




# CloudFixer: Test-Time Adaptation for 3D Point Clouds via Diffusion-Guided Geometric Transformation

Hajin Shim<sup>1,2\*</sup>  Changhun Kim<sup>1,3\*</sup>  Eunho Yang<sup>1,3</sup> 

<sup>1</sup>Korea Advanced Institute of Science and Technology (KAIST), South Korea

<sup>2</sup>Samsung Advanced Institute of Technology (SAIT), South Korea

<sup>3</sup>AITRICS, South Korea

{shimazing, changhun.kim, eunhoy}@kaist.ac.kr

**Abstract.** 3D point clouds captured from real-world sensors frequently encompass noisy points due to various obstacles, such as occlusion, limited resolution, and variations in scale. These challenges hinder the deployment of pre-trained point cloud recognition models trained on clean point clouds, leading to significant performance degradation. While test-time adaptation (TTA) strategies have shown promising results on this issue in the 2D domain, their application to 3D point clouds remains under-explored. Among TTA methods, an input adaptation approach, which directly converts test instances to the source domain using a pre-trained diffusion model, has been proposed in the 2D domain. Despite its robust TTA performance in practical situations, naively adopting this into the 3D domain may be suboptimal due to the neglect of inherent properties of point clouds, and its prohibitive computational cost. Motivated by these limitations, we propose CloudFixer, a test-time input adaptation method tailored for 3D point clouds, employing a pre-trained diffusion model. Specifically, CloudFixer optimizes geometric transformation parameters with carefully designed objectives that leverage the geometric properties of point clouds. We also substantially improve computational efficiency by avoiding backpropagation through the diffusion model and a prohibitive generation process. Furthermore, we propose an online model adaptation strategy by aligning the original model prediction with that of the adapted input. Extensive experiments showcase the superiority of CloudFixer over various TTA baselines, excelling in handling common corruptions and natural distribution shifts across diverse real-world scenarios. Our code is available at <https://github.com/shimazing/CloudFixer>.

## 1 Introduction

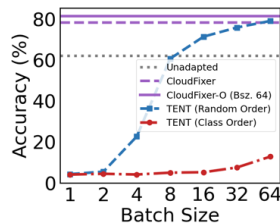
Recent advancements in 3D vision have established point clouds as expressive representations of the natural world [16], enabling lots of applications ranging from autonomous driving [8] to augmented reality [3]. These advancements have been achieved by the development of various deep neural networks tailored for

---

\* Equal contribution.

point cloud recognition [1, 45, 47, 48, 60]. These models are primarily trained on well-curated clean benchmark datasets; however, in real-world scenarios, point clouds collected from physical devices, such as LiDAR sensors, often encompass noisy points due to multiple factors, including occlusion, fluctuation in scale and density, limited resolution, and vibration or motion of the capturing devices. The distribution mismatch between the source and target domain presents challenges in deploying point cloud recognition models to real-world applications, as any misinterpretation of this data could potentially lead to catastrophic consequences.

To address distribution shift problems, a novel approach called test-time adaptation (TTA) [5, 13, 15, 25, 28, 31, 33, 41, 42, 44, 58, 59, 67] has recently emerged to adapt a pre-trained model to an arbitrary target domain during the inference phase in an unsupervised manner. Despite the abundance of TTA research in 2D vision and the necessity for developing effective TTA approaches for the 3D domain, there has been only limited investigation into TTA methods for 3D point clouds [42]. Furthermore, we unveil upon comprehensive investigation that naively applying conventional 2D test-time model adaptation approaches does not yield enough performance gain; this issue is particularly pronounced in realistic scenarios with constraints like limited batch sizes, temporally correlated test streams, and label distribution shifts, as depicted in Fig. 1. It is primarily attributed to the heavy reliance on *unstable model predictions* while underscoring the necessity for novel TTA approaches over conventional model adaptation methods.



**Fig. 1:** Accuracy of TENT [59] and CloudFixer across different batch sizes and label distributions (Random Order vs. Class Order).

In response, a method called DDA [13] was proposed to address these issues in the 2D vision domain. Instead of adapting model parameters to fit the target domain, DDA adapts the input to match the source domain using the domain translation capabilities of a pre-trained diffusion model [9, 40]. This involves forward process of adding Gaussian noise to an image, followed by iterative reverse process that aligns the image to the source domain while preserving overall shape integrity. Its per-sample basis manner operation robustness in real-world scenarios, such as limited batch sizes, temporally correlated test streams, and label distribution shifts. However, transitioning DDA to the 3D domain may be suboptimal due to its oversight of the geometric properties of point clouds and its prohibitive computational cost for real-time applications.

Motivated by these limitations, we introduce *CloudFixer*, a test-time input adaptation method tailored for 3D point clouds, leveraging a pre-trained diffusion model. Specifically, adaptation is achieved through optimization steps learning rotation matrix parameters and point displacements guided by the source diffusion model. We also suggest a per-point regularization weight for flexibility with noisy isolated points while preserving core information. To address computational costs,

our update procedure ensures that the diffusion model performs only a forward pass without backpropagation during adaptation. While our primary focus is achieving robust test-time adaptation in realistic scenarios via instance-wise input adaptation, we further propose an online model adaptation technique, minimizing a consistency loss to align class predictions of adapted and original inputs. Our experiments demonstrate that CloudFixer achieves state-of-the-art performance in various distribution shift scenarios, encompassing common and realistic corruptions in the ModelNet40-C [57] benchmark and natural distribution shifts in PointDA-10 [50]. To summarize, our contributions are threefold:

- We propose CloudFixer, which is the first test-time input adaptation strategy tailored for 3D point clouds, proposing domain-specific parameterization and objective, harnessing pre-trained diffusion models.
- CloudFixer is well-suited for real-time 3D applications as it requires neither backpropagation through the diffusion model nor an excessive generation process, enabling it to adapt a single instance in less than 1 second.
- Through extensive experiments, we demonstrate that our method achieves state-of-the-art performance across diverse distribution shift scenarios, encompassing common corruptions and natural distribution shifts.

## 2 Related Work

**Domain Adaptation and Generalization on Point Clouds** Given the frequent exposure of point clouds to distribution shifts in real-world scenarios, various domain adaptation and generalization strategies have been proposed. A significant branch involves unsupervised domain adaptation (UDA) [2, 4, 6, 11, 50, 54, 70]. These methods aim to achieve compatible performance on the target domain by leveraging labeled source data and unlabeled target data. They typically employ unsupervised objectives, such as pseudo-labeling [11], and self-supervised tasks that predict the geometric properties [2, 4, 6, 50, 54, 70]. Another big category is domain generalization (DG) [29, 61, 61, 64]. These primarily involve adversarial learning or explicit feature alignment across multiple domains [23, 61, 64] and utilize data augmentation methods [29]. Despite showcasing decent performance gains, these methods have inability to operate in cases where source data is inaccessible due to privacy or storage concerns.

**Test-Time Adaptation** Test-time adaptation (TTA) [5, 13, 15, 25, 28, 31, 33, 41, 42, 44, 58, 59, 67] strategies have emerged to address limitations in UDA/DG. These methods aim to adapt pre-trained models from a source to an arbitrary target domain on the fly, utilizing only unlabeled target data without access to source data. TTAs are typically categorized as follows. Firstly, fully TTA [25, 28, 31, 44, 59, 67] involves training the model unsupervisedly, leveraging self-training on the unlabeled target dataset. Secondly, test-time training [33, 42, 58] entails training the model with both the main objective and an additional self-supervised task and utilizing this self-supervision for adaptation during inference. Next,

batch-norm statistics calibration methods [15, 32, 41, 65] update statistics of batch normalization layers using test instances to estimate unbiased normalization statistics of target domain. In contrast, our approach diverges from these methods by projecting input instances into the training data regime as [13] using a pre-trained diffusion model on the source domain tailored for point clouds.

**Diffusion Models** Diffusion models [19, 55, 56], which approximate the *reverse* of the diffusion process to learn the training manifold, have gathered significant attention as prominent generative models. They have been widely used in various domains, including 2D/3D vision [24, 34, 51–53, 66, 69], videos [18, 20, 26], and languages [14, 30]. For 3D point clouds, several diffusion models have been proposed, especially for generation [21, 35, 69], completion [36, 37], and manipulation [66]. Diffusion models exhibit remarkable abilities in translating the domain of given inputs to the source domain. For example, ILVR [9] accomplishes domain translation by iteratively generating an image while preserving the low-pass filtering results, and SDEdit [40, 68] dilutes the reference image of a different domain through a forward process, projecting it into the latent space where the two domains intersect. Instead of these, we adopt an optimization-based method [46], while acknowledging the unique characteristics of point clouds.

### 3 Preliminary

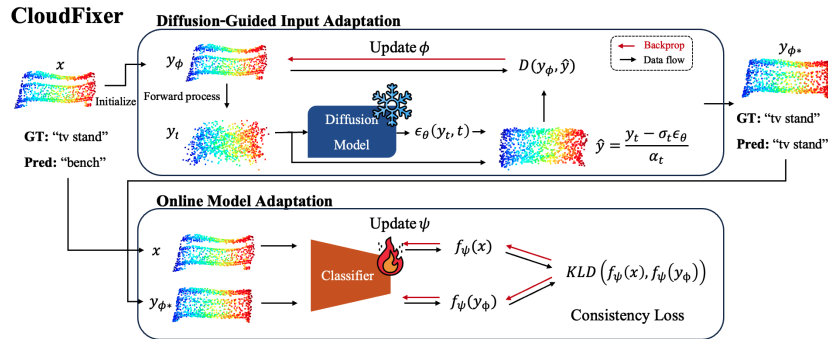
#### 3.1 Problem Setup

Let  $q(x)$  be a source distribution of point clouds  $x \in \mathbb{R}^{N \times 3}$  consisting of  $N$  points for training dataset  $\mathcal{D}_s = \{(x_i^s, c_i^s)\}_i$  in pairs of point clouds and class labels, and  $f_\psi(x) \in \mathbb{R}^C$  be a  $C$ -class classification model trained on  $\mathcal{D}_s$ . Given unlabeled target domain  $\mathcal{D}_t = \{x_i^t\}_i$  under distribution shifts, *i.e.*,  $x_i^t \not\sim q(x)$ , our test-time input adaptation aims to achieve robust prediction performance under distribution shifts. Existing TTA methods usually adapt the model  $f_\psi$  by optimizing  $\psi$  using self-training, based on the prediction on test instances or replacing the statistics in normalization layers with those of test examples [31, 44, 59, 67]. Instead, we leverage the domain translation ability of a pre-trained diffusion model to directly transform the input into the source domain.

#### 3.2 Diffusion Models

Diffusion models [19, 55, 56] are generative models that estimate the reverse process of the diffusion process to generate the data  $x_0 \sim q(x_0)$  by gradually denoising random noise  $x_T \sim \mathcal{N}(x_T; 0, I)$ . For each timestep  $t = 0, \dots, T$ , the marginal distribution of  $x_t$  given  $x_0$  is defined as  $q(x_t|x_0) = \mathcal{N}(x_t; \alpha_t x_0, \sigma_t^2 I)$ , where  $\alpha_t$  strictly decreases from 1 to 0 as  $t$  increases, and  $\sigma_t = \sqrt{1 - \alpha_t^2}$ . The models are trained for  $p_\theta(x_{t-1}|x_t)$  to approximate  $q(x_{t-1}|x_t, x_0)$  by minimizing

$$\mathcal{L}_{\text{diff}}(\theta) = \mathbb{E}_{x_0 \sim q(x_0), \epsilon \sim \mathcal{N}(0, I), t} \left[ w(t) \|\epsilon_\theta(x_t, t) - \epsilon\|_2^2 \right],$$



**Fig. 2:** CloudFixer is an optimization-based diffusion-guided input adaptation, tailored for 3D point clouds. CloudFixer iteratively optimizes geometric transformation parameters  $\phi$  for  $x$  to minimize the Chamfer distance between a parameterized point cloud  $y_\phi$  and the estimation  $\hat{y}$  from the diffusion model, aligned with the source domain. A distorted input point cloud  $x$ , leading to a misclassification, is transformed into  $y_{\phi^*}$ , correcting its prediction. Additionally, online model adaptation minimizes the KL-divergence between class predictions of the original and adapted point clouds.

where  $w(t)$  is a weight function of timestep  $t$ ,  $x_t = \alpha_t x_0 + \sigma_t \epsilon$  is the forwarded input, and the parameterized model  $\epsilon_\theta(\cdot, t)$  is to predict truly injected Gaussian noise  $\epsilon$  in the forward process. With the estimated noise, we can estimate the original data point  $\hat{x}_0$  from  $x_t$  as follows:

$$\hat{x}_0 = \frac{x_t - \sigma_t \epsilon_\theta(x_t, t)}{\alpha_t}.$$

Here,  $\epsilon_\theta(x_t, t)$ , which approximates  $-\sigma_t \nabla_{x_t} q(x_t)$  allows us to obtain the direction to move from a noised input towards the data distribution. When  $x_0$  is not from the source domain but from another shifted domain, the disparity between the estimation and the data point serves as guidance toward the source domain.

## 4 Proposed Method: CloudFixer

In this section, we introduce CloudFixer, which is a per-sample diffusion-guided test-time input adaptation strategy tailored for 3D point clouds. We outline our input adaptation and an optional online model adaptation methods in Section 4.1. Following this, we cover the details of components of CloudFixer in Section 4.2 and 4.3. The overall procedure is depicted in Algorithm 1 in Appendix A.

### 4.1 Overview

CloudFixer directly aligns the distribution shift of a given test instance with the source domain, leveraging the knowledge of the pre-trained diffusion model  $p_\theta(x)$  on the source domain. A straightforward domain translation using the

diffusion model is a generation-based method, where data points undergo a forward process, mapping them to a noisy latent space at time  $t$ , followed by a denoising process [9, 13, 40, 68]. However, at larger timestep  $t$ , there is a risk of losing class information, while a small  $t$  may not provide sufficient translation.

Building upon the limitations above, we propose novel *optimization-based geometric transformation* with diffusion guidance (upper part of Fig. 2) as follows. First, we define geometric transformation  $y_\phi(x)$  parameterized by  $\phi$  of a test input  $x$ .  $\phi$  is initialized for  $y_\phi(x)$  to be same as  $x$ . We iteratively update  $\phi$  with diffusion guidance. In each iteration, we randomly sample  $t$  from  $U[t_{min}, t_{max}]$  and conduct a forward process from  $y_\phi(x)$  to  $y_t$ . Subsequently, we obtain the estimation of the denoised point cloud  $\hat{y}$  from the pre-trained diffusion model  $\epsilon_\theta$ . Since the diffusion model is trained on the source domain,  $\hat{y}$  transitions from  $y_\phi$  to the source domain. Therefore, we can utilize  $y_\phi$  as a supervision to update  $\phi$ . Through this iterative optimization, we can effectively encourage stable and gradual translation of shifted test instances into the source domain.

Although our primary focus is on input adaptation, we further extend CloudFixer to include online model adaptation (CloudFixer-O) to enhance adaptation performance under conditions where enough batch size and an independent and identically distributed (i.i.d.) distributions are ensured. Specifically, we introduce a task that aligns the class distribution predictions of the original  $x$  and the transformed  $y_{\phi^*}$  (lower part of Fig. 2). This approach facilitates domain adaptation by establishing a correlation between source and target domains, mitigating the need for uncertain information like pseudo-labels as long as the diffusion-guided transformation is valid.

## 4.2 Diffusion-Guided Input Adaptation

**Parameterization of Geometric Transformation  $y_\phi$**  We set the parameters  $\phi = (R, \Delta)$  for transforming the given input  $x$  as  $y_\phi(x) = (x + \Delta)R^\top$  where  $R \in \mathbb{R}^{3 \times 3}$  is a rotation matrix and  $\Delta \in \mathbb{R}^{N \times 3}$  is a displacement matrix of all  $N$  points whose  $j$ -th row  $\delta_j^\top$  corresponds to each point  $j$ . We include the rotation transformation  $R$  because misalignment is a common test-time corruption of point clouds. Furthermore, the simple per-point displacement  $\Delta$  is to allow flexible transformation in response to various distribution shifts. Note here that the rotation matrix  $R = [r_1; r_2; r_3]$  is further parameterized by a 6D vector  $(a_1, a_2) \in \mathbb{R}^3 \times \mathbb{R}^3$  to satisfy the condition of rotation matrices, as derived by the following operation as in [17]:

$$r_1 = \frac{a_1}{\|a_1\|_2}, \quad r_2 = \frac{u_2}{\|u_2\|_2}, \quad u_2 = a_2 - (r_1 \cdot a_2)r_1, \quad \text{and} \quad r_3 = r_1 \times r_2.$$

**Objective** As we mentioned in Section 3.2, a pre-trained diffusion model provides guidance toward the source domain of the noised input. For each iteration, after perturbing  $y_\phi$  with the forward process at time  $t$  as  $y_t = \alpha_t y_\phi + \sigma_t \epsilon$ , we estimate the denoised point cloud  $\hat{y} = (y_t - \sigma_t \epsilon_\theta(y_t, t)) / \alpha_t$ . The estimation moves from  $y_\phi$  towards the source domain by the diffusion model. Therefore, we update

the parameters to reduce the distance  $D$  between  $y_\phi$  and the diffusion model’s estimation  $\hat{y}$  as follows:

$$\phi \leftarrow \phi - \eta \left( \nabla_{y_\phi} D(\hat{y}, y_\phi) \frac{\partial y_\phi}{\partial \phi} + \lambda \nabla_\phi \text{Reg}(\phi) \right), \quad (1)$$

where  $\eta$  is a learning rate. When the distance is the square of  $\ell_2$  distance  $\|y - y_\phi\|_2^2$ , this update is equivalent to the Score Distillation Sampling (SDS) loss [46] up to the scaling of each timestep  $t$ . However, we observe that optimizing this objective does not lead to a stable convergence towards the source domain. Instead, by leveraging the characteristic of point clouds as unordered sets, we use the Chamfer distance to account for the permutation-invariant nature of 3D point clouds defined as follows:

$$D(x, y) = \frac{1}{|x|} \sum_{i=1}^{|x|} \min_j \|x(i) - y(j)\|_2^2 + \frac{1}{|y|} \sum_{j=1}^{|y|} \min_i \|x(i) - y(j)\|_2^2,$$

where  $x(i)$  and  $y(j)$  are 3D coordinates of point  $i$  and  $j$  of  $x$  and  $y$ , respectively. It is worth noting that in CloudFixer, the diffusion model is used solely for predicting  $\hat{y}$  for the supervision of  $y_\phi$ , and it does not involve backpropagation through the diffusion model. This stands in stark contrast to the 2D input adaptation-based TTA method DDA [13], which requires backpropagation through the diffusion model for guidance. This key difference is crucial, especially in the 3D domain where efficiency is essential. The originality of CloudFixer compared to DDA is summarized in Appendix B.

**Regularization** To simply regulate excessive changes in the point cloud, we introduce a novel regularization objective by penalizing the squared norm of the displacement  $\delta_j$  for each point  $j$ , as  $\text{Reg}(\Delta) = \sum_j w_j \|\delta_j\|_2^2$ . Here, for each point, we calculate weights  $\{w_j\}_j$  by taking the inverse of the average distance to its  $k$ -nearest neighbors, providing greater flexibility for noisy isolated points.

**Voting** Due to the stochasticity of the input adaptation, we can obtain  $K$  different transformations  $\{y_{\phi_j}(x)\}_{j=1}^K$  of a given input  $x$ . We enhance classification performance by averaging the multiple predictions  $\sum_j f_\psi(y_{\phi_j})/K$ , where  $f_\psi(\cdot) \in [0, 1]^C$  is a model prediction of class probability.

### 4.3 Online Model Adaptation

We extend CloudFixer to its online version, *CloudFixer-O*, by introducing a model adaptation process that incorporates the adapted inputs  $\{y_{\phi_j}(x)\}_{j=1}^K$  as well as the original  $x$ . We perform an  $M$ -step update on the model except for a classification head by minimizing the KL-divergence between the class probability distribution of the test input  $x$  and its adapted inputs  $\{y_{\phi_j}\}$  as follows:

$$\min_\psi \sum_{j=1}^K KL \left( f_\psi(x) \middle| f_\psi(y_{\phi_j}(x)) \right),$$

where  $f_\psi(\cdot) \in [0, 1]^C$  is model prediction of class probability. Intuitively, our objective guides the feature encoder of the classifier to extract class information, disregarding shifted features, by aligning the class prediction of the test input with the source-closer adapted input.

## 5 Experiments

This section rigorously and thoroughly demonstrates the empirical efficacy of CloudFixer. We begin by providing a detailed description of our experimental setup in Section 5.1, followed by an elucidation of the fundamental research questions as follows: Does CloudFixer consistently outperform baselines under challenging yet realistic scenarios such as limited batch size, temporally correlated test streams, and label distribution shifts, as well as under mild conditions across various benchmarks and classifier architectures? (Section 5.2 and Section 5.3) Do the components of CloudFixer truly contribute to performance enhancement, and are they optimal choices? (Section 5.3) Moreover, when visualized, does CloudFixer genuinely transform point clouds into the desired source domain? Does CloudFixer demonstrate strengths in computational efficiency (Section 5.4) and hyperparameter sensitivity (Section 5.4), which can be pivotal at test time?

### 5.1 Experimental Setup

**Datasets** We evaluate our method on ModelNet40-C [57] which includes the various types common corruptions of ModelNet40 [62]. It suggests 15 common and realistic corruptions that are categorized into 3 types—density, noise, and transformation. We abbreviate the corruption names as follows: OC (Occlusion), LD (LiDAR), DI (Density Inc.), DD (Density Dec.), CO (Cutout), UNI (Uniform noise), GAU (Gaussian noise), IP (Impulse), US (Upsampling), BG (Background noise), ROT (Rotation), SH (Shear), FFD (Distortion), RBF (Distortion with RBF Kernel), and IR (Distortion with Inverse RBF Kernel). We also utilize a domain adaptation benchmark, PointDA-10 [50], which provides natural distribution shifts of sim-to-real and vice versa by pairing two out of the three datasets—ModelNet [63], ShapeNet [7], and ScanNet [10]—as the source and target domains. ModelNet and ShapeNet are generated from 3D CAD models, while ScanNet is created by scanning real scenes. Further details are in Appendix E.

**Model Architectures** For classifiers, we employ Point2Vec [1], which has demonstrated notable success on ModelNet40 [62], as a classifier backbone for ModelNet40-C, by using a publicized checkpoint in the official repository. We additionally evaluate CloudFixer’s performance on PointMLP [38], PointNeXt [49], and PointMAE [45]. For PointDA-10, we use DGCNN [60], which serves as the backbone for most UDA works, and manually train it for each of the three datasets. As a diffusion model, we adopt `base40M-uncond` from Point-E [43] in Point-E repository and manually train it on each source dataset.



**Table 1:** Accuracy on ModelNet40-C with a limited batch size of 1 using Point2Vec.

Method	Density Corruptions					Noise Corruptions					Transformation Corruptions					Avg.
	OC	LD	DI	DD	CO	UNI	GAU	IP	US	BG	ROT	SH	FFD	RBF	IR	
Unadapted	41.09	20.02	90.03	<u>86.51</u>	83.91	63.41	49.68	66.25	38.33	37.68	47.04	79.29	75.97	74.68	77.51	62.09
PL [28]	30.11	4.38	<b>91.65</b>	<b>88.82</b>	<b>85.74</b>	88.78	5.88	80.83	4.70	5.02	30.92	83.71	81.32	82.17	83.91	56.53
TENT [59]	4.05	4.05	5.27	4.29	5.55	4.38	4.05	3.89	4.05	4.05	4.05	4.17	4.05	4.05	4.42	4.29
SHOT [31]	4.01	4.05	4.05	4.05	4.09	4.05	4.05	4.01	4.01	4.05	4.05	4.05	4.05	4.05	4.05	4.04
SAR [44]	16.37	5.71	90.44	83.43	83.47	81.40	73.10	32.17	14.83	24.03	19.73	77.51	75.93	76.54	78.04	55.51
DUA [41]	41.37	20.26	89.71	86.10	83.71	62.56	49.64	65.68	37.48	36.95	47.37	<b>79.58</b>	75.57	73.70	77.23	61.79
LAME [5]	40.96	20.50	90.19	85.70	84.16	63.57	49.92	66.25	38.33	37.32	47.04	79.34	75.24	74.72	77.59	62.06
MEMO [67]	<b>46.27</b>	36.67	<u>90.76</u>	82.94	81.77	79.90	76.18	80.88	70.91	32.94	63.57	<u>79.74</u>	<u>79.21</u>	78.28	81.16	70.75
DDA [13]	<u>42.67</u>	36.30	89.26	83.10	<u>84.44</u>	90.44	89.38	<u>90.76</u>	86.26	53.69	49.68	78.69	77.43	80.71	82.78	74.37
CloudFixer	41.00	<u>38.82</u>	87.32	80.27	83.06	<u>91.09</u>	<u>90.52</u>	<u>90.76</u>	<u>89.06</u>	<u>75.49</u>	<u>81.04</u>	78.28	78.73	<u>82.98</u>	<u>85.09</u>	<u>78.23</u>
+ Voting ( $K=5$ )	41.00	<b>38.90</b>	88.05	80.55	84.16	<b>91.45</b>	<b>91.29</b>	<b>91.98</b>	<b>89.79</b>	<b>76.58</b>	<b>83.51</b>	79.38	<b>79.70</b>	<b>84.12</b>	<b>85.90</b>	<b>79.09</b>

**Table 2:** Accuracy on ModelNet40-C with temporally correlated non-i.i.d. test stream using Point2Vec, where the test set is sorted based on label order.

Method	Density Corruptions					Noise Corruptions					Transformation Corruptions					Avg.
	OC	LD	DI	DD	CO	UNI	GAU	IP	US	BG	ROT	SH	FFD	RBF	IR	
Unadapted	41.33	14.22	90.15	75.20	75.93	63.01	50.32	<u>67.06</u>	37.60	36.75	47.61	<u>79.05</u>	75.45	74.68	77.39	60.38
PL [28]	16.69	11.43	24.03	21.88	20.14	20.83	19.77	20.10	18.64	9.76	16.90	22.45	20.75	21.03	20.62	19.00
TENT [59]	12.16	8.59	16.61	16.33	14.26	14.55	14.75	14.91	14.55	6.24	8.35	10.25	11.51	15.92	13.01	12.80
SHOT [31]	12.16	8.59	16.61	16.33	14.26	14.55	14.75	14.91	14.55	6.24	8.35	10.25	11.51	15.92	13.01	12.80
SAR [44]	16.61	10.86	23.78	20.79	20.58	20.79	19.00	18.68	17.79	9.85	17.26	22.29	21.07	20.79	20.75	18.73
DUA [41]	19.29	13.49	26.86	24.96	23.74	24.11	22.53	23.10	21.23	12.12	19.21	24.68	23.62	23.70	22.49	21.68
LAME [5]	<b>43.52</b>	14.14	<b>95.38</b>	<u>84.44</u>	<b>84.76</b>	70.46	53.12	73.22	40.92	41.45	<u>51.34</u>	<b>86.14</b>	<b>83.75</b>	<u>81.48</u>	<u>84.64</u>	65.92
MEMO [67]	40.76	20.38	<u>90.28</u>	<b>85.90</b>	84.00	62.40	49.23	65.68	37.52	36.67	47.16	<u>79.05</u>	75.32	74.43	77.47	61.75
DDA [13]	<u>42.67</u>	<u>36.30</u>	89.26	83.10	<u>84.44</u>	<u>90.44</u>	<u>89.38</u>	<b>90.76</b>	<u>86.26</u>	<u>53.69</u>	49.68	78.69	77.43	80.71	82.78	<u>74.37</u>
CloudFixer	41.00	<b>38.82</b>	87.32	80.27	83.06	<b>91.09</b>	<b>90.52</b>	<b>90.76</b>	<b>89.06</b>	<b>75.49</b>	<b>81.04</b>	78.28	<u>78.73</u>	<b>82.98</b>	<b>85.09</b>	<b>78.23</b>

**Baselines** To compare CloudFixer with existing TTA methods, we implement nine TTA baselines: PL [28], TENT [59], SHOT [31], SAR [44], DUA [41], LAME [5], MEMO [67], DDA [13], and MATE [42]. The details of these baselines are in Appendix F. They represent diverse TTA categories, including test-time training [42], fully test-time adaptation [28, 31, 44, 59, 67], batch-norm statistics calibration [41], input adaptation [13], and output adaptation [5]. Notably, we extend DDA [13] to the 3D domain by using farthest point sampling instead of low-pass filtering with the chamfer distance for regularization. MATE [42] is the only test-time training method tailored for 3D point clouds.

**Implementation Details** We conduct zero centering and scale each point cloud to a unit ball before passing it to the classifier by following [60]. Meanwhile, we standardize the point clouds to achieve a unit variance for diffusion models by following [66]. For diffusion models, we use the polynomial noise scheduling as in [21] and set the total number of timesteps as  $T = 500$ . The batch size [31, 59] is set to 64 for the TTA baselines unless specified except CloudFixer and other per-sample TTA baselines (MEMO [67], DDA [13]) which inherently operate with a batch size of 1. For input adaptation, we perform 30 steps of updates with AdaMax optimizer [27]. The timestep interval for the diffusion forward process is set as  $[0.02T, 0.12T]$ . Further details are in Appendices G and H and our repository.

**Table 3:** Macro-recall on ModelNet40-C with label distribution shifts, featuring a high class imbalance ratio of 100, using Point2Vec.

Method	Density Corruptions					Noise Corruptions					Transformation Corruptions					Avg.
	OC	LD	DI	DD	CO	UNI	GAU	IP	US	BG	ROT	SH	FFD	RBF	IR	
Unadapted	47.59	21.29	<b>88.78</b>	<u>82.03</u>	79.76	57.77	46.04	59.41	35.81	29.25	49.88	<u>75.81</u>	75.62	75.95	74.07	59.94
PL [28]	47.87	39.53	83.04	80.33	77.18	74.62	68.10	69.55	65.95	28.03	56.35	73.57	72.15	75.28	75.79	65.82
TENT [59]	35.71	32.99	76.93	74.12	65.13	64.46	66.74	63.95	66.09	27.39	46.49	55.69	65.42	69.19	67.04	58.49
SHOT [31]	35.44	28.90	66.72	65.08	64.71	56.88	59.36	58.21	58.92	33.41	37.10	61.73	64.25	62.40	66.92	54.67
SAR [44]	<b>49.07</b>	40.27	85.63	79.98	75.48	71.33	68.28	68.08	64.04	28.58	55.48	72.34	75.44	75.73	72.84	65.50
DUA [41]	51.03	<b>42.56</b>	85.16	<b>82.89</b>	81.98	77.39	71.37	74.81	67.55	29.78	<u>59.86</u>	<b>78.07</b>	<u>76.95</u>	<u>76.25</u>	77.45	68.87
LAME [5]	45.37	15.91	86.52	75.81	72.24	49.42	40.09	51.46	27.22	25.15	43.77	67.49	68.43	65.64	67.46	53.47
MEMO [67]	<u>47.97</u>	19.53	<u>88.31</u>	81.67	80.26	57.00	47.05	57.19	34.01	28.54	49.26	72.84	73.63	74.36	75.28	59.13
DDA [13]	47.88	39.49	84.93	80.77	<u>82.58</u>	<u>84.71</u>	<u>85.77</u>	<u>85.49</u>	<u>78.81</u>	<u>43.07</u>	51.67	74.79	75.23	75.55	<u>78.79</u>	<u>71.30</u>
<b>CloudFixer</b>	44.43	<u>40.91</u>	86.87	77.23	<b>83.75</b>	<b>91.46</b>	<b>91.72</b>	<b>92.23</b>	<b>85.80</b>	<b>75.53</b>	<b>76.83</b>	75.36	<b>78.68</b>	<b>80.11</b>	<b>83.04</b>	<b>77.60</b>

**Table 4:** Accuracy on ModelNet40-C using Point2Vec under the mild conditions of a batch size of 64 and an i.i.d. test stream.

Method	Density Corruptions					Noise Corruptions					Transformation Corruptions					Avg.
	OC	LD	DI	DD	CO	UNI	GAU	IP	US	BG	ROT	SH	FFD	RBF	IR	
Unadapted	41.09	20.02	90.03	<u>86.51</u>	83.91	63.41	49.68	66.25	38.33	37.68	47.04	79.29	75.97	74.68	77.51	62.09
PL [28]	46.07	38.17	90.56	84.20	82.13	83.06	78.81	81.77	76.66	38.05	63.49	81.40	79.09	79.21	81.36	72.27
TENT [59]	<u>47.33</u>	37.88	<u>91.25</u>	<b>87.44</b>	<u>86.43</u>	<u>89.14</u>	<u>87.80</u>	<u>85.41</u>	<u>87.64</u>	<u>67.91</u>	71.23	<b>85.37</b>	<b>85.05</b>	<b>86.30</b>	<u>86.51</u>	<u>78.85</u>
SHOT [31]	46.72	<b>47.45</b>	86.83	84.60	82.90	80.92	79.78	74.92	80.92	61.51	<u>74.31</u>	78.48	81.16	81.73	82.54	74.98
SAR [44]	46.27	36.67	90.76	82.94	81.77	79.90	76.18	80.88	70.91	32.94	63.57	79.74	79.21	78.28	81.16	70.75
DUA [41]	<b>47.85</b>	39.55	<b>91.45</b>	85.25	83.51	81.28	77.84	82.62	74.92	38.49	66.00	82.66	80.23	<u>80.51</u>	82.94	73.01
LAME [5]	39.18	9.52	90.03	76.30	76.54	61.79	46.31	63.90	35.09	31.93	45.75	78.44	75.04	74.35	76.74	58.73
<b>CloudFixer-O</b>	46.39	<u>44.94</u>	90.92	84.76	<b>86.99</b>	<b>91.94</b>	<b>91.86</b>	<b>91.82</b>	<b>92.14</b>	<b>74.92</b>	<b>85.98</b>	<u>83.83</u>	<u>82.09</u>	<b>86.30</b>	<b>87.40</b>	<b>81.49</b>

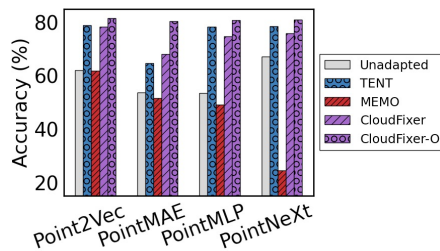
## 5.2 Main Results

**Results on Challenging Real-world Scenarios** We initially assess CloudFixer in a genuinely plausible real-world scenario to verify its practical reliability. We conduct validation under the batch size of 1 in Table 1, considering real-time inference, temporally correlated test streams in Table 2, and significant label distribution shifts due to high class-imbalances in Table 3 on ModelNet40-C using Point2Vec. CloudFixer consistently outperforms other methods across various corruptions, particularly in noise and transformation corruptions. This results in a remarkable average performance improvement of 17% to 19% compared to ‘Unadapted’ in all challenging real-world scenarios. In contrast, entropy minimization variants, such as TENT and SHOT, only achieve an accuracy of about 10% in Table 1 and Table 2, indicating model collapse. While SAR was proposed to overcome such challenging scenarios in 2D vision, it also fails in the point cloud domain. Although Per-sample adaptation methods, such as MEMO and DDA, do not collapse in these settings, MEMO shows only marginal improvement, and DDA, despite having excessive computational cost (see Section 5.4), consistently exhibits lower average performance than ours. This underscores the effectiveness of optimizing geometric transformations specialized for 3D point clouds.

**Results on Mild Conditions** While our primary focus remains on ensuring the effectiveness of our method in plausible real-world scenarios, we also verify CloudFixer in a mild scenario of batch size of 64 and i.i.d. test stream, following

**Table 5:** Accuracy on PointDA-10 using DGCNN. We report the performance of all method, except MEMO, DDA, CloudFixer, including CloudFixer-O with a batch size of 64.

Method	M → S	M → S*	S → M	S → S*	M → S*	S* → M	S* → S	Avg.
Unadapted	81.38	52.23	77.10	44.83	62.03	64.69	63.71	
PL [28]	75.52	50.99	66.82	<u>50.42</u>	58.88	61.12	60.63	
TENT [59]	76.77	52.63	65.07	49.46	57.59	59.31	60.14	
SHOT [31]	14.53	14.53	67.17	47.77	44.63	19.62	34.71	
SAR [44]	74.12	49.75	66.36	50.03	54.32	58.83	58.90	
DUA [41]	74.80	50.25	67.29	49.24	59.93	60.03	60.26	
LAME [5]	82.26	<b>53.93</b>	<b>80.84</b>	44.83	20.68	43.42	54.33	
MEMO [67]	75.84	47.26	73.25	50.99	47.90	60.27	59.25	
DDA [13]	82.46	51.89	<u>80.72</u>	44.77	<u>65.42</u>	66.85	<u>65.35</u>	
Ours								
CloudFixer	<u>83.46</u>	53.12	77.80	43.30	63.97	67.85	64.92	
+ Voting ( $K=3$ )	<b>83.51</b>	<u>53.47</u>	77.92	43.64	64.48	<u>68.37</u>	65.23	
CloudFixer-O	80.42	53.08	75.00	<b>51.55</b>	<b>66.00</b>	<b>69.02</b>	<b>65.85</b>	



**Fig. 3:** The average accuracy of CloudFixer and other baselines across all corruptions in ModelNet40-C for various classifier architectures.

the standard evaluation setup used for conventional TTA methods. For this purpose, we use a Point2Vec on ModelNet40-C for common corruptions in Table 4 and DGCNN on PointDA-10 [50] in Table 5 for natural distribution shift. In Table 4, we evaluate the online extension CloudFixer-O for comparison. We observe that our method consistently achieves the best performance on 9 corruptions and the second best performance on 3 corruptions out of 15, thus achieving state-of-the-art performance on average also in the mild settings. This is attributed to the effectiveness of our online model adaptation strategy, as well as our input adaptation, which can be leveraged when considering the mild conditions. A similar trend is observed with natural distribution shift in Table 5.

**Across Various Classifier Architectures** We investigate adaptation performance across different architectures to verify that CloudFixer operates in an architecture-agnostic manner due to its focus on input adaptation. Specifically, we report the average adaptation accuracy of CloudFixer across all corruptions in ModelNet40-C for various baseline architectures, including Point2Vec [1], PointMAE [45], PointMLP [38], and PointNeXt [49], along with TENT [59], which shows the best performance among baselines with Point2Vec, and MEMO [67], which is the instance-based model adaptation method. As shown in Fig. 3, we observe that CloudFixer-O achieves an average performance gain ranging from at least 13% (PointNeXt) to a maximum of 27% (PointMAE and PointMLP) across various corruptions, achieving the highest performance across all architectures compared to baselines. With PointMAE, only a single instance-based CloudFixer outperforms TENT with a batch size of 64. This demonstrates the architecture-agnostic benefits of CloudFixer. The detailed performance for each corruption across various architectures can be found in Appendix I.1.

### 5.3 Ablation Study

**Parameterization, Objective, and Diffusion Timesteps** This paragraph comprehensively demonstrates whether the components of CloudFixer truly

**Table 6:** Ablation study of the core strategies in CloudFixer, including parameterization, objective, displacement regularization, forward timesteps, voting, and online adaptation.

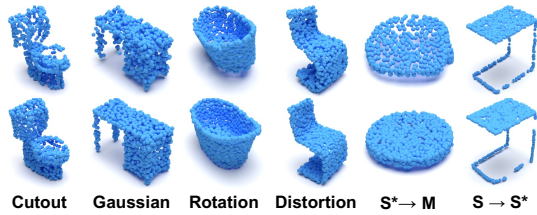
Setting	Density		Noise		Transform		Avg.
	LD	CO	US	BG	ROT	RBF	
Unadapted	20.02	83.91	38.33	37.68	47.04	74.68	51.30
No Parameterization	39.02	82.13	<u>91.81</u>	75.64	49.79	80.87	70.85
Rotation $\rightarrow$ Affine	37.88	82.50	87.16	74.96	66.05	82.66	72.59
Squared $\ell_2$	37.12	83.71	71.80	<u>76.34</u>	78.93	80.23	72.47
Diffusion Loss	33.14	73.70	86.95	39.10	59.08	70.46	62.18
No Reg.	36.26	73.26	85.17	61.30	55.71	79.05	65.27
Uniform Reg.	<u>39.30</u>	89.10	84.68	48.82	80.02	83.87	71.49
$t \sim U[0.01T, 0.02T]$	21.03	82.86	84.93	54.09	58.79	40.07	58.10
$t \sim U[0.4T, 0.5T]$	17.06	48.10	38.61	58.14	60.01	37.16	42.72
<b>CloudFixer</b>	38.82	83.06	89.06	75.49	81.04	82.98	75.79
+Voting ( $K = 5$ )	38.90	<u>84.16</u>	89.79	<b>76.58</b>	<u>83.51</u>	<u>84.12</u>	<u>76.83</u>
<b>CloudFixer-O</b>	<b>44.94</b>	<b>86.99</b>	<b>92.14</b>	74.92	<b>85.98</b>	<b>86.30</b>	<b>79.20</b>

**Table 7:** Accuracy on ModelNet40-C, using TTA baselines including MATE [42] with batch size of 1 and 64 using PointMAE.

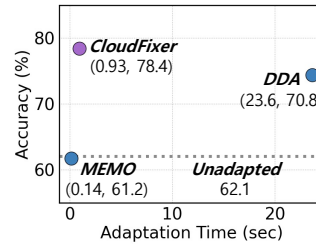
Method	OC	LD	DD	GAU	BG	IR	Avg.
Unadapted	34.72	15.19	78.65	51.30	16.87	62.03	43.13
MEMO [67]	33.18	14.79	78.08	47.57	6.81	59.97	40.07
DDA [13]	39.91	37.07	79.70	<u>87.20</u>	19.89	76.30	56.68
MATE(1) [42]	<b>52.55</b>	<b>49.85</b>	<b>82.54</b>	78.97	13.05	<u>78.44</u>	<u>59.23</u>
<b>CloudFixer</b>	34.24	35.62	71.15	87.16	<u>50.36</u>	72.45	58.50
<b>CloudFixer-O</b>	<u>47.53</u>	<u>48.95</u>	<u>81.92</u>	<b>88.98</b>	<b>59.44</b>	<b>84.76</b>	<b>68.53</b>
PL [28]	48.01	41.45	79.13	70.38	17.83	70.18	54.50
TENT [59]	47.53	41.82	79.94	73.38	17.30	71.27	55.21
SHOT [31]	<b>55.71</b>	<u>50.16</u>	72.33	67.34	14.59	67.75	54.65
SAR [44]	47.73	45.62	80.79	72.00	10.98	72.20	54.89
DUA [41]	50.85	45.83	83.14	75.24	19.17	75.41	58.27
LAME [5]	34.89	16.05	80.92	50.49	8.51	62.40	42.21
MATE(1) [42]	48.52	36.51	<u>84.40</u>	71.07	11.18	72.49	54.03
MATE(10) [42]	54.09	46.47	84.28	76.34	19.69	78.97	59.97
MATE(20) [42]	<u>55.06</u>	48.87	83.31	<u>76.42</u>	<u>21.35</u>	<u>80.19</u>	<u>60.87</u>
<b>CloudFixer-O</b>	52.55	<b>53.61</b>	<b>84.76</b>	<b>90.56</b>	<b>67.50</b>	<b>87.07</b>	<b>72.68</b>

contribute to performance enhancement and if they are optimal choices. To achieve this, we validate the core components of CloudFixer, including the parameterization of geometric transformation, objective function with chamfer distance, per-point regularization, the range of timestep ( $t_{min}$ ,  $t_{max}$ ), the voting mechanism, and the online input adaptation in Table 6. First, we confirm the suitability of our geometric transformation parameterization of rotation and displacement. Adapting the input without parameterization (No Parameterization) or substituting the rotation matrix with a more parameter-rich affine matrix (Rotation  $\rightarrow$  Affine) leads to performance degradation. Second, we validate our optimization objective. When employing the squared  $\ell_2$  loss without accounting for the unordered nature of point clouds (Squared  $\ell_2$ ), or when naively adopting the noise matching loss of the diffusion model (Diffusion Loss), both lead to a decrease in average performance. Next, we explore the range of timesteps and results indicate that maintaining a small value for  $t$  ( $t \sim U[0.01T, 0.02T]$ ) leads to a performance drop due to failure to overlap the source and target domains. Conversely, keeping a large value for  $t$  ( $t \sim U[0.4T, 0.5T]$ ) results in a significant drop in adaptation performance, likely due to loss of original content. Finally, we observe that both voting (+Voting ( $K = 5$ )) and online model adaptation (CloudFixer-O) consistently boost the performance of the original CloudFixer. Further discussions regarding the ablation study can be found in Appendices I.2 and I.3.

**Batch Size of CloudFixer-O** We also evaluate the extended online model adaptation version, CloudFixer-O, compared to other baselines under varying batch sizes. Table 7 is separated by a double line, with the upper part displaying results for a batch size of 1 and the lower part for a batch size of 64. To include MATE [42] in this comparison, we adopt the specific architecture, PointMAE [45], utilized by MATE. ‘MATE( $n$ )’ represents  $n$ -times model updates per batch during online adaptation. Although CloudFixer-O is designed for mild conditions, we find



**Fig. 4:** Point cloud visualization examples demonstrate the effects of CloudFixer on various common corruption types in ModelNet40-C and natural distributions in PointDA-10. The upper row showcases corrupted examples, while the lower row illustrates the corresponding results after applying CloudFixer.



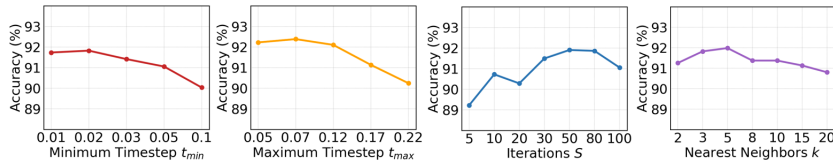
**Fig. 5:** Computational cost of CloudFixer and other per-sample episodic TTA baselines by averaging the adaptation time across all corruption types within the ModelNet40-C.

that it also performs well for a batch size of 1, achieving the best average accuracy even compared to the baselines with a batch size of 64. Additionally, MATE performs better than instance-based CloudFixer but worse than CloudFixer-O with a batch size of 1. Moreover, the enhancement in performance with a larger batch size is limited for MATE, in contrast to CloudFixer-O. To be specific, the performance of MATE appears to saturate at approximately 61%, even as the number of update steps increases. On the other hand, CloudFixer-O exhibits a notable improvement, achieving an average accuracy of 72.68% with more than a 4% increase in accuracy points as the batch size increases from 1 to 64.

#### 5.4 Further Analysis

**Qualitative Analysis** In this section, we assess whether the performance improvement observed in CloudFixer is incidental or can be attributed to the actual transformation of shifted test instances into the source domain through CloudFixer’s input adaptation strategy. To investigate this, we visually examine point clouds before and after adaptation across different target domains, as illustrated in Fig. 4. Additionally, we present more adaptation examples for all common corruptions in Fig. J4 to Fig. J6. As depicted in Fig. 4, we observe that CloudFixer successfully restores corrupted target instances for each common corruption—Cutout, Gaussian, Rotation, and Distortion—back to clean instances from the source domain of ModelNet40. It is worth noting that this phenomenon extends to natural distribution shifts as well; irregular point clouds from ScanNet are transformed into regular patterns resembling those from ModelNet ( $S^* \rightarrow M$ ), while conversely, the legs of a desk from ShapeNet are converted into the irregular form of ScanNet ( $S \rightarrow S^*$ ). We also provide more adaptation examples in Appendix J.

**Computational Efficiency** To assess the real-time feasibility of CloudFixer, we conduct a comparison with per-sample episodic TTA baselines (MEMO, DDA).



**Fig. 6:** Hyperparameter sensitivity analysis regarding  $(t_{min}, t_{max})$  for the forward process, the number of adaptation steps  $S$ , and the number of nearest neighbors  $k$  used in displacement regularization using Point2Vec on IP of ModelNet40-C.

The assessment involves averaging the adaptation time across all corruption types within the ModelNet40-C, utilizing a single RTX 3090 GPU. CloudFixer achieves an acceptable adaptation speed within one second (0.93 seconds on average) with a moderate number of iterations and the absence of backpropagation through the diffusion model or extensive iterative generation. This stands in stark contrast to DDA, which also utilizes the same diffusion model architecture as CloudFixer but takes approximately 23.6 seconds, making it prohibitively costly for practical real-time usage. Moreover, CloudFixer can take the advantage of batch processing. Combining this observation with our hyperparameter sensitivity experiments, we anticipate further reduction in adaptation time by employing a number of iterations smaller than 30.

**Hyperparameter Sensitivity** To evaluate hyperparameter sensitivity, we conduct an analysis involving the minimum and maximum timesteps  $t_{min}, t_{max}$  for the forward process, the number of adaptation steps  $S$ , and the number of nearest neighbors  $k$  used in displacement regularization using Point2Vec on ‘IP’ of ModelNet40-C. While keeping the remaining hyperparameters fixed as best with  $(t_{min}, t_{max}, S, k) = (0.02, 0.12, 30, 5)$ , we vary one at a time. Considering the unadapted performance at 66.25%, as depicted in Fig. 6, CloudFixer illustrates the establishment of a sweet spot for all critical hyperparameters, indicating its insensitivity to hyperparameters. This highlights a significant advantage in utilization during inference time, where hyperparameter optimization is infeasible due to uncertainty about potential distribution shifts.

## 6 Conclusion

In this paper, we have introduced CloudFixer, a novel test-time input adaptation method for 3D point clouds. Leveraging a pre-trained diffusion model’s domain translation capability, CloudFixer directly optimizes carefully designed geometric transformation parameters to translate input point clouds, taking into account computational costs. Extensive experiments show that our method achieves state-of-the-art results across diverse distribution shift scenarios. Our approach advances test-time input adaptation for 3D point cloud recognition, highlighting essential design principles that integrate geometric optimization and diffusion model knowledge.

## Acknowledgments

This work was supported by the Institute for Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIP) (No.2019-0-00075, Artificial Intelligence Graduate School Program (KAIST)).

## References

1. Abou Zeid, K., Schult, J., Hermans, A., Leibe, B.: Point2vec for self-supervised representation learning on point clouds. In: GCPR (2023)
2. Achituve, I., Maron, H., Chechik, G.: Self-supervised learning for domain adaptation on point clouds. In: WACV (2021)
3. Alexiou, E., Upenik, E., Ebrahimi, T.: Towards subjective quality assessment of point cloud imaging in augmented reality. In: IEEE MMSP (2017)
4. Alliegro, A., Boscaini, D., Tommasi, T.: Joint supervised and self-supervised learning for 3d real world challenges. In: ICPR (2021)
5. Boudiaf, M., Mueller, R., Ben Ayed, I., Bertinetto, L.: Parameter-free online test-time adaptation. In: CVPR (2022)
6. Cardace, A., Spezialetti, R., Ramirez, P.Z., Salti, S., Di Stefano, L.: Self-distillation for unsupervised 3d domain adaptation. In: WACV (2023)
7. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015)
8. Chen, S., Liu, B., Feng, C., Vallespi-Gonzalez, C., Wellington, C.: 3d point cloud processing and learning for autonomous driving: Impacting map creation, localization, and perception. IEEE SPM (2020)
9. Choi, J., Kim, S., Jeong, Y., Gwon, Y., Yoon, S.: ILVR: conditioning method for denoising diffusion probabilistic models. In: ICCV (2021)
10. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: Scannet: Richly-annotated 3d reconstructions of indoor scenes. In: CVPR (2017)
11. Fan, H., Chang, X., Zhang, W., Cheng, Y., Sun, Y., Kankanhalli, M.: Self-supervised global-local structure modeling for point cloud domain adaptation with reliable voted pseudo labels. In: CVPR (2022)
12. Foret, P., Kleiner, A., Mobahi, H., Neyshabur, B.: Sharpness-aware minimization for efficiently improving generalization. In: ICLR (2021)
13. Gao, J., Zhang, J., Liu, X., Darrell, T., Shelhamer, E., Wang, D.: Back to the source: Diffusion-driven adaptation to test-time corruption. In: CVPR (2023)
14. Gong, S., Li, M., Feng, J., Wu, Z., Kong, L.: Diffuseq: Sequence to sequence text generation with diffusion models. In: ICLR (2023)
15. Gong, T., Jeong, J., Kim, T., Kim, Y., Shin, J., Lee, S.J.: Note: Robust continual test-time adaptation against temporal correlation. In: NeurIPS (2022)
16. Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., Bennamoun, M.: Deep learning for 3d point clouds: A survey. IEEE TPAMI (2020)
17. Hempel, T., Abdelrahman, A.A., Al-Hamadi, A.: 6d rotation representation for unconstrained head pose estimation. In: ICIP (2022)
18. Ho, J., Chan, W., Saharia, C., Whang, J., Gao, R., Gritsenko, A., Kingma, D.P., Poole, B., Norouzi, M., Fleet, D.J., et al.: Imagen video: High definition video generation with diffusion models. arXiv preprint arXiv:2210.02303 (2022)

19. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: *NeurIPS (2020)*
20. Ho, J., Salimans, T., Gritsenko, A., Chan, W., Norouzi, M., Fleet, D.J.: Video diffusion models. *arXiv preprint arXiv:2204.03458 (2022)*
21. Hooeboom, E., Satorras, V.G., Vignac, C., Welling, M.: Equivariant diffusion for molecule generation in 3d. In: *ICML (2022)*
22. Huang, C., Cao, Z., Wang, Y., Wang, J., Long, M.: Metasets: Meta-learning on point sets for generalizable representations. In: *CVPR (2021)*
23. Huang, H., Chen, C., Fang, Y.: Manifold adversarial learning for cross-domain 3d shape representation. In: *ECCV (2022)*
24. Jun, H., Nichol, A.: Shap-e: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463 (2023)*
25. Kim, C., Park, J., Shim, H., Yang, E.: SGEM: Test-time adaptation for automatic speech recognition via sequential-level generalized entropy minimization. In: *INTERSPEECH (2023)*
26. Kim, G., Shim, H., Kim, H., Choi, Y., Kim, J., Yang, E.: Diffusion video autoencoders: Toward temporally consistent face video editing via disentangled video encoding. In: *CVPR (2023)*
27. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: *ICLR (2015)*
28. Lee, D.H., et al.: Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In: *Workshop on challenges in representation learning, ICML (2013)*
29. Lehner, A., Gasperini, S., Marcos-Ramiro, A., Schmidt, M., Mahani, M.A.N., Navab, N., Busam, B., Tombari, F.: 3d-vfield: Adversarial augmentation of point clouds for domain generalization in 3d object detection. In: *CVPR (2022)*
30. Li, X., Thiekstun, J., Gulrajani, I., Liang, P.S., Hashimoto, T.B.: Diffusion-lm improves controllable text generation. In: *NeurIPS (2022)*
31. Liang, J., Hu, D., Feng, J.: Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In: *ICML (2020)*
32. Lim, H., Kim, B., Choo, J., Choi, S.: Ttn: A domain-shift aware batch normalization in test-time adaptation. In: *ICLR (2023)*
33. Liu, Y., Kothari, P., Van Delft, B., Bellot-Gurlet, B., Mordan, T., Alahi, A.: Ttt++: When does self-supervised test-time training fail or thrive? In: *NeurIPS (2021)*
34. Liu, Z., Feng, Y., Black, M.J., Nowrouzezahrai, D., Paull, L., Liu, W.: Meshdiffusion: Score-based generative 3d mesh modeling. In: *ICLR (2023)*
35. Luo, S., Hu, W.: Diffusion probabilistic models for 3d point cloud generation. In: *CVPR (2021)*
36. Lyu, Z., Kong, Z., Xu, X., Pan, L., Lin, D.: A conditional point diffusion-refinement paradigm for 3d point cloud completion. In: *ICLR (2022)*
37. Ma, C., Yang, Y., Guo, J., Pan, F., Wang, C., Guo, Y.: Unsupervised point cloud completion and segmentation by generative adversarial autoencoding network. In: *NeurIPS (2022)*
38. Ma, X., Qin, C., You, H., Ran, H., Fu, Y.: Rethinking network design and local geometry in point cloud: A simple residual mlp framework. In: *ICLR (2022)*
39. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. In: *ICLR (2018)*
40. Meng, C., Song, Y., Song, J., Wu, J., Zhu, J.Y., Ermon, S.: Sdedit: Image synthesis and editing with stochastic differential equations. In: *ICLR (2022)*
41. Mirza, M.J., Micorek, J., Possegger, H., Bischof, H.: The norm must go on: Dynamic unsupervised domain adaptation by normalization. In: *CVPR (2022)*



42. Mirza, M.J., Shin, I., Lin, W., Schriebl, A., Sun, K., Choe, J., Possegger, H., Kozinski, M., Kweon, I.S., Yoon, K.J., et al.: Mate: Masked autoencoders are online 3d test-time learners. In: ICCV (2023)
43. Nichol, A., Jun, H., Dhariwal, P., Mishkin, P., Chen, M.: Point-e: A system for generating 3d point clouds from complex prompts. arXiv preprint arXiv:2212.08751 (2022)
44. Niu, S., Wu, J., Zhang, Y., Wen, Z., Chen, Y., Zhao, P., Tan, M.: Towards stable test-time adaptation in dynamic wild world. In: ICLR (2023)
45. Pang, Y., Wang, W., Tay, F.E., Liu, W., Tian, Y., Yuan, L.: Masked autoencoders for point cloud self-supervised learning. In: ECCV (2022)
46. Poole, B., Jain, A., Barron, J.T., Mildenhall, B.: Dreamfusion: Text-to-3d using 2d diffusion. In: ICLR (2023)
47. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: CVPR (2017)
48. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: NeurIPS (2017)
49. Qian, G., Li, Y., Peng, H., Mai, J., Hammoud, H., Elhoseiny, M., Ghanem, B.: Pointnext: Revisiting pointnet++ with improved training and scaling strategies. In: NeurIPS (2022)
50. Qin, C., You, H., Wang, L., Kuo, C.C.J., Fu, Y.: Pointdan: A multi-scale 3d domain adaption network for point cloud representation. In: NeurIPS (2019)
51. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., Chen, M.: Hierarchical text-conditional image generation with clip latents. arXiv preprint arXiv:2204.06125 (2022)
52. Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., Sutskever, I.: Zero-shot text-to-image generation. In: ICML (2021)
53. Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E.L., Ghasemipour, K., Gontijo Lopes, R., Karagol Ayan, B., Salimans, T., et al.: Photorealistic text-to-image diffusion models with deep language understanding. In: NeurIPS (2022)
54. Shen, Y., Yang, Y., Yan, M., Wang, H., Zheng, Y., Guibas, L.J.: Domain adaptation on point clouds via geometry-aware implicit. In: CVPR (2022)
55. Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. In: ICLR (2021)
56. Song, Y., Sohl-Dickstein, J., Kingma, D.P., Kumar, A., Ermon, S., Poole, B.: Score-based generative modeling through stochastic differential equations. In: ICLR (2021)
57. Sun, J., Zhang, Q., Kailkhura, B., Yu, Z., Xiao, C., Mao, Z.M.: Benchmarking robustness of 3d point cloud recognition against common corruptions. arXiv preprint arXiv:2201.12296 (2022)
58. Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A., Hardt, M.: Test-time training with self-supervision for generalization under distribution shifts. In: ICML (2020)
59. Wang, D., Shelhamer, E., Liu, S., Olshausen, B., Darrell, T.: Tent: Fully test-time adaptation by entropy minimization. In: ICLR (2021)
60. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. In: ACM TOG (2019)
61. Wei, X., Gu, X., Sun, J.: Learning generalizable part-based feature representation for 3d point clouds. In: NeurIPS (2022)
62. Wu, Z., Song, S., Khosla, A., Tang, X., Xiao, J.: 3d shapenets for 2.5 d object recognition and next-best-view prediction. arXiv preprint arXiv:1406.5670 (2014)
63. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: CVPR (2015)

64. Xiao, H., Cheng, M., Shi, L.: Learning cross-domain features for domain generalization on point clouds. In: PRCV (2022)
65. You, F., Li, J., Zhao, Z.: Test-time batch statistics calibration for covariate shift. In: ICLR (2022)
66. Zeng, X., Vahdat, A., Williams, F., Gojcic, Z., Litany, O., Fidler, S., Kreis, K.: Lion: Latent point diffusion models for 3d shape generation. In: NeurIPS (2022)
67. Zhang, M., Levine, S., Finn, C.: Memo: Test time robustness via adaptation and augmentation. In: NeurIPS (2022)
68. Zhao, M., Bao, F., Li, C., Zhu, J.: Egsde: Unpaired image-to-image translation via energy-guided stochastic differential equations. In: NeurIPS (2022)
69. Zhou, L., Du, Y., Wu, J.: 3d shape generation and completion through point-voxel diffusion. In: ICCV (2021)
70. Zou, L., Tang, H., Chen, K., Jia, K.: Geometry-aware self-training for unsupervised domain adaptation on object point clouds. In: ICCV (2021)