39.7	36.0	37.2	26.3	35.6	27.5	25.2	31.4	30.0	37.8	24.2	26.0	30.0	26.3	31.3	30.9
	BeCoTTA [14]	42.1	38.0	42.2	30.2	42.9	31.7	29.8	35.1	33.9	38.5	27.9	32.0	36.7	31.6	39.9	35.5
	TCA	38.5	36.0	36.6	**25.8**	34.6	**27.2**	**25.1**	30.5	**27.0**	**30.1**	**24.1**	**25.7**	**27.3**	26.6	30.3	**29.7**
ImageNet-C	Source only	97.8	97.1	98.2	81.7	89.8	85.2	78.0	83.5	77.1	75.9	41.3	94.5	82.5	79.3	68.6	82.0
	TENT [29]	81.6	74.6	72.7	77.6	73.8	65.5	**55.3**	61.6	63.0	51.7	38.2	72.1	50.8	47.4	53.3	62.6
	Ada [5]	82.9	80.9	78.4	81.4	78.7	72.9	64.0	63.5	64.5	53.5	38.4	66.7	54.6	49.4	53.0	65.5
	CoTTA [30]	84.7	82.1	80.6	81.3	79.0	68.6	57.5	60.3	60.5	48.3	36.6	66.1	47.3	41.2	46.0	62.7
	RMT [8]	80.2	76.4	74.5	77.1	74.4	66.2	57.6	57.0	**59.1**	48.0	39.1	60.6	47.3	42.5	**43.4**	60.2
	DSS [33]	82.3	78.4	76.7	81.9	77.8	66.9	60.9	**50.8**	60.9	**47.7**	**35.4**	69.0	47.5	**40.9**	46.2	62.2
	BeCoTTA [14]	84.1	74.3	**72.2**	77.4	**71.9**	63.4	**55.1**	57.2	61.2	50.7	36.4	66.1	49.2	45.6	48.4	60.9
	TCA	**78.3**	**71.8**	73.5	**74.4**	73.5	**63.3**	56.5	56.9	59.4	48.1	39.6	**59.6**	**47.2**	42.9	44.7	**59.3**

Table 1. Classification error rate (%) on CIFAR10-to-CIFAR10-C, CIFAR100-to-CIFAR100-C, and ImageNet-to-ImageNet-C. All results are evaluated with the largest corruption severity level 5 in an online manner. We report the performance of our method averaged over 5 runs. Bold text indicates the best.

Method	CCC-Easy	CCC-Medium	CCC-Hard	Average
CoTTA [30]	14.9±0.88	7.7±0.43	1.1±0.16	7.9
ETA [21]	41.4±0.95	1.1±0.43	0.2±0.05	14.2
EATA [21]	48.2±0.60	35.4±1.02	8.7±0.80	30.8
SANTA [4]	47.8±0.46	32.7±0.80	9.1±0.60	29.9
RDumb [22]	**49.3±0.88**	38.9±1.40	9.6±1.60	32.6
TCA	49.1±0.35	**39.5±0.53**	**10.1±0.22**	**32.9**

Table 2. Classification accuracy (%) on CCC-benchmark, which is a long-sequence tasks. We report the performance of our method averaged over 5 runs. Bold text indicates the best.

taining 32 × 32 color images from 100 classes. It includes 15 different corruptions, each at five severity levels, applied to the test images of CIFAR-100, resulting in 10,000 images in total.

ImageNet-C extends the ImageNet [12] dataset, which comprises over 14 million images across more than 20,000 categories. ImageNet-C includes 15 different corruptions, with each corruption having five severity levels. These corruptions are applied to the validation images of ImageNet.

CCC-benchmark is a much larger dataset that first features smooth transitions from one domain to another, mimicking how environments change in reality. It consists of 3 difficulty levels, each with 15 types of corruption, 3 random seeds, and 3 levels of transition speed. Each difficulty level includes 9 combinations for continual domain testing across 15 distinct domains, and every single combination contains 7, 500, 000 images.

4.1.2. Baseline and Implementation details

We strictly adhere to the CTTA setup, where no source data is accessed. All models, including CoTTA, TENT continual, AdaContrast, and DSS, are evaluated online, based on a maximum corruption severity level of five across all datasets. Model predictions are first generated before adapting to the current test stream. Similar to CoTTA, we employ standard pre-trained WideResNet [38], ResNeXt-29 [36], and ResNet-50 [7] as the source models for CIFAR10-C, CIFAR100-C, and ImageNet-C. We weight the losses using $\lambda_1 = 0.025$ and $\lambda_2 = 0.15$.

4.2. Main Results

Table 1 shows the classification task results of various methods, including entropy regularization, unsupervised contrastive learning, self-training, and pseudo-label filtering, across three standard datasets. Directly testing the source model on target domains in sequence yields high average errors of 43.5%, 46.4%, and 77.2% on CIFAR10-C, CIFAR100-C, and ImageNet-C respectively. Although the TENT-based method aids in sequential adaptation to the target domain, it may suffer from the accumulation of errors over time. The TENT-based method results in a substantially higher error rate of 60.9% in the long run on CIFAR100-C. CoTTA [30] significantly improves the quality of pseudo-labels by employing averaged augmentation outputs, establishing a standard baseline for the CTTA method. RMT[8] employs a symmetric cross-entropy optimization self-training framework with better gradient properties, achieving some progress on CTTA. DSS [33] notes the existence of high- and low-quality samples in the data stream, but it still does not consider the model's varying trends in representing data of different qualities. BeCoTTA [14] considers the challenges of practical model deployment, improving computational efficiency at the cost of some performance loss. In contrast, our method outperforms all previous competing methods and reduces the average error rate to 14.7%,29.7% and 59.3%, respectively. This indicates that maintaining a balanced and stable inter-class topology and uniformity of intra-class features effectively mitigates the issue of error accumulation in CTTA.

Table 2 presents a comparison of the accuracy of TCA with other methods in long-sequence tasks. TCA achieves impressive performance with accuracies of 49.1%, 39.5%, and 10.1% across three difficulty settings. This demonstrates that TCA effectively addresses complex and challenging long-sequence tasks by maintaining the latent topological relationships between classes.

4.3. Batch size for TCA

As shown in Table 3, we demonstrate that our method remains effective even with a small batch size. Specifically, when the batch size is 1, and the nodes in the class topological graph are not fully updated, the feature of the current test sample serves as the proxy for the class centroid. In this case, the intra-class loss $\mathcal{L}_{\text{intra}} = 0$, and the inter-class loss $\mathcal{L}_{\text{inter}}$ is calculated based on the centroids updated after each iteration. This aligns with the logic of the testing scenario. Similarly, this approach can also address the class imbalance caused by a small batch size. The design of Batch Imbalance Topology Weighting (BTW) further considers inter-class imbalance, assigning lower weights to classes with fewer samples to prevent the occurrence of excessive outliers.

4.4. Uniformity Analysis

4.4.1. Inter-class Uniformity

We compare the representational capabilities of TCA and CoTTA during the CTTA process on Cifar10-C. To demonstrate the overall effectiveness of the algorithms, we visualize the features of ten random batches from three domain distributions at the beginning, middle, and end of the model testing phase. In accordance with the UA [32] evaluation criteria, We plot feature distributions with Gaussian kernel density estimation (KDE) in $\mathbb{R}^2$ and von Mises-Fisher

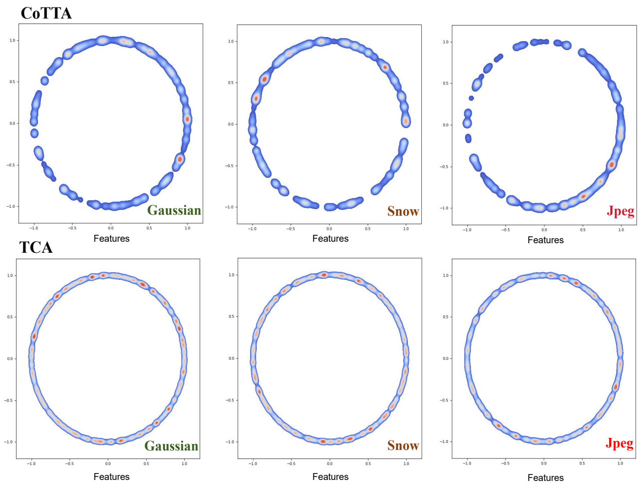

Figure 3. Visualization of features from ten random batches of Cifar10 under three noise distributions: gaussian, snow, and jpeg. The upper image represents CoTTA, while the lower image depicts TCA.

(vMF) KDE on angles (i.e., $\arctan2(y, x)$ for each point $(x, y) \in S^1$). As illustrated in Fig. 3, compared to CoTTA, TCA exhibits a more uniform inter-class feature distribution, maintaining a stable and even inter-class topological relationship.

4.4.2. Intra-class Uniformity

We further compare the intra-class representational capabilities of the two methods. We randomly select features from three classes in a specific domain during the testing phase, visualizing the middle ten batches. As shown in Fig. 4, TCA further maintains the compactness and balance of intra-class features, alleviating the blurring of decision boundaries caused by uneven feature distribution.

4.5. t-SNE Analysis

We employ t-SNE [28] for dimensionality reduction to visually illustrate the model representations of different methods during the testing phase. As shown in Fig 5, We compare the feature distributions of three methods on CIFAR10-C and the feature distribution of the model on the source domain. In CoTTA and RMT, the feature groups exhibit highly uneven distributions, with some class features located outside the main feature clusters, while others are densely concentrate in certain areas of the feature space. This distribution is almost entirely inconsistent with the potential inter-class topology of the model in the source domain, leading to deteriorating representation quality and error accumulation. In contrast, the TCA method achieves a more uniform distribution of inter-class features in the feature space, with intra-class features remaining compact and avoiding excessive overlap, ultimately preserving a similar

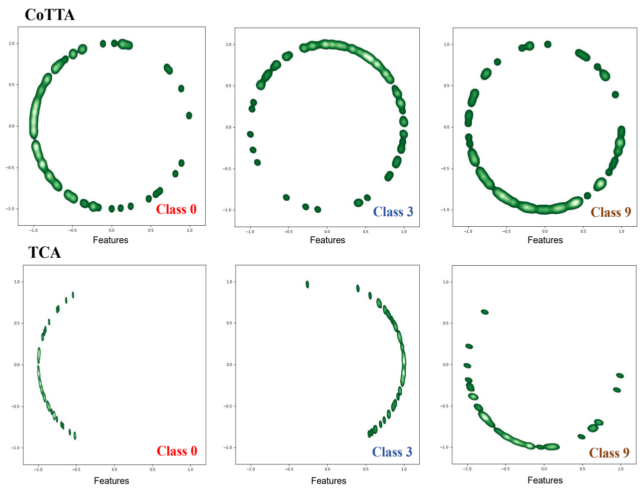

Figure 4. Visualization of intra-class feature distribution in CIFAR-10 under elastic noise conditions involved selecting class 0, class 3, and class 6 (from left to right) and randomly visualizing 10 batches for each. The upper figure illustrates CoTTA, whereas the lower figure depicts TCA.

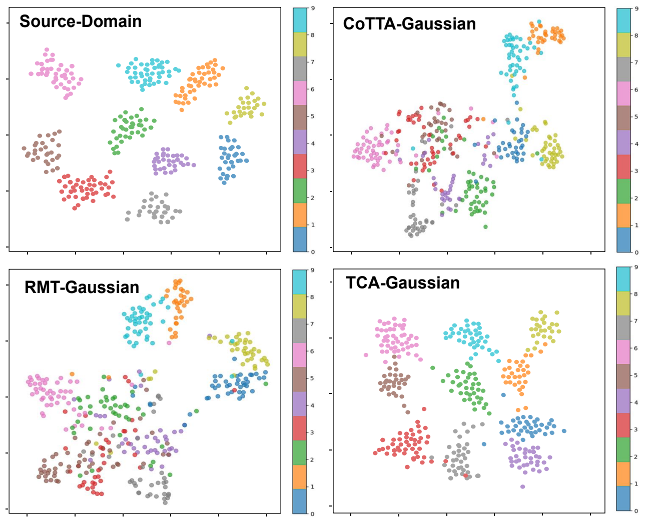

Figure 5. Visualization of t-SNE for four methods on CIFAR10-C. We select a random batch of features from the Gaussian domain for comparative visualization.

	Batchsize	CIFAR10-C	CIFAR100-C	ImageNet-C
CoTTA	1	81.6	85.3	91.2
	50	18.1	38.5	65.3
	100	16.8	35.2	64.9
TCA	1	61.5	66.2	75.4
	50	15.2	32.6	65.2
	100	14.9	30.2	61.4
	150	14.8	30.1	60.1
	200	**14.7**	**29.7**	**59.3**
	300	14.8	29.8	59.4

Table 3. Analysis of Batch size

inter-class topology to that of the source domain. This alleviates the error accumulation caused by misrepresentation.

4.6. Ablation Study

We conduct ablation studies across three classification task scenarios on Tab. 4. It is observed that directly optimizing $\mathcal{L}_{inter}$ already leads to improved model performance, indicating that promoting inter-class feature uniformity helps to optimize decision boundaries. Further optimizing $\mathcal{L}_{intra}$ significantly enhances the model's performance, demonstrating that maintaining latent inter-class topological uniformity, combined with inter-class uniformity and intra-class compactness, not only results in smoother and more natural decision boundaries but also mitigates the interference caused by outliers. Finally, incorporating BTW into $\mathcal{L}_{inter}$ further integrates class distribution information, preserving the latent geometric topological structure among classes.

As shown below, using only $\mathcal{L}_{align}$ results in marginal improvements, as it mainly acts as redundant data augmentation without stabilizing inter-class and intra-class topological relationships. Similarly, using only $\mathcal{L}_{intra}$ maintains intra-class compactness but fails to address inter-class topology collapse, though it still offers some performance gains. Note that BTW functions as a weighted matrix and cannot be ablated independently.

BTW	$\mathcal{L}_{intra}$	$\mathcal{L}_{inter}$	$\mathcal{L}_{align}$	CIFAR10-C	CIFAR100-C	ImageNet-C
-	-	-	-	16.05	32.11	63.14
-	-	-	✓	15.92	31.66	62.83
-	-	✓	-	15.57	31.38	62.08
-	✓	-	-	15.68	31.44	62.41
-	✓	✓	-	15.41	31.01	60.95
✓	✓	✓	-	14.85	29.83	59.51
✓	✓	✓	✓	14.70	29.72	59.31

Table 4. Ablation studies on four components (BTW, $\mathcal{L}_{intra}$, $\mathcal{L}_{inter}$, $\mathcal{L}_{align}$) across CIFAR10-C, CIFAR100-C, and ImageNet-C datasets.

5. Conclusion

In this paper, we propose Topological Consistency Adaptation (TCA), a novel approach for Continual Test-time Adaptation (CTTA) that addresses key challenges such as domain shifts, feature instability, and class imbalance. By introducing a class topological consistency constraint, we ensure the stability of inter-class relationships and prevent topology collapse during continuous adaptation. The incorporation of an intra-class compactness loss further stabilizes feature distributions, supporting consistent class boundaries. Additionally, the batch imbalance topology weighting mechanism optimizes centroid distances, adapting to imbalances within each batch to maintain a coherent topological structure. Experimental results show that TCA significantly improves the model's adaptability to continuously changing domains while maintaining high accuracy and stability.

Acknowledgements

This research was supported by National Science Foundation of China (NO. 62476189, No. 62406323), China Postdoctoral Science Foundation (No. 2024M753496), and Postdoctoral Fellowship Program of CPSF (No. GZC20232993).

References

[1] Malik Boudiaf, Tom Denton, Bart Van Merriënboer, Vincent Dumoulin, and Eleni Triantafillou. In search for a generalizable method for source free domain adaptation. In *ICML*, pages 2914–2931, 2023. 1

[2] Dhanajit Brahma and Piyush Rai. A probabilistic framework for lifelong test-time adaptation. In *CVPR*, pages 3582–3591, 2023. 2

[3] Nam Cao and Olga Saukh. Geometric data augmentations to mitigate distribution shifts in pollen classification from microscopic images. In *ICPADS*, 2023. 2

[4] Goirik Chakrabarty, Manogna Sreenivas, and Soma Biswas. Santa: Source anchoring network and target alignment for continual test time adaptation. *Transactions on Machine Learning Research*, 2023. 6

[5] Dian Chen, Dequan Wang, Trevor Darrell, and Sayna Ebrahimi. Contrastive test-time adaptation. In *CVPR*, pages 295–305, 2022. 2, 6

[6] Taicai Chen, Yue Duan, Dong Li, Lei Qi, Yinghuan Shi, and Yang Gao. Pg-lbo: Enhancing high-dimensional bayesian optimization with pseudo-label and gaussian process guidance. In *AAAI*, 2024. 1

[7] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. In *NeurIPS*, 2021. 6

[8] Mario Döbler, Robert A Marsden, and Bin Yang. Robust mean teacher for continual and gradual test-time adaptation. In *CVPR*, pages 7704–7714, 2023. 1, 2, 3, 5, 6, 7

[9] Matthew Gwilliam and Abhinav Shrivastava. Beyond supervised vs. unsupervised: Representative benchmarking and analysis of image representation learning. In *CVPR*, pages 9642–9652, 2022. 2

[10] Feng Hou, Jin Yuan, Ying Yang, Yang Liu, Yang Zhang, Cheng Zhong, Zhongchao Shi, Jianping Fan, Yong Rui, and Zhiqiang He. Domainverse: A benchmark towards real-world distribution shifts for tuning-free adaptive domain generalization. In *ICML*, 2024. 2

[11] Gahyeon Kim, Sohee Kim, and Seokju Lee. Aapl: Adding attributes to prompt learning for vision-language models. In *CVPR*, 2023. 2

[12] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. In *Technical report*, 2009. 5, 6

[13] Jogendra Nath Kundu, Naveen Venkat, R Venkatesh Babu, et al. Universal source-free domain adaptation. In *CVPR*, pages 4544–4553, 2020. 1

[14] Daeun Lee, Jaehong Yoon, and Sung Ju Hwang. Becotta: Input-dependent online blending of experts for continual test-time adaptation. *arXiv preprint arXiv:2402.08712*, 2024. 2, 6, 7

[15] Jiexi Liu and Songcan Chen. Timesurl: Self-supervised contrastive learning for universal time series representation learning. In *AAAI*, pages 13918–13926, 2024. 2

[16] Jiaming Liu, Ran Xu, Senqiao Yang, Renrui Zhang, Qizhe Zhang, Zehui Chen, Yandong Guo, and Shanghang Zhang. Continual-mae: Adaptive distribution masked autoencoders for continual test-time adaptation. In *CVPR*, pages 28653–28663, 2024. 2

[17] Yawei Luo, Liang Zheng, Tao Guan, Junqing Yu, and Yi Yang. Taking a closer look at domain shift: Category-level adversaries for semantics consistent domain adaptation. In *CVPR*, pages 2507–2516, 2019. 2

[18] Robert A Marsden, Mario Döbler, and Bin Yang. Introducing intermediate domains for effective self-training during test-time. In *IJCNN*, pages 1–10, 2024. 2

[19] Hao Miao, Yan Zhao, Chenjuan Guo, Bin Yang, Kai Zheng, Feiteng Huang, Jiandong Xie, and Christian S. Jensen. A unified replay-based continuous learning framework for spatio-temporal prediction on streaming data. In *ICDE*, 2024. 2

[20] Jaemin Na, Jung-Woo Ha, Hyung Jin Chang, Dongyoon Han, and Wonjun Hwang. Switching temporary teachers for semi-supervised semantic segmentation. In *NeurIPS*, 2023. 5

[21] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *ICML*, pages 16888–16905. PMLR, 2022. 6

[22] Ori Press, Steffen Schneider, Matthias Kümmerer, and Matthias Bethge. Rdumb: A simple approach that questions our progress in continual test-time adaptation. *Advances in Neural Information Processing Systems*, 36:39915–39935, 2023. 6

[23] Sanqing Qu, Tianpei Zou, Lianghua He, Florian Röhrbein, Alois Knoll, Guang Chen, and Changjun Jiang. Lead: Learning decomposition for source-free universal domain adaptation. In *CVPR*, 2024. 2

[24] Fei Shen, Xiaoyu Du, Liyan Zhang, Xiangbo Shu, and Jinhui Tang. Triplet contrastive representation learning for unsupervised vehicle re-identification. *arXiv preprint arXiv:2301.09498*, 2023. 2

[25] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *J. Big Data*, 6 (1):1–48, 2019. 3

[26] Karin Stacke, Gabriel Eilertsen, Jonas Unger, and Claes Lundström. Measuring domain shift for deep learning in histopathology. *IEEE.JBHI*, 25(2):325–336, 2020. 2

[27] Jiayao Tan, Fan Lyu, Chenggong Ni, Tingliang Feng, Fuyuan Hu, Zhang Zhang, Shaochuang Zhao, and Liang Wang. Less is more: Pseudo-label filtering for continual test-time adaptation. *arXiv preprint arXiv:2406.02609*, 2024. 2

[28] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *J Mach Learn Res*, 9(11), 2008. 7

[29] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020. 6

[30] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *CVPR*, pages 7201–7211, 2022. 1, 2, 6, 7

[31] Shuai Wang, Daoan Zhang, Zipei Yan, Jianguo Zhang, and Rui Li. Feature alignment and uniformity for test time adaptation. In *CVPR*, pages 20050–20060, 2023. 2

[32] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *ICML*, pages 9929–9939. PMLR, 2020. 2, 4, 5, 7

[33] Yanshuo Wang, Jie Hong, Ali Cheraghian, Shafin Rahman, and David Ahmedt-Aristizabal. Continual test-time domain adaptation via dynamic sample selection. In *WACV*, pages 1701–1710, 2024. 6, 7

[34] Hai Wu, Shijia Zhao, Xun Huang, Chenglu Wen, Xin Li, and Cheng Wang. Commonsense prototype for outdoor unsupervised 3d object detection. In *CVPR*, 2024. 1

[35] Shiqi Yang, Yaxing Wang, Joost Van De Weijer, Luis Herranz, and Shangling Jui. Generalized source-free domain adaptation. In *CVPR*, pages 8978–8987, 2021. 1

[36] Dong Yina, Raphael Gontijo Lopes, Jon Shlens, Ekin Dogus Cubuk, and Justin Gilmer. A fourier perspective on model robustness in computer vision. In *NeurIPS*, page 32, 2019. 6

[37] Longhui Yuan, Binhui Xie, and Shuang Li. Robust test-time adaptation in dynamic scenarios. In *CVPR*, pages 15922–15932, 2023. 1

[38] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, page 87.1–87.12, 2016. 6

[39] Daoan Zhang, Chenming Li, Haoquan Li, Wenjian Huang, Lingyun Huang, and Jianguo Zhang. Rethinking alignment and uniformity in unsupervised image semantic segmentation. In *AAAI*, pages 11192–11200, 2023. 2

[40] Zhilin Zhu, Xiaopeng Hong, Zhiheng Ma, Weijun Zhuang, Yaohui Ma, Yong Dai, and Yaowei Wang. Reshaping the online data buffering and organizing mechanism for continual test-time adaptation. In *ECCV*, pages 415–433, 2025. 2