Test-Time Adaptation in Point Clouds: Leveraging Sampling Variation with Weight Averaging

Ali Bahri* Moslem Yazdanpanah Mehrdad Noori Sahar Dastani
Milad Cheraghalikhani David Osowiechi Farzad Beizaee Gustavo A. Vargas Hakim
Ismail Ben Ayed Christian Desrosiers

ÉTS Montreal, Canada
International Laboratory on Learning Systems (ILLS)

Abstract

Test-Time Adaptation (TTA) addresses distribution shifts during testing by adapting a pretrained model without access to source data. In this work, we propose a novel TTA approach for 3D point cloud classification, combining sampling variation with weight averaging. Our method leverages Farthest Point Sampling (FPS) and K-Nearest Neighbors (KNN) to create multiple point cloud representations, adapting the model for each variation using the TENT algorithm. The final model parameters are obtained by averaging the adapted weights, leading to improved robustness against distribution shifts. Extensive experiments on ModelNet40-C, ShapeNet-C, and ScanObjectNN-C datasets, with different backbones (Point-MAE, PointNet, DGCNN), demonstrate that our approach consistently outperforms existing methods while maintaining minimal resource overhead. The proposed method effectively enhances model generalization and stability in challenging real-world conditions. The implementation is available at: https://github.com/AliBahri94/SVWA_TTA.git.

1. Introduction

Deep neural networks have recently demonstrated impressive capabilities in classifying 3D point clouds [17–19, 28, 32, 33]. However, this success typically relies on the assumption that the test data is drawn from the same distribution as the training data. In real-world applications, this assumption is often invalid. When the test distribution (*target*) differs from the training distribution (*source*), the challenge of distribution shifts arises. In 3D data, such differences can vary widely, as they may be caused by various factors including the type sensor (e.g., RGB-D camera or Lidar), conditions of the environment (e.g., low light for RGB-D camera), and occlusions. This make it impractical to pretrain the network for every possible shift encountered during testing. It is thus essential to develop methods that can adapt to these distribution changes in real-time, and without supervision, during the test phase.

By addressing a more realistic setting where distribution shifts can also occur after training, Test-Time Adaptation (TTA) recently became a focal point for researchers in machine learning and computer vision [4, 5, 8, 12, 20, 29]. TTA uses unlabeled test data to adapt a source-pretrained model to distribution shifts occurring in the testing phase. In this paper, we consider the fully-TTA setting where the model is pretrained on source data in a standard supervised manner, without any additional mechanism for adaptation, and the model is only adapted in testing. This setting contrast with Test-Time Training (TTT), where specialized strategies for adaptation are incorporated during source pretraining.

In recent years, various TTA methods have been introduced in the 2D image domain. Key strategies include regularizing the classifier on test data using objective functions based on the prediction entropy [11, 31], or updating batch normalization statistics to align with the test data distribution [14]. In the context of 3D point cloud classification, TTA is a relatively new and emerging field, with only two approaches proposed for this task: MATE [15] and BFTT3D [26]. However, MATE [15] can technically be categorized as Test-Time Training rather than TTA, as it involves using a masked autoencoder during the source pretraining phase. On the other hand, BFTT3D [26] employs a set of source prototypes to adapt to new target domains. While this prototype memory maintains privacy, it does not fully align with the core principle of TTA which aims to avoid reliance on source data during adaptation.

In this paper, we propose the first fully-TTA strategy for 3D point cloud classification. Our approach is inspired by the concept of seeking flat minima via weight averaging as

*Correspondence to ali.bahri.1@ens.etsmtl.ca

highlighted in the SWA [9] and SWAD [2] papers. Focusing on flat minima, our method aims to enhance model robustness against distribution shifts, which are common in real-world scenarios. A key innovation in our approach is the use of sampling variation to drive the adaptation. Specifically, we employ Farthest Point Sampling (FPS) and K-Nearest Neighbors (KNN) to generate multiple variations of sampled points within the point cloud, thereby introducing controlled stochasticity during adaptation. By combining the weights obtained using differently-sampled point clouds, the model is steered away from sharp minima which are more prone to overfitting and less robust to distribution shifts.

The iterative adaptation process of our method, which is guided by the prediction entropy minimization strategy of TENT [25], ensures that each variation in the sampling contributes to a broader exploration of the loss landscape. By saving the model weights after each adaptation, and subsequently averaging them, we converge to a flatter and more stable region in the loss landscape. This weight averaging technique, inspired by the SWAD approach, mitigates the impact of noise and outliers within individual samples, leading to a more robust and generalizable model.

We outline the main contributions of our work as follows:

- **Novelty**: Addressing the lack of studies in this field, we introduce the first fully-TTA method specifically designed for 3D point cloud classification. Our method proposes a novel strategy for this challenging task, which combines sampling variation and weight averaging at test time.
- **Robustness**: Our method, which achieves complete TTA without accessing any source data, demonstrates superior efficiency compared to leading approaches like TENT even with very small batch sizes.
- **State-of-art performance**: Through an extensive set of experiments involving three datasets modeling a broad range of corruptions and three different backbones for point cloud classification, we show that our method achieves state-of-art performance in most test cases.

2. Related Work

Test-Time Adaptation. TTA tackles domain adaptation in the more realistic and challenging scenario where the target domain data is unlabeled and we have no access to source domain samples. The primary challenge of this task involves accurately estimating the target domain's distribution and indirectly comparing it to the source domain's characteristics. A typical approach to reduce domain shift when source data is unavailable involves fine-tuning the model using an unsupervised loss based on the target distribution. The TTT algorithm [23] enhances the model by updating its parameters in real-time through a self-supervised task applied to the test data. TENT [25] updates the trainable batch normalization parameters during testing by minimizing the entropy of the model's predictions. Source hypothesis transfer (SHOT) [11] combines prediction entropy minimization with a diversity regularization prior (maximizing the entropy of the class marginal) to train a robust feature extractor from a pretrained source model. TTT++ [13] incorporates an additional self-supervised branch that utilizes contrastive learning within the source model to aid in adapting to the target domain. TTTFlow [16] utilizes unsupervised normalizing flows as an alternative to self-supervision for the auxiliary task.

Test-Time Point Cloud Adaptation. The concept of TTA, initially tailored for 2D images, often faces challenges when applied to 3D data, necessitating specialized approaches. So far, very few works have studied this problem in the context of 3D point cloud data. One of the first Test-Time Training (TTT) approaches specifically designed for 3D data, MATE [15], employs a Masked Autoencoder (MAE) reconstruction objective to enhance the robustness of a point cloud classification network to distribution shifts in test data. The Continual Test-Time Domain Adaptation (CTDA) method [27] employs Dynamic Sample Selection (DSS) to handle noisy pseudo-labels while adapting a pretrained model to new target domains without accessing source data. This approach, enhancing model performance through dynamic thresholding and positive-negative learning, has proven effective in both the 2D image and 3D point cloud domains.

The work in [21] presents a multi-modal extension of TTA for 3D semantic segmentation. The proposed method introduces Intra-modal Pseudolabel Generation (Intra-PG) to generate reliable pseudo labels within each modality and Inter-modal Pseudo-label Refinement (Inter-PR) to refine these labels across modalities. Hatem et al. [7] introduced a TTA technique for point cloud upsampling, leveraging meta-learning to improve model generalization during inference. In point cloud registration, Point-TTA [6] offers a TTA framework improving model generalization by adapting to each test instance through self-supervised auxiliary tasks. This method allows the model to handle unseen data distributions during testing without prior knowledge. Finally, BFTT3D [26] introduces a backpropagation-free Test-Time Adaptation (TTA) method specifically designed for 3D data, addressing domain shifts with a two-stream architecture that maintains both source and target domain knowledge. However, this approach does not fully align with the core principle of TTA, which aims to avoid reliance on source data during adaptation. Zhang et al. [31] introduced MEMO, a method that enhances model robustness during test time by applying data augmentations to a sin-

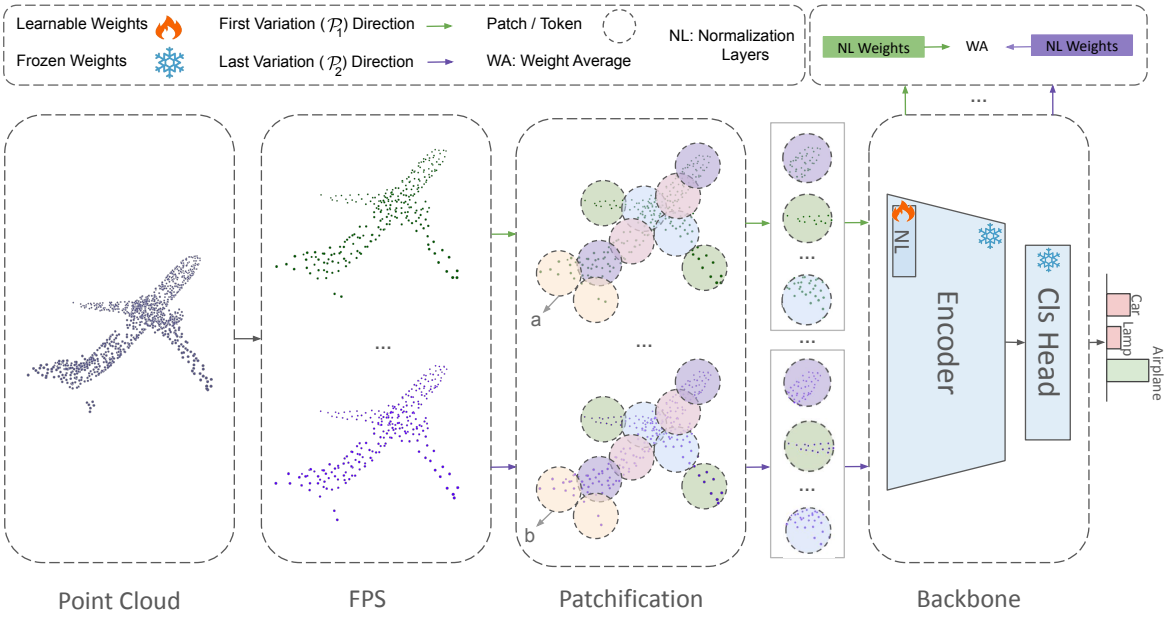

Figure 1. Overview of our 3D TTA methodology. First, FPS is applied to generate different samplings from the input point cloud. Patchification is then performed using FPS for patch centers and KNN to form patches (a and b). The Normalization Layer (NL) weights are adapted using the TENT algorithm for each sampling. Finally, weight averaging is applied across all adapted weights to enhance robustness and generalization.

gle test input and adapting model parameters to minimize the entropy of the averaged output distribution across these augmentations.

Weight Averaging. Weight averaging has become a prominent technique for enhancing the generalization of deep neural networks during training. Stochastic Weight Averaging (SWA) [9] improves model generalization by averaging weights from different training epochs, promoting smoother optimization and convergence to well-generalized solutions. Building on this, SWAD [2] refines the approach by densely sampling weights throughout the training process, further boosting generalization and robustness across tasks. Addressing limitations of traditional WA techniques that average weights post-training, Lookaround [30] introduces a novel optimization strategy integrating diversity into the training process. This method iteratively alternates between two steps: the "around step," which enhances network diversity by training multiple models on differently augmented data, and the "average step" which consolidates these models into a single network.

3. Method

We propose a novel Test-Time Adaptation (TTA) strategy tailored for 3D point cloud classification, which focuses on enhancing model robustness against distribution shifts. Our method introduces a dual approach: first, we create diverse perspectives of the input data to address distributional changes by utilizing sampling variation; second, we integrate this variation with a weight averaging technique. This combination operates within a purely test-time framework, eliminating the need for any source data during the adaptation process. The overview of our TTA method for point cloud data is presented in Figure 1.

3.1. Sampling Variation

We start with a 3D point cloud $P \in \mathbb{R}^{N^p \times 3}$ consisting of N^p points. To address the challenges posed by distribution shifts, we introduce a method that leverages sampling variation during test-time adaptation. This begins by creating patches from the point cloud, where each patch is defined by a center point c_i and its neighboring points. We employ FPS to select a set of center points $\{c_1, c_2, \ldots, c_N\}$. FPS works by iteratively selecting the point in P that is furthest from all previously selected centers:

$$c_i = \arg\max_{p \in P} \min_{c_j \in \{c_1, \ldots, c_{i-1}\}} \|p - c_j\|_2 \quad (1)$$

Then, for each center c_i, we use KNN to find its neighboring points, forming a patch of neighboring points P_i as follows:

$$P_i = \left\{ p \in P \mid p \in \text{KNN}(c_i) \right\} \quad (2)$$

This process converts the point cloud into a collection of patches $\mathcal{P} \in \mathbb{R}^{N \times K \times 3}$, where N is the number of patches and K is the number of neighbors in each patch (including the center).

During test-time adaptation, we generate multiple versions of $\mathcal{P}$ by varying the selection of centers and neighbors forming $\{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_V\}$ of size N^V. For Non-Transformer networks, we assume $K = 1$, meaning that each patch consists of only the center point selected by FPS without including any neighbors.

This variation in sampling creates different representations of the same underlying 3D structure and can be interpreted as transformations that slightly modify the local geometric structure of patches while maintaining the overall global structure of the point cloud. This helps the model generalize better by encouraging it to learn robust features that are less sensitive to such variations (e.g., a and b in Figure 1).

For each $\mathcal{P}_v$, the model is adapted using the TENT [25] algorithm, where normalization layers' parameters γ and β are updated to minimize the entropy $\mathcal{H}$ of the model's output:

$$\gamma_i^*, \beta_i^* = \arg\min_{\gamma,\beta} \mathbb{E}\left[\mathcal{H}\left(f_\theta\left(\mathcal{P}_v\right)\right)\right]. \tag{3}$$

Here, the expectation $\mathbb{E}$ is taken over the distribution of sampling variations $\mathcal{P}_v$, and $\mathcal{H}$ represents the entropy of the model's predictions, which is minimized to encourage confident and stable predictions.

3.2. Integrating Sampling Variation with Weight Averaging

For each variation $\mathcal{P}_v$, the model is adapted using the TENT algorithm, as outlined in Equation (3). The key innovation here is to combine these adaptations using a refined weight averaging technique inspired by [2, 30], leading to a more stable and generalizable model.

The concept of **weight averaging** aims to identify a solution in the parameter space that resides within a *flat region* of the loss landscape. Flat minima are characterized by a low loss that remains relatively constant under small variations of the model parameters. Solutions in such regions tend to generalize better to distribution shifts because the model's performance is less sensitive to minor variations or noise in the input data and model parameters.

This integrating sampling variation with weight averaging is related to Robust Risk Minimization (RRM) [2]. In our method, we follow a similar principle in the context of test-time adaptation. In RRM, the goal is to minimize the worst-case empirical loss within a neighborhood of model parameters. This is formulated as

$$\hat{\mathcal{E}}_{\mathcal{D}}^{\gamma}(\theta) = \max_{\|\Delta\| \leq \gamma} \hat{\mathcal{E}}_{\mathcal{D}}(\theta + \Delta), \tag{4}$$

where $\gamma > 0$ defines the neighborhood around the model parameters θ, and Δ represents small perturbations.

While we do not explicitly optimize this loss function, the concept of locating flat minima is analogous to our use of parameter averaging. By averaging the weights of models adapted from different $\mathcal{P}_v$, we effectively find a solution in the weight space that resides in a flatter region of the loss landscape. This improves the model's generalization during test-time adaptation and enhances its robustness to distribution shifts.

In our approach, after adapting the model for each variation $\mathcal{P}_v$, we store the adapted weights θ_v. These weights are then averaged to obtain the final model weights θ_{avg}:

$$\theta_{\text{avg}} = \frac{1}{N^V} \sum_{v=1}^{V} \theta_v \tag{5}$$

This averaging process provides two key benefits:

1. **Flat Minima:** Averaging the weights from different adapted models helps to locate a point in the weight space that is situated at the intersection of several flat regions. This approach reduces the model's sensitivity to specific input data configurations, thereby enhancing its robustness against distribution shifts.

2. **Error Reduction:** The averaging process mitigates the influence of errors or noise that may be present in individual model adaptations. By smoothing out fluctuations in different sub-samples, the average model becomes more stable.

4. Experiments

In this section, we conduct a comprehensive evaluation of our proposed method across multiple 3D point cloud datasets, focusing on both Transformer-based and non-Transformer-based backbones. To thoroughly assess the robustness and generalization capabilities of our approach, we perform experiments on three benchmark datasets: *ModelNet40-C*, *ShapeNet-C*, and *ScanObjectNN-C*. These datasets encompass a range of real-world challenges, including varying levels of corruption and noise, allowing us to demonstrate the effectiveness of our method in diverse and complex scenarios.

4.1. Implementation Details

During the TTA phase, we utilized multiple backbones, Point-MAE, PointNet, and DGCNN, as the source models. Each one is independently trained on its corresponding clean dataset to evaluate the robustness of our method. Pretrained backbones were adapted using our proposed approach. For all backbones, we used the AdamW optimizer with a learning rate of 0.001, consistent with the base learning rate of the TENT algorithm. Detailed information on the settings and hyperparameters are provided in the Supplementary Material. Additionally, **Resource Overhead – Time** and **Resource Overhead – Memory** related to our

method are discussed in detail in the Supplementary Material.

For the Transformer-based Point-MAE backbone, we explored two distinct adaptation strategies. In the first approach, only the Batch Normalization (BN) layers were adapted, following the standard TENT adaptation approach. In the second approach, we adapted both Layer Normalization (LN) and Batch Normalization (BN) layers. This configuration was then compared with the TENT algorithm, which similarly adapts both LN and BN layers. The results of both approaches are compared in detail in the Supplementary Material.

We only adapted the BN layers for the non-transformer-based PointNet and DGCNN backbones since these architectures do not LN layers. The input point cloud size was set to 1024 points for all experiments. All experiments were conducted using a single NVIDIA A6000 GPU, ensuring consistency across all tested configurations.

4.2. Datasets

ModelNet-40C. ModelNet-40C [22] is a robustness benchmark for point cloud classification, designed to assess how well architectures can handle real-world distribution shifts. It introduces 15 common corruption types to the original ModelNet-40 test set, categorized into three groups: transformation, noise, and density. These corruptions simulate real-world issues, such as sensor faults and noise in LiDAR scans, providing a realistic challenge for evaluating model performance under varying conditions.

ShapeNet-C. ShapeNetCore-v2 [3] is a large-scale dataset used for point cloud classification, consisting of 51,127 shapes from 55 categories. It is divided into training (70%), validation (10%), and test (20%) sets. To assess model robustness under real-world conditions, [15] applies 15 different types of corruptions to the test set, similar to those in ModelNet-40-C. These corruptions were generated using an open-source implementation provided by [22]. This modified version of the dataset is referred to as ShapeNet-C.

ScanObjectNN-C. ScanObjectNN [24] is a real-world point cloud classification dataset consisting of 15 categories. It includes 2,309 samples for training and 581 samples for testing. To evaluate model robustness, [15] introduces 15 distinct corruptions to the test set, following the methodology outlined in [22]. This modified version of this dataset is referred to as ScanObjectNN-C.

4.3. Main Results

In all result tables, *source only* refers to testing the pretrained model directly on the corrupted dataset without any adaptation. While BFTT3D [26] is included in the tables, its results are not directly comparable to ours because since, as mentioned in the Introduction, this method relies on source data during TTA. Except for the ones marked with $*$, all results are reproduced. Moreover, results marked with † indicate dependence on the source data during TTA.

ModelNet-40C. We evaluated the effectiveness of our method on the ModelNet40-C dataset using three different backbones, Point-MAE, PointNet, and DGCNN, under various corruptions. As reported in Table 1, our method consistently outperforms prior approaches across different backbones and corruptions. For the Point-MAE backbone, our method improves performance across all corruption types, with the average accuracy increasing from 72.1% (TENT) and 69.4% (MEMO) to 75.0%. This reflects the robustness of our approach in handling distribution shifts by leveraging the proposed sampling variation and weight averaging technique, as discussed in Section 3. In the PointNet backbone, our method achieves an overall improvement in mean accuracy from 60.4% (TENT) 58.5 (MEMO) to 61.3%. This improvement is driven by the better handling of difficult corruptions such as *rotation*, *distortion*, and *occlusion*, where our method consistently performs better than TENT. For the DGCNN backbone, our method performs exceptionally well on specific corruptions such as *occlusion* and *lidar*, achieving improvements of 12.7% and 8.3%, respectively, over TENT, and 21.2% and 22.7%, respectively, over MEMO. This leads to a higher overall average, increasing from 71.6% (TENT) and 63.5% (MEMO) to 74.2%. These results highlight the effectiveness of our method in improving robustness under various corruption scenarios, particularly when adapting to real-world data shifts during test-time adaptation.

ShapeNet-C. In Table 2, we evaluate our method on the ShapeNet-C dataset across the same three backbones. For the Point-MAE backbone, our method outperforms TENT and MEMO, achieving a mean accuracy of 67.0% compared to TENT's 66.1% and MEMO's 42.3%. Particularly large gains in performance are obtained by our method for the *background* and *dens-inc* corruptions. We observe similar improvements with the PointNet backbone, as our method achieves a mean accuracy of 56.9%, surpassing TENT's score of 55.5% and MEMO's score of 28.0%. Our model shows significant performance gains in difficult corruptions like *upsampling*, *rbf*, *rotation*, *shear*, and *rbf-inv*. These results demonstrate the effectiveness of our method in improving robustness across various corruption types, ensuring the model generalizes better to corrupted data during test-time adaptation. For the DGCNN backbone, our method outperforms MEMO, although its performance is somewhat comparable to the TENT method.

ScanObjectNN-C. As a final evaluation, we assessed our method on the ScanObjectNN-C dataset, using three backbones. As presented in Table 3, our method achieved notable improvements in performance across multiple corrup-

	Method	uni	gauss	backg	impul	upsam	rbf	rbf-inv	den-dec	dens-inc	shear	rot	cut	distort	oclsion	lidar	Mean
Point-MAE	Source-Only	66.6	59.1	7.2	31.8	74.6	67.7	69.8	59.3	75.1	74.4	38.0	53.7	70.0	38.6	23.4	53.9
	DUA* [14]	65.0	58.5	14.7	48.5	68.8	62.8	63.2	62.1	66.2	68.8	46.2	53.8	64.7	41.2	36.5	54.7
	TTT-Rot* [23]	61.3	58.3	34.5	48.9	66.7	63.6	63.9	59.8	68.6	55.2	27.3	54.6	64.0	40.0	29.1	53.0
	T3A* [8]	64.1	62.3	33.4	65.0	75.4	63.2	66.7	57.4	63.0	72.7	32.8	54.4	67.7	39.1	18.3	55.7
	SHOT [11]	76.9	71.2	21.4	63.6	79.0	71.2	73.3	75.0	79.6	76.6	55.7	71.8	71.2	46.1	44.7	65.1
	MATE* [15]	75.0	71.1	27.5	67.5	78.7	69.5	72.0	79.1	84.5	75.4	44.4	73.6	72.9	39.7	34.2	64.3
	PL [10]	81.9	78.1	25.4	69.3	77.0	77.4	78.4	83.7	86.8	81.4	63.0	82.3	78.1	52.3	49.5	71.0
	MEMO [31]	83.8	81.0	30.8	68.7	83.6	75.3	77.3	83.0	85.3	74.6	58.0	74.9	67.6	50.3	47.3	69.4
	TENT [25]	82.0	78.6	26.7	71.0	78.2	78.4	79.9	84.6	87.0	82.7	65.4	83.5	79.6	52.3	51.2	72.1
	Ours	**85.0**	**83.9**	**33.0**	**74.6**	**87.0**	**80.9**	**82.3**	**85.1**	**88.0**	**82.7**	**66.9**	**84.0**	**80.5**	**56.2**	**55.3**	**75.0**
PointNet	Source-Only	43.2	82.8	4.1	41.6	43.8	44.1	44.6	84.3	86.2	33.6	24.0	83.6	37.9	22.6	21.8	46.5
	LAME [1]	23.4	15.3	4.0	4.5	32.2	6.6	8.8	40.1	65.2	3.5	2.5	31.48	7.5	4.0	4.0	16.8
	SHOT [11]	71.8	83.5	11.9	66.4	71.2	61.3	61.6	82.7	83.7	41.2	29.1	80.5	49.0	28.7	28.7	56.8
	DUA [14]	67.5	84.7	8.3	61.8	67.1	60.2	60.7	84.6	86.3	43.0	31.1	84.2	51.5	32.9	35.6	57.3
	PL [10]	72.2	83.5	12.8	67.8	73.3	64.0	66.5	84.2	85.8	46.5	35.7	84.1	56.4	30.0	31.1	59.6
	BFTT3D † [26]	85.5	81.7	19.3	68.1	85.2	71.0	72.8	87.2	89.5	60.3	31.4	85.3	66.0	45.9	44.1	66.2
	MEMO [31]	72.5	83.4	11.0	66.2	72.4	63.2	65.5	83.6	85.4	47.0	35.2	80.4	52.3	33.0	26.6	58.5
	TENT [25]	72.4	83.5	**13.1**	68.4	**74.1**	65.0	68.3	84.1	86.3	48.1	37.6	**84.5**	57.8	30.4	32.9	60.4
	Ours	**73.1**	**84.0**	12.1	**69.0**	73.5	**66.1**	**68.6**	**84.5**	**86.1**	**49.2**	**40.8**	84.0	**59.7**	**34.8**	**34.2**	**61.3**
DGCNN	Source-Only	68.1	74.9	13.6	55.1	74.7	74.7	75.3	51.5	82.2	76.7	57.2	57.6	76.4	32.2	12.0	58.8
	SHOT [11]	75.3	78.2	29.1	66.6	76.0	69.9	68.9	50.4	68.8	59.4	44.7	42.1	47.1	12.5	7.4	53.1
	DUA [14]	81.6	83.1	39.4	74.0	83.4	81.2	82.1	71.4	85.5	81.2	72.3	74.3	80.5	39.0	25.5	70.3
	PL [10]	79.6	81.9	35.4	73.6	79.1	80.0	80.8	78.9	86.8	81.1	72.1	80.0	79.6	38.2	33.8	70.7
	BFTT3D † [26]	80.5	80.0	41.5	77.7	75.4	78.0	79.3	76.0	83.5	81.2	68.5	78.4	78.6	43.7	37.8	70.7
	MEMO [31]	79.0	80.7	50.1	67.9	78.7	74.1	74.6	75.7	77.8	66.0	57.9	62.6	56.3	30.7	20.5	63.5
	TENT [25]	**80.4**	**82.0**	38.6	**75.2**	80.1	**80.8**	**81.1**	78.7	86.0	**81.4**	74.3	80.8	**80.5**	39.2	34.9	71.6
	Ours	80.1	81.4	**56.8**	73.2	**83.1**	79.1	80.6	**81.9**	**86.6**	79.3	**74.5**	**81.6**	79.0	**51.9**	**43.2**	**74.2**

Table 1. Top-1 Classification Accuracy (%) for all distribution shifts in the ModelNet-40C dataset. $*$ and † are explained in Section 4.3.

	Method	uni	gauss	backg	impul	upsam	rbf	rbf-inv	den-dec	dens-inc	shear	rot	cut	distort	oclsion	lidar	Mean
Point-MAE	Source-Only	77.4	71.8	8.6	54.4	77.9	75.5	76.0	85.3	76.5	80.5	57.1	85.1	76.0	11.0	7.1	61.3
	DUA* [14]	76.1	70.1	14.3	60.9	76.2	71.6	72.9	80.0	83.8	77.1	57.5	75.0	72.1	11.9	12.1	60.8
	TTT-Rot* [23]	74.6	72.4	23.1	59.9	74.9	73.8	75.0	81.4	82.0	69.2	49.1	79.9	72.7	14.0	12.0	60.9
	T3A* [8]	70.0	60.5	6.5	40.7	67.8	67.2	68.5	79.5	79.9	72.7	42.9	79.1	66.8	7.7	5.6	54.4
	SHOT [11]	78.6	74.2	10.4	62.3	73.9	68.3	64.9	68.1	53.0	52.5	31.4	54.2	41.2	2.3	1.7	49.1
	MATE* [15]	77.8	74.7	4.3	66.2	78.6	76.3	75.3	86.1	86.6	79.2	56.1	84.1	76.1	12.3	13.1	63.1
	PL [10]	80.8	78.4	16.4	71.4	81.2	79.5	79.8	85.3	83.2	81.6	69.4	84.9	79.5	10.7	10.3	66.2
	MEMO [31]	81.2	74.5	17.5	60.3	63.8	58.9	54.0	50.2	40.4	33.8	25.6	23.4	18.4	16.6	16.4	42.3
	TENT [25]	81.1	78.7	14.3	**71.4**	81.6	79.4	79.7	**85.9**	82.6	**81.8**	70.0	**85.0**	**79.6**	9.9	10.4	66.1
	Ours	**82.6**	**81.0**	**21.2**	71.2	**82.5**	**79.8**	**80.1**	85.2	**84.8**	81.1	**70.4**	83.9	78.9	**11.2**	**11.5**	**67.0**
PointNet	Source-Only	59.9	76.1	9.3	52.8	60.3	55.3	55.0	83.1	82.6	42.9	26.3	83.0	47.8	6.3	5.5	49.8
	SHOT [11]	65.5	77.7	6.4	39.2	37.9	27.6	25.6	51.2	39.0	9.0	7.1	39.3	11.8	1.8	2.1	29.4
	DUA [14]	66.7	78.6	12.5	57.1	66.4	59.6	60.9	83.1	82.4	46.1	34.1	83.0	52.7	8.2	9.6	53.4
	PL [10]	67.0	78.7	11.8	57.4	67.8	61.5	62.2	83.2	82.8	48.5	37.6	83.0	54.9	8.5	9.0	54.3
	MEMO [31]	65.9	77.0	**13.9**	42.9	36.4	27.9	21.9	17.8	16.7	17.3	16.6	16.4	16.5	**17.1**	**16.0**	28.0
	TENT [25]	67.2	79.1	12.8	59.2	67.8	62.9	63.6	**83.4**	**82.8**	51.6	40.9	**83.0**	58.2	9.6	9.5	55.5
	Ours	**68.3**	**79.3**	12.4	**61.2**	**70.7**	**65.8**	**66.8**	82.2	82.3	**56.6**	**46.0**	80.9	**60.4**	10.0	10.1	**56.9**
DGCNN	Source-Only	74.7	74.1	30.5	55.9	75.0	77.3	78.1	85.1	81.6	79.2	66.2	84.7	77.4	8.0	6.5	63.6
	SHOT [11]	73.1	68.2	14.6	52.2	48.9	43.0	36.3	43.8	27.8	22.4	15.4	25.0	14.6	1.9	1.8	32.6
	DUA [14]	78.1	77.4	24.5	72.8	78.1	79.9	80.4	85.3	82.5	81	73.3	84.6	79.1	9.3	10.5	66.4
	PL [10]	78.3	78.2	24.1	73.4	79.3	80.4	81.1	85.5	82.7	81.5	74.5	84.7	80.0	10.1	11.6	67.0
	MEMO [31]	77.5	69.0	34.0	40.0	27.3	23.9	20.6	18.1	17.2	17.0	16.9	16.7	16.7	**18.1**	**16.8**	28.7
	TENT [25]	78.7	78.7	27.7	**73.5**	**78.6**	**79.8**	**80.2**	**84.4**	**81.8**	**79.9**	**74.6**	**83.7**	**79.2**	10.6	14.2	**67.0**
	Ours	**80.0**	**80.0**	**59.0**	69.5	77.0	74.9	74.7	80.0	78.9	74.9	70.5	77.9	72.4	9.1	12.0	66.0

Table 2. Top-1 Classification Accuracy (%) for all distribution shifts in the ShapeNet-C dataset. $*$ is explained in Section 4.3.

	Method	uni	gauss	backg	impul	upsam	rbf	rbf-inv	den-dec	dens-inc	shear	rot	cut	distort	oclsion	lidar	Mean
Point-MAE	Source-Only	20.8	32.5	16.7	16.0	22.4	32.2	35.3	68.8	64.9	35.8	28.6	69.7	33.9	9.3	9.1	33.1
	DUA* [14]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	46.0
	TTT-Rot* [23]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	46.1
	T3A* [8]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	40.3
	SHOT [11]	38.7	53.7	17.0	31.4	37.9	47.3	46.5	71.7	70.7	50.0	45.5	70.3	49.0	9.4	8.8	43.2
	PL [10]	38.1	56.0	17.4	31.4	39.6	51.0	53.7	74.6	74.4	54.7	49.0	75.4	56.0	9.0	7.4	45.8
	MATE* [15]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	47.0
	MEMO [31]	40.4	55.5	18.6	**34.0**	**41.4**	51.8	53.3	73.6	74.6	54.5	46.7	73.0	55.7	8.8	**9.2**	46.1
	TENT [25]	38.1	55.8	16.6	32.4	39.6	51.0	**54.7**	**74.2**	**75.0**	54.5	**50.0**	**74.2**	**56.8**	**8.8**	7.6	45.9
	Ours	**41.0**	**58.8**	**18.7**	33.6	39.6	**51.9**	52.5	72.1	74.2	**55.7**	48.2	74.0	54.3	8.2	9.0	**46.1**
PointNet	Source-Only	20.6	36.1	10.3	18.4	20.6	25.6	28.4	63.2	64.5	27.9	23.1	62.8	27.0	6.9	10.0	29.7
	SHOT [11]	42.2	62.3	15.0	33.8	41.6	35.9	35.3	66.0	66.4	36.3	32.8	65.4	40.4	9.0	8.4	39.4
	DUA [14]	28.1	43.3	13.9	21.7	31.2	29.1	34.8	62.9	65.6	30.1	26.0	63.3	31.2	7.6	9.2	33.2
	PL [10]	40.6	61.5	16.2	36.5	40.4	35.5	36.1	64.8	65.4	36.3	35.1	66.8	39.6	8.0	9.4	39.5
	BFTT3D † [26]	42.9	60.1	30.0	34.8	44.1	45.4	47.0	78.7	78.5	41.3	31.2	75.9	44.1	15.0	14.5	45.5
	MEMO [31]	41.6	**62.5**	**17.6**	**36.1**	**41.2**	**37.5**	**37.7**	**67.6**	65.4	39.1	31.8	65.0	39.8	10.4	9.4	40.2
	TENT [25]	40.2	61.5	16.2	35.7	40.8	35.7	37.1	65.4	66.4	35.9	**35.1**	**67.0**	39.6	8.8	**10.1**	39.7
	Ours	**42.0**	61.9	15.8	35.9	41.0	37.1	37.3	67.4	**68.2**	**40.6**	35.0	66.8	**40.2**	**10.5**	9.2	**40.6**
DGCNN	Source-Only	26.7	39.9	27.0	22.5	29.6	41.5	41.3	70.6	60.2	43.4	34.4	70.2	43.0	8.7	9.3	37.9
	SHOT [11]	36.8	48.9	28.1	45.8	33.8	50.7	49.6	70.8	65.1	53.3	47.4	67.5	51.7	8.7	8.8	44.5
	DUA [14]	38.4	51.9	30.2	48.4	39.2	54.0	55.2	73.6	70.7	57.8	50.3	74.3	56.9	9.2	10.8	48.1
	PL [10]	38.4	53.6	27.6	52.1	39.1	55.0	58.0	73.6	68.2	57.5	53.3	71.3	57.3	9.2	8.1	48.1
	BFTT3D † [26]	42.7	58.2	47.7	56.6	46.6	63.2	65.9	77.5	82.3	67.0	61.5	77.8	65.4	14.5	14.6	56.1
	MEMO [31]	37.7	52.6	**28.0**	**52.4**	39.2	51.2	54.2	67.2	62.0	48.4	47.9	59.5	50.3	**11.3**	**13.0**	45.0
	TENT [25]	39.1	**55.2**	25.3	50.5	39.7	54.0	59.2	**73.4**	68.4	58.8	54.2	71.9	56.2	10.1	8.3	48.3
	Ours	**39.2**	54.2	27.8	51.6	**43.0**	**58.3**	**59.9**	70.5	**71.2**	**61.4**	**55.4**	**74.1**	**60.6**	10.2	10.6	**49.9**

Table 3. Top-1 Classification Accuracy (%) for all distribution shifts in the ScanObjectNN-C dataset. $*$ and † are explained in Section 4.3.

tions and backbones. For the Point-MAE backbone, our approach improves the mean accuracy to 46.1%, surpassing TENT's 45.9%, with notable gains in corruptions such as *Gaussian noise* (58.8%) and *uniform noise* (41.0%). On this dataset with this backbone, our performance is equal to MEMO. Likewise, our method improved the mean accuracy to 40.6% for the PointNet backbone, outperforming TENT's 39.7%. This improvement is particularly evident in challenging corruptions such as *occlusion* (10.5%) and *shear* (40.6%). On this dataset and with this backbone, MEMO performs comparably to our method. In the DGCNN backbone, our approach yielded a mean accuracy of 49.9%, outperforming TENT's 48.3% and MEMO's 45.0%. Our method showed strong improvements like in *upsample* (43.0%), *rbf* (58.3%), and *shear* (61.4%), further demonstrating its robustness in handling distribution shifts across the dataset.

4.4. Ablation Study

In this section, we conduct a comprehensive ablation study on the ModelNet40-C dataset using the Point-MAE backbone to examine the impact of various factors on our model's performance in TTA. Specifically, we evaluate four aspects: *Sampling Variation*, *Number of Iterations*, *Batch Size*, and *Types of Augmentation*. For consistency, all experiments were conducted with a learning rate of 0.001.

Sampling Variation. We first explore the effect of increasing the number of sampling variations N^V. Figure 2 shows the steady rise in model accuracy as we increase the number of sampling variations. This validates our idea of combining weight averaging and sampling variation to enhance the model's robustness. As N^V increases from 2 to 12, the accuracy improves approximately from 74.0% to 76.0%. This demonstrates that leveraging more diverse sampling during TTA enables better adaptation to distribution shifts. Based on these results, we selected $N^V = 6$ for all subsequent experiments to balance computational efficiency and performance improvement. For this experiment, the batch size and iteration are set to 128 and 1, respectively.

Number of Iterations. As shown in Figure 5, the accuracy improves as the number of iterations increases. Our method and TENT show performance gains with more iterations, but our method consistently outperforms TENT at every stage of the iteration process. Starting with a gap at the first iteration, our approach maintains a steady improvement, surpassing TENT at each level. Based on these results, we opted to use just one iteration in subsequent experiments to prioritize faster model adaptation while still achieving a competitive accuracy boost. For this experiment, the batch size is set to 128.

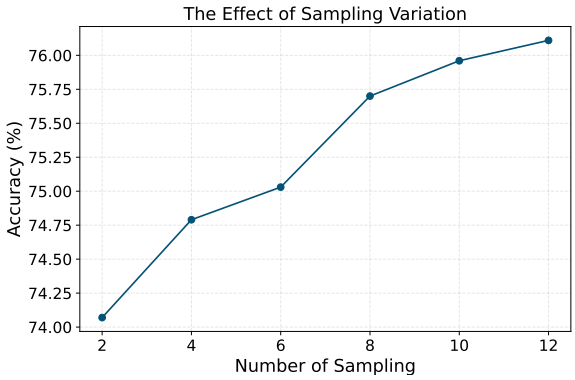

Figure 2. Impact of Different N^V on Model Accuracy

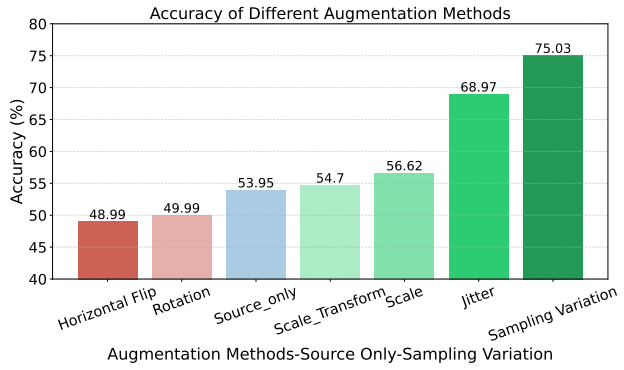

Figure 3. Comparison between Sampling Variation and Different Augmentations

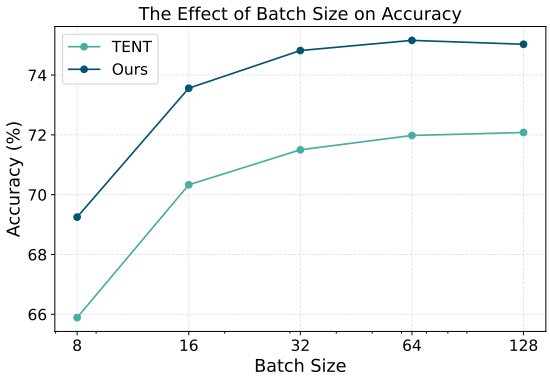

Figure 4. Impact of Batch Size on Accuracy for Two Methods

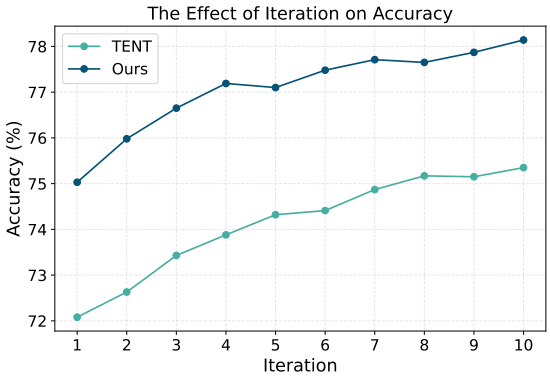

Figure 5. Impact of Iteration on Accuracy for Two Methods

Batch Size. As illustrated in Figure 4, our method consistently outperforms TENT across all batch sizes. Notably, the improvement remains steady even at lower batch sizes, such as 8 and 16, where our method achieves approximately 3% higher accuracy. This demonstrates that our approach is robust across different batch settings, making it effective even in scenarios with limited data. Based on these results, we selected a batch size of 128. For this experiment, the number of iterations is set to 1.

Types of Augmentation. It can be argued that the effect of sampling variation is akin to applying various augmentations to the data samples. Hence, we have assessed the integration of different augmentations into our weight averaging process. As shown in Figure 3, we applied augmentations such as *Horizontal Flip*, *Rotation*, *Scale*, *Scale Transform*, *Jitter*, and *Sampling Variation* to investigate their impact.

Suprisingly, certain augmentations like *Horizontal Flip* and *Rotation* worsened the model's performance, reducing accuracy even below the *source-only* model. However, augmentations like *Jitter* and especially *Sampling Variation* resulted in improvements, with *Sampling Variation* outperforming all other strategies. This highlights that our approach, which leverages sampling variation combined with weight averaging, is highly effective for boosting performance compared to data augmentations. For this experiment, the batch size and iteration are set to 128 and 1, respectively.

5. Conclusion

In this paper, we introduced a novel TTA framework combining weight averaging with sampling variation to enhance model robustness against distribution shifts in 3D point cloud data. Evaluated on multiple backbones and datasets, our method outperforms existing approaches such as TENT, particularly under challenging corruptions. Through ablation studies, we demonstrated the effectiveness of our approach across different batch sizes, iterations, and sampling variations. Our method offers a robust, efficient solution for improving generalization in 3D point cloud classification.

References

[1] Malik Boudiaf, Romain Mueller, Ismail Ben Ayed, and Luca Bertinetto. Parameter-free online test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8344–8353, 2022. 6

[2] Junbum Cha, Sanghyuk Chun, Kyungjae Lee, Han-Cheol Cho, Seunghyun Park, Yunsung Lee, and Sungrae Park. SWAD: Domain generalization by seeking flat minima. *Advances in Neural Information Processing Systems*, 34:22405–22418, 2021. 2, 3, 4

[3] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. ShapeNet: An information-rich 3D model repository. *arXiv preprint arXiv:1512.03012*, 2015. 5

[4] Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei Efros. Test-time training with masked autoencoders. *Advances in Neural Information Processing Systems*, 35:29374–29385, 2022. 1

[5] Jin Gao, Jialing Zhang, Xihui Liu, Trevor Darrell, Evan Shelhamer, and Dequan Wang. Back to the source: Diffusion-driven adaptation to test-time corruption. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11786–11796, 2023. 1

[6] Ahmed Hatem, Yiming Qian, and Yang Wang. Point-TTA: Test-time adaptation for point cloud registration using multitask meta-auxiliary learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16494–16504, 2023. 2

[7] Ahmed Hatem, Yiming Qian, and Yang Wang. Test-time adaptation for point cloud upsampling using meta-learning. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1284–1291. IEEE, 2023. 2

[8] Yusuke Iwasawa and Yutaka Matsuo. Test-time classifier adjustment module for model-agnostic domain generalization. *Advances in Neural Information Processing Systems*, 34:2427–2440, 2021. 1, 6, 7

[9] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018. 2, 3

[10] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896. Atlanta, 2013. 6, 7

[11] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International conference on machine learning*, pages 6028–6039. PMLR, 2020. 1, 2, 6, 7

[12] Hyesu Lim, Byeonggeun Kim, Jaegul Choo, and Sungha Choi. TTN: A domain-shift aware batch normalization in test-time adaptation. *arXiv preprint arXiv:2302.05155*, 2023. 1

[13] Yuejiang Liu, Parth Kothari, Bastien Van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. TTT++: When does self-supervised test-time training fail or thrive? *Advances in Neural Information Processing Systems*, 34:21808–21820, 2021. 2

[14] 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. 1, 6, 7

[15] M Jehanzeb Mirza, Inkyu Shin, Wei Lin, Andreas Schriebl, Kunyang Sun, Jaesung Choe, Mateusz Kozinski, Horst Possegger, In So Kweon, Kuk-Jin Yoon, et al. MATE: Masked autoencoders are online 3D test-time learners. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16709–16718, 2023. 1, 2, 5, 6, 7

[16] David Osowiechi, Gustavo A Vargas Hakim, Mehrdad Noori, Milad Cheraghalikhani, Ismail Ben Ayed, and Christian Desrosiers. TTTFlow: Unsupervised test-time training with normalizing flow. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2126–2134, 2023. 2

[17] Yatian Pang, Wenxiao Wang, Francis EH Tay, Wei Liu, Yonghong Tian, and Li Yuan. Masked autoencoders for point cloud self-supervised learning. In *European conference on computer vision*, pages 604–621. Springer, 2022. 1

[18] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 1

[19] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 1

[20] 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

[21] Inkyu Shin, Yi-Hsuan Tsai, Bingbing Zhuang, Samuel Schulter, Buyu Liu, Sparsh Garg, In So Kweon, and Kuk-Jin Yoon. MM-TTA: Multi-modal test-time adaptation for 3D semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16928–16937, 2022. 2

[22] Jiachen Sun, Qingzhao Zhang, Bhavya Kailkhura, Zhiding Yu, Chaowei Xiao, and Z Morley Mao. Benchmarking robustness of 3D point cloud recognition against common corruptions. *arXiv preprint arXiv:2201.12296*, 2022. 5

[23] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, pages 9229–9248. PMLR, 2020. 2, 6, 7

[24] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1588–1597, 2019. 5

[25] 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. 2, 4, 6, 7

[26] Yanshuo Wang, Ali Cheraghian, Zeeshan Hayder, Jie Hong, Sameera Ramasinghe, Shafin Rahman, David Ahmedt-Aristizabal, Xuesong Li, Lars Petersson, and Mehrtash Harandi. Backpropagation-free network for 3d test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23231–23241, 2024. 1, 2, 5, 6, 7

[27] Yanshuo Wang, Jie Hong, Ali Cheraghian, Shafin Rahman, David Ahmedt-Aristizabal, Lars Petersson, and Mehrtash Harandi. Continual test-time domain adaptation via dynamic sample selection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1701–1710, 2024. 2

[28] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (tog)*, 38(5):1–12, 2019. 1

[29] Teresa Yeo, Oğuzhan Fatih Kar, Zahra Sodagar, and Amir Zamir. Rapid network adaptation: Learning to adapt neural networks using test-time feedback. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4674–4687, 2023. 1

[30] Jiangtao Zhang, Shunyu Liu, Jie Song, Tongtian Zhu, Zhengqi Xu, and Mingli Song. Lookaround optimizer: k steps around, 1 step average. *Advances in Neural Information Processing Systems*, 36, 2024. 3, 4

[31] Marvin Zhang, Sergey Levine, and Chelsea Finn. Memo: Test time robustness via adaptation and augmentation. *Advances in neural information processing systems*, 35:38629–38642, 2022. 1, 2, 6, 7

[32] Renrui Zhang, Ziyu Guo, Peng Gao, Rongyao Fang, Bin Zhao, Dong Wang, Yu Qiao, and Hongsheng Li. Point-M2AE: Multi-scale masked autoencoders for hierarchical point cloud pre-training. *Advances in neural information processing systems*, 35:27061–27074, 2022. 1

[33] Renrui Zhang, Liuhui Wang, Yu Qiao, Peng Gao, and Hongsheng Li. Learning 3D representations from 2D pre-trained models via image-to-point masked autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21769–21780, 2023. 1