Appendix

A Algorithm of CloudFixer

We present the detailed algorithm of CloudFixer in Algorithm 1, which encompasses both the original input adaptation and the online model adaptation. It is worth noting that the pre-trained diffusion model ϵ_θ is frozen and utilized solely to acquire $\hat{y}$, guiding y back to the source domain.

Algorithm 1 CloudFixer

Require: Test instance $x \in \mathbb{R}^{N\times 3}$, Diffusion model ϵ_θ, Classifier f_ψ, Timestep schedule (t_{min}, t_{max}), # of iteration S, Regularization scheduling $\lambda(\cdot)$, Learning rate scheduling $\eta_{\texttt{input}}(\cdot), \eta_{\texttt{model}}(\cdot)$, # of nearest neighbor k, # of vote K

▷ Diffusion-Guided Input Adaptation

$\Delta \leftarrow \mathbf{0},\ \ R \leftarrow I,\ \ m \leftarrow 0$

while $m < K$ **do** ▷ Conducted as batch processing

  $n \leftarrow 0$

  while $n < S$ **do**

    $y[m] \leftarrow R(x + \Delta)$

    $t,\ \epsilon \leftarrow U[t_{min}, t_{max}],\ \mathcal{N}(0, I)$

    $y_t \leftarrow \alpha_t y[m] + \sigma_t \epsilon$

    $\hat{y} \leftarrow (y_t - \sigma_t \epsilon_\theta(y_t, t))/\alpha_t$

    $w_i \leftarrow 1/\sum_{j\in kNN(i)} \|x_i - x_j\|_2,\ \forall i \in \{1, \ldots, N\}$

    $w_i \leftarrow w_i/\sum_j w_j,\ \forall i \in \{1, \ldots, N\}$

    $\mathcal{L}_{\texttt{input}} \leftarrow D(\texttt{stopgrad}(\hat{y}), y[m]) + \lambda(n)\sum_j w_j \|\delta_j\|_2^2)$

    $(\Delta, R) \leftarrow (\Delta, R) - \eta_{\texttt{input}}(n)\nabla_{(\Delta,R)}\mathcal{L}_{\texttt{input}}$ ▷ AdaMax in our implementation

    $n \leftarrow n + 1$

  end while

  $m \leftarrow m + 1$

end while

▷ Online Model Adaptation (CloudFixer-O. Optional)

$o \leftarrow 0,$

while $o < O$ **do**

  $\mathcal{L}_{\texttt{model}} \leftarrow \sum_{m=1}^{K} KL(f_\psi(x), f_\psi(y[m]))$

  $\psi \leftarrow \psi - \eta_{\texttt{model}}(o)\nabla_\psi \mathcal{L}_{\texttt{model}}$ ▷ AdamW in our implementation

  $o \leftarrow o + 1$

end while

B Originality and Advantages over DDA

While DDA [13] and CloudFixer both use diffusion models for input adaptation, their core ideas and methods are distinct. DDA uses an iterative generative process while preserving low-frequency information, whereas CloudFixer uses a parameterized 3D geometric transformation guided by the source diffusion

model. Consequently, *DDA struggles with misalignment and structural corruption*, leading to *significant performance differences*, especially in BG and ROT. DDA also requires more iterative steps and backpropagation through diffusion models, making it *over 25 times slower than CloudFixer* (**??**). Last but not least, the online model adaptation utilizing the consistency proposed by us is a distinctive feature of our model, clearly distinguished by DDA.

C Limitations and Broader Impacts

Limitations One prevalent test-time corruption for 3D point clouds is occlusion, where the input data is incomplete. When the source domain contains clean point clouds and the target domain involves occluded point clouds, such domain translation problem becomes a point cloud completion task. Since the pre-trained diffusion model operates on normalized point clouds, and normalizing severely occluded point clouds to be zero-centered with a unit standard deviation results in a significant scale and center location shift for clean point clouds, this presents a complex research problem. For example, in the case of an occluded chair with only the backrest visible, the backrest's scale may increase, shifting downward and potentially causing misclassification as a monitor. Achieving accurate translation for complete chairs requires substantial point movement. However, since CloudFixer regularizes large steps, our model currently has limitations in addressing this specific corruption type. Developing TTA methods to effectively handle such corruption stands as a promising future research direction.

Broader Impacts The increasing demand for computer vision, particularly in 3D vision applications like autonomous systems and virtual reality, has stimulated significant research into domain adaptation methods to address distribution shifts. Test-time adaptation holds the potential to enhance the performance of 3D perception models in autonomous systems, encompassing applications such as self-driving cars, drones, and robotics. In this regard, the application of our method allows for the translation of test data into a source domain format without the need for source data in on-device settings, thereby conducting input adaptation. This capability is poised to have a significant impact on the evolving field of 3D vision, fostering further growth in the field. We hope this research provides valuable insights to the academic community and enhances the robustness of point cloud recognition models in real-world scenarios.

D Reproducibility Statement

For reproducibility, we provide our implementation code in our github repository. We supply the bash scripts `scripts/train_dm.sh` for training a diffusion model on ModelNet40 and `scripts/run_cloudfixer.sh` for adaptation on ModelNet40-C using CloudFixer and `scripts/run_baselines.sh` for adaptation on ModelNet40-C using

baselines, respectively. Our hyperparameter optimization results for TTA baselines, which are elucidated in Appendix H can be found in `cfgs/hparams`. For a pair comparison and to ensure reproducibility, we consistently set the random seed to 2 for all conducted experiments. All hyperparameters and associated details are in our source code. Lastly, checkpoints for ModelNet40 classifiers can be downloaded in the following links for each architecture: Point2Vec, PointMLP, PointNeXt, and PointMAE. We refer readers to `README.md` for the remaining details.

E Dataset Details

E.1 ModelNet40-C

Fig. J2 showcases 15 corruptions of ModelNet40-C [57] on which we conduct the experiments. ModelNet40-C involves applying synthetic corruptions to the test set of ModelNet40 which comprises 2468 point clouds of 40 classes. ModelNet40-C contains broad corruption types categorized into three classes—density corruptions, noise corruptions, and transformation corruptions. For all experiments, we use severity level 5. We refer the readers to the official repository.

E.2 PointDA-10

PointDA-10 [50] incorporates natural distribution shifts in real-world scenarios. This dataset is originally proposed to serve as a benchmark for unsupervised domain adaptation by using two out of the three datasets—ModelNet [63], ShapeNet [7], and ScanNet [10]—as the source and target domains. The benchmark is composed of 10 common classes shared across the three datasets. ModelNet and ShapeNet are synthetically generated from 3D CAD models, while ScanNet is created by scanning real-world scenes. Their class distributions are not even but imbalanced. Examples are provided in Fig. J3.

F Baseline Details

PL Pseudo-Labeling (PL) [28] conducts pseudo-labeling based on the model's predictions and updates the model weights at test-time through cross-entropy loss. PL involves two crucial test-time hyperparameters: learning rate and adaptation steps.

TENT Test ENTropy minimization (TENT) [59] is a test-time adaptation method in which the model undergoes fine-tuning using an unsupervised loss with the objective of reducing the prediction entropy of the model. TENT entails two important test-time hyperparameters: learning rate and adaptation steps.

SHOT Source HypOthesis Transfer (SHOT) [31] is a fully test-time adaptation method that employs unsupervised objectives such as entropy minimization, diversity maximization, and self-supervised pseudo-labeling. It also updates the statistics of batch normalization layers using the statistics of the test batch. SHOT incorporates three critical test-time hyperparameters: learning rate, adaptation steps, and the pseudo-labeling loss weight, to adjust the relative weight of the loss induced by self-supervised pseudo-labeling.

SAR Sharpness-Aware and Reliable entropy minimization (SAR) [44] utilizes entropy minimization, but it excludes instances with high prediction entropy to prevent model collapse and incorporates the Sharpness-Aware Minimization (SAM) [12], fostering the model's adaptation to flat minima. SAR has four essential test-time hyperparameters: learning rate, adaptation steps, an entropy threshold to exclude adaptation from high entropy samples, and an epsilon threshold employed in the calculation of sharpness in SAM [12].

DUA Dynamic Unsupervised Adaptation (DUA) [41] is a test-time adaptation method that focuses on calibrating batch normalization statistics. At the onset of test time, all batch normalization layers are set to a trained state. For each incoming test batch, the model undergoes multiple forward passes, iteratively updating batch normalization statistics as moving averages. DUA utilizes two test-time hyperparameters: adaptation steps and a decay factor, which dictates the rate at which the moving average progresses.

LAME Laplacian Adjusted Maximum-likelihood Estimation (LAME) [5] operates by updating the model's prediction output rather than the model parameters. It conducts on a batch of given test data, adjusting the probabilities of similar samples in the feature space to be similar when the original prediction probabilities for the test data batch are provided. LAME is characterized by three crucial test-time hyperparameters: the kernel affinity, used to define the kernel density function, the number of neighbors to be regarded as similar, and the maximum number of steps for iterative output probability optimization.

MEMO Marginal Entropy Minimization with One test point (MEMO) [67] is a single-instance TTA technique. It utilizes multiple data augmentations to generate multiple predictions for the test instance. Subsequently, it minimizes the entropy of the average probability of these predictions. This approach not only achieves the effect of traditional entropy minimization but also encourages similarity among predictions from different viewpoints. MEMO is defined by three key test-time hyperparameters: learning rate, adaptation steps, and the number of augmentations that can be applied to each test instance.

DDA Diffusion-Driven Adaptation (DDA) [13] is a single-instance test-time adaptation for 2D image classification tasks using a diffusion model, similar to our approach. DDA is characterized by two essential test-time hyperparameters: guidance weight, which determines the extent to which the original content is preserved, and the low-pass filtering scale. Unlike our method, DDA is a generation-based method that employs low-pass filtering in the reverse process to preserve class information after the forward process. In contrast to this approach, our method achieves significantly faster execution times by avoiding back-propagation through the diffusion model and delivers superior performance by leveraging optimization-based techniques tailored for 3D tasks.

MATE MATE [42] is a test-time training approach for 3D point clouds, enhancing deep network robustness in point cloud classification against distribution shifts. It employs a masked autoencoder test-time objective, constructing batches of 48 from single corrupted point clouds, randomly masking 90% of each sample, and fine-tuning

the model using reconstruction loss. MATE efficiently adapts with minimal fractions of points from each test sample, sometimes as low as 5%, making it lightweight for real-time applications. However, MATE necessitates manipulation of the training procedure and relies on the specific architecture of PointMAE for self-reconstruction.

G Further Implementation Details

Pre-processing Before passing point clouds to classifiers, we conduct zero centering and scale it to a unit ball, adhering to the original configuration outlined in [60] for each point cloud. In our pre-processing step for inputs of diffusion models, following zero centering, we standardize the point clouds to achieve a unit variance following [66].

Pre-training Diffusion Models For 4 source datasets—ModelNet40-C, ShapeNet, ModelNet, and ScanNet—we use the same setting as follows. We use the polynomial noise scheduling used in [21] and set the total number of timesteps of the diffusion model as $T = 500$. The diffusion model is trained for 5000 epochs on each dataset with an exponential moving average (EMA) decay of 0.9999 and a batch size of 64.

Baselines For model adaptation methods (PL, TENT, SHOT, SAR, and MEMO), we also update batch normalization statistics by initializing all layers to a trained state at the onset of the inference phase. PL, TENT, and SAR exclusively fine-tune the affine parameters of batch normalization statistics following the default settings of previous works [44, 59], while SHOT and MEMO optimize all parameters. For PL, TENT, SHOT, and SAR, we configure all settings to be online, where the model is not reset to the pre-trained state upon receiving the test batch. In contrast, for LAME, MEMO, and DDA, we set all settings to be episodic, where the model is reset to the pre-trained state every time a test batch is received. We adopt a standard batch size of 64 for the online TTA baseline method, following common practice [31, 59], unless otherwise specified. Notably, our proposed method, CloudFixer, along with per-sample TTA baselines (MEMO, DDA), can operate with a batch size of 1.

CloudFixer With regard to our input adaptation method, we employ a consistent configuration across all datasets. We conduct 30 iterations of updates utilizing the AdaMax optimizer [27] with a learning rate that linearly increases for 6 steps (20% of warmup for total steps) from 0 to 0.2 and then linearly decreases to 0.01 for the remaining steps. The timestep interval for the diffusion forward process is defined as $[t_{min}, t_{max}]$ with $t_{min} = 0.02T$ and $t_{max} = 0.12T$. For the computation of weights w_{jj} to regulate δ_j, we designate the number of nearest neighbors as $k = 5$. For regularization, $\lambda(\cdot)$ is initialized to 10 and cosine annealed to 1 for the 30 steps.

CloudFixer-O Under mild conditions, we propose to adapt model parameters as well as inputs. Model adaptation is conducted in an online manner. With a batch size of 64, we update models one time per batch using AdamW with a learning rate 10^{-5} and 10^{-4} on ModelNet40-C and PointDA-10, respectively. For each instance, we obtain 3 different transformations. Furthermore, we report the results of CloudFixer-O with a batch size of 1 on ModelNet40-C using PointMAE. In this case, we update models

Table H1: Hyperparameter search space of test-time adaptation baseline methods.

Method	Hyperparameter Search Space
PL [28]	learning rate: $\{10^{-4}, 10^{-3}, 10^{-2}\}$, adaptation steps: $\{1, 3, 5, 10\}$
TENT [59]	learning rate: $\{10^{-4}, 10^{-3}, 10^{-2}\}$, adaptation steps: $\{1, 3, 5, 10\}$
SHOT [31]	learning rate: $\{10^{-4}, 10^{-3}, 10^{-2}\}$, adaptation steps: $\{1, 3, 5, 10\}$, pseudo-labeling loss weight: $\{0, 0.1, 0.3, 0.5, 1\}$
SAR [44]	learning rate: $\{10^{-4}, 10^{-3}, 10^{-2}\}$, adaptation steps: $\{1, 3, 5, 10\}$, entropy threshold: $\{0.2, 0.4, 0.6, 0.8\}$, epsilon threshold: $\{0.01, 0.05, 0.1\}$
DUA [41]	adaptation steps: $\{1, 3, 5, 10\}$, decay factor: $\{0.9, 0.94, 0.99\}$
LAME [5]	kernel affinity: $\{\text{rbf}, \text{kNN}, \text{linear}\}$, # of neighbors: $\{1, 3, 5, 10\}$, max steps: $\{1, 10, 100\}$
MEMO [67]	learning rate: $\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}\}$, adaptation steps: $\{1, 2\}$, # of augmentations: $\{16, 32, 64\}$
DDA [13]	guidance weight: $\{3, 6, 9\}$, low pass filtering scale: $\{2, 4, 6\}$

one time per batch using AdamW with a learning rate 10^{-6}. We use a smaller learning rate because the update step increases as a batch size decreases. For each instance, we obtain 48 different transformations following MATE [42].

H Hyperparameter Optimization

Due to the well-known sensitivity of hyperparameters in TTA methods [13, 44], we conduct rigorous hyperparameter optimization when reproducing the baselines. As optimizing hyperparameters for each test dataset is unfeasible in real-world scenarios, we tune the hyperparameters of each TTA method using the original test set of the source classifier. Generally, the batch size is a critical hyperparameter in test-time adaptation, and while increasing it typically improves adaptation performance, it cannot be increased indefinitely given the online nature of the practical test-time situations. Therefore, we initially set the batch size to 64 following the standard setting [13, 44]. Subsequently, we optimize the hyperparameters of each method by conducting a random search with a maximum iteration of 30. Our hyperparameter search space is reported in Table H1.

I Additional Experiments

I.1 Results for All Corruption Types Across Various Architectures

In this subsection, we present the adaptation results under mild conditions, employing a batch size of 64 and i.i.d. test stream, which is an extension of the findings depicted in **??**. We consider three additional architectures: PointMLP [38] in Table I1, PointNeXt in Table I2, and PointMAE [45] in Table I3. Note here that we maintain the same test-time hyperparameters used in the Point2Vec experiments. We find that CloudFixer consistently shows its superiority regardless of architecture. Moreover, CloudFixer-O also brings successful performance enhancement across diverse classifier architectures.

I.2 Ablation Study on All Corruption Types

In our main paper, we partially report the ablation studies through **??** due to space constraints. We also show the ablation results on all corruptions of ModelNet40C in

Table I1: Accuracy on ModelNet40-C using PointMLP under the mild conditions of a batch size of 64 and an i.i.d. test stream except for DDA, MEMO, and CloudFixer which operate with a batch size of 1.

	Method	Density Corruptions					Noise Corruptions					Transformation Corruptions					Avg.
		OC	LD	DI	DD	CO	UNI	GAU	IP	US	BG	ROT	SH	FFD	RBF	IR	
	Unadapted	29.01	6.11	87.92	79.70	79.13	53.12	37.96	36.53	20.82	7.24	48.37	85.37	79.21	74.55	77.87	53.53
Online	PL [28]	31.04	17.54	68.35	69.85	55.63	65.56	61.02	58.51	54.62	65.48	51.05	69.89	73.26	68.72	63.33	58.26
Online	TENT [59]	**50.36**	31.69	**88.82**	**85.58**	**85.25**	84.89	80.11	85.01	80.63	78.08	80.79	**87.32**	**85.13**	85.41	85.90	78.33
Online	SHOT [31]	44.53	31.32	80.06	79.09	77.51	74.23	74.84	73.14	72.85	71.84	77.19	78.53	78.00	79.70	79.86	71.51
Online	SAR [44]	44.77	28.48	31.60	43.44	30.02	30.39	38.86	43.96	76.09	18.35	49.47	29.38	46.19	52.51	42.10	40.37
Online	DUA [41]	47.29	30.43	88.74	84.56	83.79	82.62	78.65	81.56	78.28	67.99	76.09	85.98	83.55	83.23	84.20	75.80
Episodic	LAME [5]	28.89	5.71	88.05	80.59	79.94	52.92	36.67	34.76	19.89	8.55	48.26	85.78	79.94	75.81	77.92	53.58
Episodic	MEMO [67]	27.23	4.66	83.18	77.47	76.54	44.25	30.31	28.32	13.94	6.16	45.26	79.58	75.36	71.19	74.43	49.19
Episodic	DDA [13]	37.56	26.13	88.70	80.55	84.44	88.86	86.26	87.16	82.17	26.13	51.46	84.36	81.28	82.25	84.16	71.43
Episodic	**CloudFixer**	34.16	32.86	84.68	75.61	78.57	87.84	87.40	88.94	86.39	67.63	75.73	81.73	78.16	78.93	81.56	74.68
Episodic	+ **Voting** ($K = 3$)	34.56	33.79	85.70	77.23	80.11	89.10	87.97	89.83	87.80	69.45	80.67	82.78	79.66	80.79	82.86	76.15
Episodic	**CloudFixer-O**	44.69	**38.65**	88.25	82.78	84.97	**90.40**	**89.18**	**89.91**	**90.32**	**82.66**	**85.33**	85.98	83.47	**85.86**	**87.56**	**80.67**

Table I2: Accuracy on ModelNet40-C using PointNeXt under the mild conditions of a batch size of 64 and an i.i.d. test stream except for DDA, MEMO, and CloudFixer which operate with a batch size of 1.

	Method	Density Corruptions					Noise Corruptions					Transformation Corruptions					Avg.
		OC	LD	DI	DD	CO	UNI	GAU	IP	US	BG	ROT	SH	FFD	RBF	IR	
	Unadapted	41.41	27.95	87.84	86.18	86.06	69.12	57.90	70.58	77.02	50.77	42.50	79.01	76.45	75.04	77.55	67.03
Online	PL [28]	51.58	45.99	88.90	**87.36**	86.87	86.51	84.81	86.91	87.40	79.13	76.78	**84.32**	83.18	83.91	85.21	79.92
Online	TENT [59]	49.68	44.85	89.22	86.91	87.20	85.25	84.08	85.45	86.83	76.50	70.66	82.74	81.73	82.41	83.51	78.47
Online	SHOT [31]	**54.13**	**50.28**	88.90	87.24	87.20	83.55	82.86	86.51	85.58	80.27	77.96	83.10	**84.85**	84.16	84.52	80.07
Online	SAR [44]	48.87	38.82	87.76	84.85	85.78	84.48	83.83	84.28	86.02	73.26	69.00	81.20	79.21	80.83	82.05	76.68
Online	DUA [41]	49.35	44.45	88.90	87.20	87.64	84.93	83.71	86.18	86.75	76.82	70.54	82.94	81.69	82.70	83.87	78.51
Episodic	LAME [5]	41.13	28.04	88.09	86.47	85.66	70.18	57.90	70.91	77.11	51.78	42.75	79.09	76.54	75.00	77.88	67.24
Episodic	MEMO [67]	8.83	3.12	34.40	24.88	23.10	32.13	29.46	30.51	39.10	7.13	20.79	33.10	29.21	25.85	27.07	24.58
Episodic	DDA [13]	41.41	37.84	**90.11**	85.13	**87.76**	89.55	89.06	88.98	90.32	58.43	45.06	78.00	79.29	82.01	82.86	75.05
Episodic	**CloudFixer**	41.65	35.01	81.93	77.96	83.23	87.60	87.80	88.49	88.94	76.42	71.47	74.96	76.46	80.96	83.43	75.75
Episodic	**CloudFixer-O**	44.81	41.69	86.14	83.79	85.41	**89.63**	**89.99**	**89.71**	**91.09**	**85.13**	**81.73**	81.24	80.51	**84.76**	**86.63**	**80.15**

Table I4. As in **??**, the originally proposed setting, CloudFixer, achieves the best performance on average even when considering all corruptions. Similar to the ablation study in the main paper **??**, this consistently validates the significant performance improvement contributed by various components of CloudFixer, including geometric transformation parameterization, objective function with chamfer distance, per-point regularization, timestep range (t_{min}, t_{max}), voting mechanism, and online input adaptation, affirming their optimality. However, we can observe that ablated settings demonstrate improved performance in certain cases. For example, in the case of Shear (SH) corruption, utilizing affine transformation instead of rotation yields the best performance. This observation can be attributed to the original definition of Shear transformation which falls within affine transformations.

I.3 Further Ablation Study on Distance Metrics

We propose using Chamfer distance as our distance metric between a point cloud undergoing adaptation and an estimate from the diffusion model. This choice is motivated by the unordered nature of the points in point clouds. The other possible choice is to naively use squared ℓ_2 distance which is used to train our diffusion model. We present various adaptation examples in Fig. J1, using two different distance metrics.

Table I3: Accuracy on ModelNet40-C using PointMAE under the mild conditions of a batch size of 64 and an i.i.d. test stream except for MATE (bsz. 1), DDA, MEMO, CloudFixer, and CloudFixer-O (bsz. 1) which operate with a batch size of 1.

	Method	Density Corruptions					Noise Corruptions					Transformation Corruptions					Avg.
		OC	LD	DI	DD	CO	UNI	GAU	IP	US	BG	ROT	SH	FFD	RBF	IR	
	Unadapted	34.72	15.19	85.53	78.65	74.64	61.35	51.30	55.47	51.01	16.87	31.56	67.02	61.87	58.31	62.03	53.70
Online	PL [28]	48.01	41.45	80.83	79.13	78.44	73.74	70.38	68.44	69.57	17.83	50.85	69.81	68.68	69.25	70.18	63.77
Online	TENT [59]	47.53	41.82	82.58	79.94	79.34	74.15	73.38	69.65	69.49	17.30	51.26	71.39	69.08	70.62	71.27	64.59
Online	SHOT [31]	**55.71**	50.16	73.66	72.33	71.35	70.71	67.34	66.73	67.10	14.59	53.36	64.71	65.11	68.31	67.75	61.93
Online	SAR [44]	47.73	45.62	82.58	80.79	79.46	75.49	72.00	69.25	69.89	10.98	51.30	70.79	70.34	70.58	72.20	64.60
Online	DUA [41]	50.85	45.83	84.76	83.14	82.70	77.88	75.24	73.46	73.01	19.17	55.79	74.31	72.89	73.54	75.41	67.87
Online	MATE (bsz. 1) [42]	52.55	49.85	85.66	82.54	80.63	82.86	78.97	73.62	60.29	13.05	57.54	75.89	75.53	76.05	78.44	68.23
Online	MATE (bsz. 64) [42]	55.06	48.87	87.24	83.31	83.87	80.55	76.42	69.57	60.78	21.35	59.72	79.29	77.67	78.93	80.19	69.52
Episodic	LAME [5]	34.89	16.05	86.55	80.92	78.32	62.68	50.49	52.92	52.23	8.51	30.88	68.03	62.52	59.56	62.40	53.80
Episodic	MEMO [67]	33.18	14.79	85.01	78.08	76.78	59.16	47.57	49.23	48.06	6.81	29.90	67.46	60.70	56.97	59.97	51.58
Episodic	DDA [13]	39.91	37.07	87.07	79.70	81.16	87.60	87.20	88.09	85.13	19.89	38.37	69.98	70.83	73.70	76.30	68.13
Episodic	**CloudFixer**	34.24	35.62	79.62	71.15	73.54	86.71	87.16	87.72	82.25	50.36	54.05	65.40	68.60	70.66	72.45	67.97
Episodic	**CloudFixer-O (bsz. 1)**	47.53	48.95	85.01	81.52	83.31	88.41	88.98	88.25	88.41	59.44	75.36	75.49	78.48	84.08	84.76	77.20
Episodic	**CloudFixer-O (bsz. 64)**	52.55	**53.61**	**88.17**	**84.76**	**86.91**	**89.91**	**90.56**	**90.48**	**89.79**	**67.50**	**78.97**	**79.66**	**80.15**	**85.33**	**87.07**	**80.36**

Table I4: Ablation study conducted on all corruptions of ModelNet40C involves dissecting the core strategies in CloudFixer. This includes parameterization, objective, displacement regularization, forward timesteps, voting, and online adaptation.

Setting	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	**86.51**	83.91	63.41	49.68	66.25	38.33	37.68	47.04	79.29	75.97	74.68	77.51	62.09
No Parameterization	41.49	39.02	87.96	80.75	82.13	91.21	90.92	91.73	91.81	75.64	49.79	77.63	78.16	80.87	83.42	76.17
Rotation → Affine	40.07	37.88	86.83	79.70	82.50	90.15	89.91	90.11	87.16	74.96	66.05	**84.32**	78.32	82.66	83.59	76.95
Squared ℓ_2	40.92	37.12	90.40	84.04	83.71	84.93	82.90	88.29	71.80	76.34	78.93	81.36	76.94	80.23	82.70	76.04
Diffusion Loss	39.87	33.14	84.36	76.09	73.70	88.21	87.40	82.13	86.95	39.10	59.08	75.97	71.03	70.46	72.08	69.30
No Reg.	36.59	36.26	74.11	64.02	73.26	87.76	87.32	87.84	85.17	61.30	55.71	63.21	69.00	79.05	79.86	69.36
Uniform Reg.	41.61	39.30	89.10	82.62	84.68	91.05	90.56	84.68	87.80	48.82	80.02	79.90	79.34	83.87	85.62	76.60
$t \sim U[0.01T, 0.02T]$	41.05	21.03	89.67	83.35	82.86	86.10	83.39	81.36	84.93	54.09	58.79	80.19	45.71	40.07	40.68	64.88
$t \sim U[0.4T, 0.5T]$	26.34	17.06	45.34	42.54	48.10	55.71	54.98	62.28	38.61	58.14	60.01	51.82	36.71	37.16	39.22	44.93
CloudFixer	41.00	38.82	87.32	80.27	83.06	91.09	90.52	90.76	89.06	75.49	81.04	78.28	78.73	82.98	85.09	78.23
+ **Voting** ($K = 5$)	41.00	38.90	88.05	80.55	84.16	91.45	91.29	**91.98**	89.79	**76.58**	83.51	79.38	79.70	84.12	85.90	79.09
CloudFixer-O	**46.39**	**44.94**	**90.92**	84.76	**86.99**	**91.94**	**91.86**	91.82	**92.14**	74.92	**85.98**	83.83	**82.09**	**86.30**	**87.40**	**81.49**

The adapted results with Chamfer distance consistently appear to be clearer than the others. This indicates considering the nature of point clouds is important for stable adaptation.

I.4 Comparison with Data Augmentation

In this section, we compare CloudFixer with data augmentation strategies, which are commonly used in domain generalization. Domain generalization methods like MetaSets [22] might be effective for specific shifts like sim-to-real transfer, but they heavily rely on presumed augmentations, limiting efficacy in TTA scenarios with arbitrary target domains. Besides, CloudFixer operates at test time, complementing rather than competing with train time domain generalization. MetaSets' augmentations improve Point2Vec on ModelNet40-C (Avg.) from 62.09% to 68.75%, notably enhancing 'Density Corruptions' (*e.g.*, OC: 41.09% to 61.46%), but show no improvement for 'Transformation Corruptions.' CloudFixer-O further improves the performance to **83.05%**, handling corruptions MetaSets could not, showing domain generalization and ours are complementary.

Table I5: Accuracy on ModelNet40-C using Point2Vec under the mild conditions of a batch size of 64 and an i.i.d. test stream except for MetaSets [22], CloudFixer, and CloudFixer-O.

Method	Density Corruptions					Noise Corruptions					Transformation Corruptions					Avg.
	OC	LD	DI	DD	CO	UNI	GAU	IP	US	BG	ROT	SH	FFD	RBF	IR	
Unadapted	0.4109	0.2002	0.9003	0.8558	0.8391	0.6341	0.4968	0.6625	0.3833	0.3768	0.4704	0.7929	0.7597	0.7468	0.7751	0.6203
MetaSets [22]	**0.6146**	0.5057	**0.9340**	**0.9263**	**0.9246**	0.8132	0.6852	0.7459	0.5466	0.1102	0.4413	0.8006	0.7703	0.7318	0.7626	0.6875
+ **CloudFixer**	0.5239	0.4976	0.8995	0.9024	0.9109	0.9105	0.9080	**0.9157**	0.9055	0.6451	0.8148	0.7950	0.7978	0.8278	0.8359	0.8060
+ **CloudFixer-O**	0.5474	**0.5255**	0.9015	0.8991	0.9125	**0.9169**	**0.9182**	0.9141	**0.9169**	**0.7098**	**0.8630**	**0.8416**	**0.8432**	**0.8716**	**0.8764**	**0.8305**

I.5 Efficacy on Adversarial Attack

Another potential application of CloudFixer is its effectiveness in countering adversarial attacks. To this end, we compare the adaptation performance for adversarial examples generated using the projected gradient descent method [39] on an original clean test set of ModelNet40 using PointMLP in Table I6. The adversarial attack utilized a step size of $4 \cdot 10^{-3}$ over 30 steps, with a bound of 0.16 for each coordinate, where both the step size and the bound are measured as ℓ_∞ distance. We could compare with adversarial robustness-focused methods, but they often require specialized training. Since our primary focus is on TTA, we opt to compare with TTA methods. Some TTA methods such as TENT, SAR, and LAME fail to recover performance significantly. However, CloudFixer demonstrates substantially higher performance compared to other benchmarks, achieving accuracy levels close to the oracle. This underscores the reliability of CloudFixer in test scenarios where natural distribution shifts may occur or unintended attacks may arise.

Table I6: Accuracy of various test-time adaptation methods under adversarial attacks [39] on the original clean test set of ModelNet40 using PointMLP. We further provide the clean (oracle) accuracy without any adversarial attacks.

Method	Accuracy
Clean (Oracle)	93.84
Adversarial	11.30
PL	42.34
TENT	16.29
SHOT	35.25
SAR	11.59
DUA	15.68
LAME	11.26
MEMO	13.49
DDA	39.18
CloudFixer	**79.58**

J Adaptation Examples

We visualize originally corrupted and translated examples using CloudFixer on ModelNet40-C from Fig. J4 to J6. These examples provide additional qualitative analysis, confirming that for various severe corruption types, CloudFixer can truly translate the input point clouds to a clean source domain.

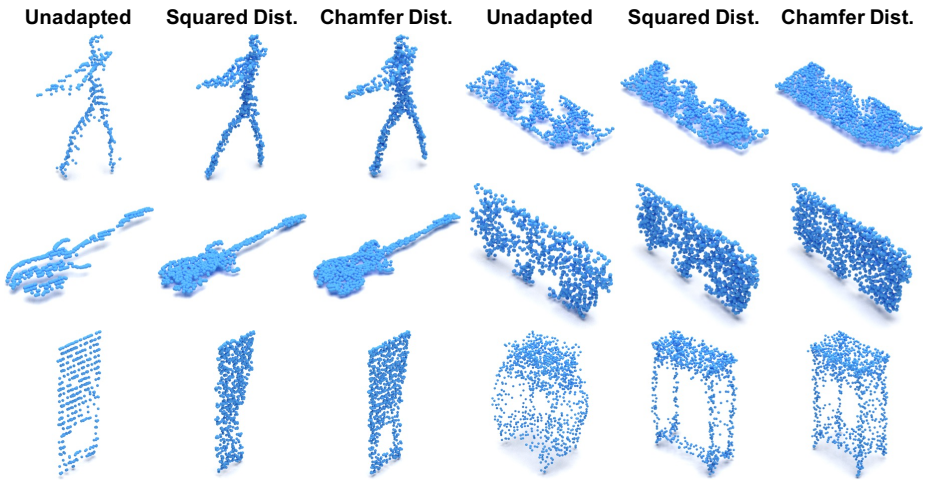

Fig. J1: Qualitative analysis of the ablation study on distance metrics with illustrative examples.

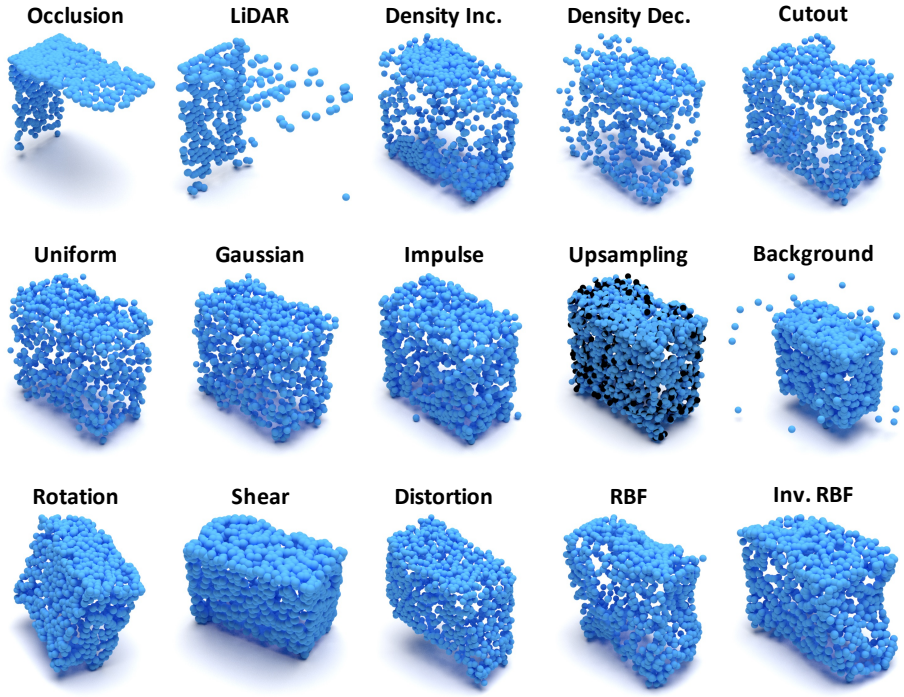

Fig. J2: An illustrative example depicting 15 distinct corruption types—Occlusion, LiDAR, Density Inc., Density Dec., Cutout, Uniform, Gaussian, Impulse, Upsampling, Background, Rotation, Shear, Distortion, RBF, Inv. RBF—for a single original point cloud associated with the dresser class in the test set of ModelNet40-C.

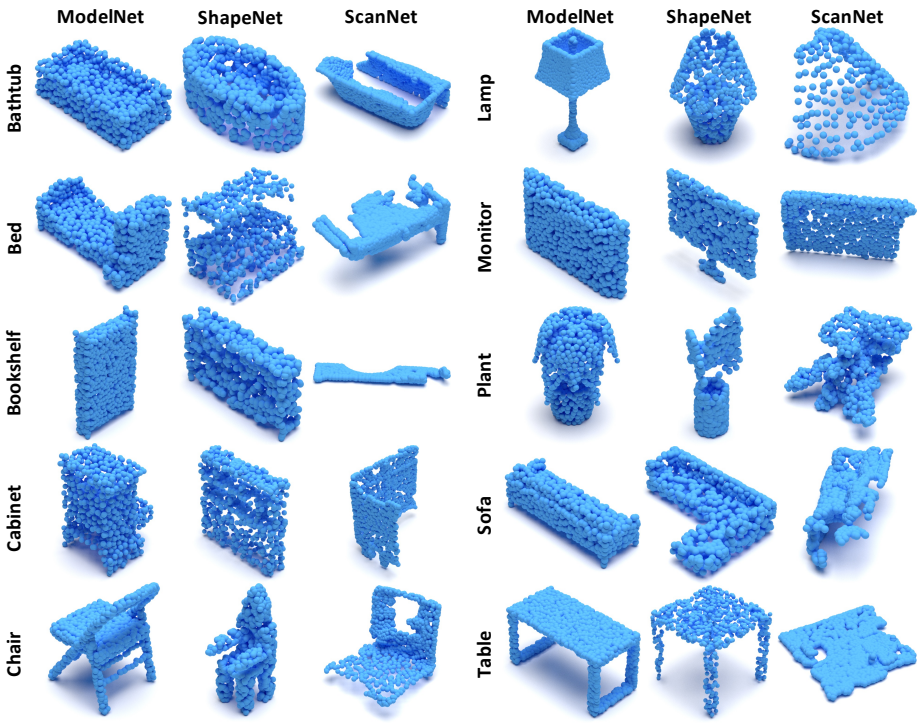

Fig. J3: An illustrative example showcasing 10 different classes—Bathtub, Bed, Bookshelf, Cabinet, Chair, Lamp, Monitor, Plant, Sofa, Table—in PointDA-10.

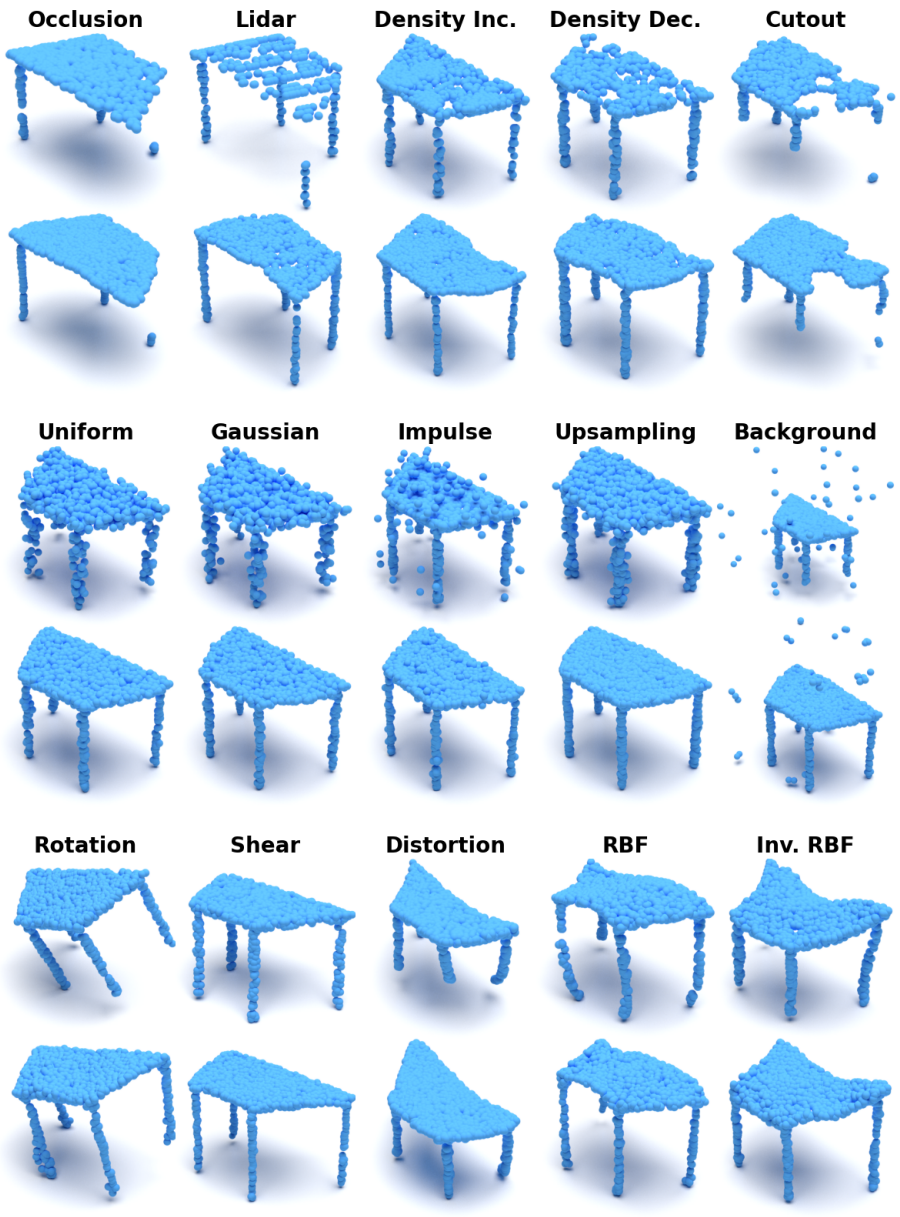

Fig. J4: Comparison between the corrupted and adapted point clouds following the application of CloudFixer to 15 different corruption types. The example pertains to a single original point cloud belonging to the desk class in the test set of ModelNet40-C.

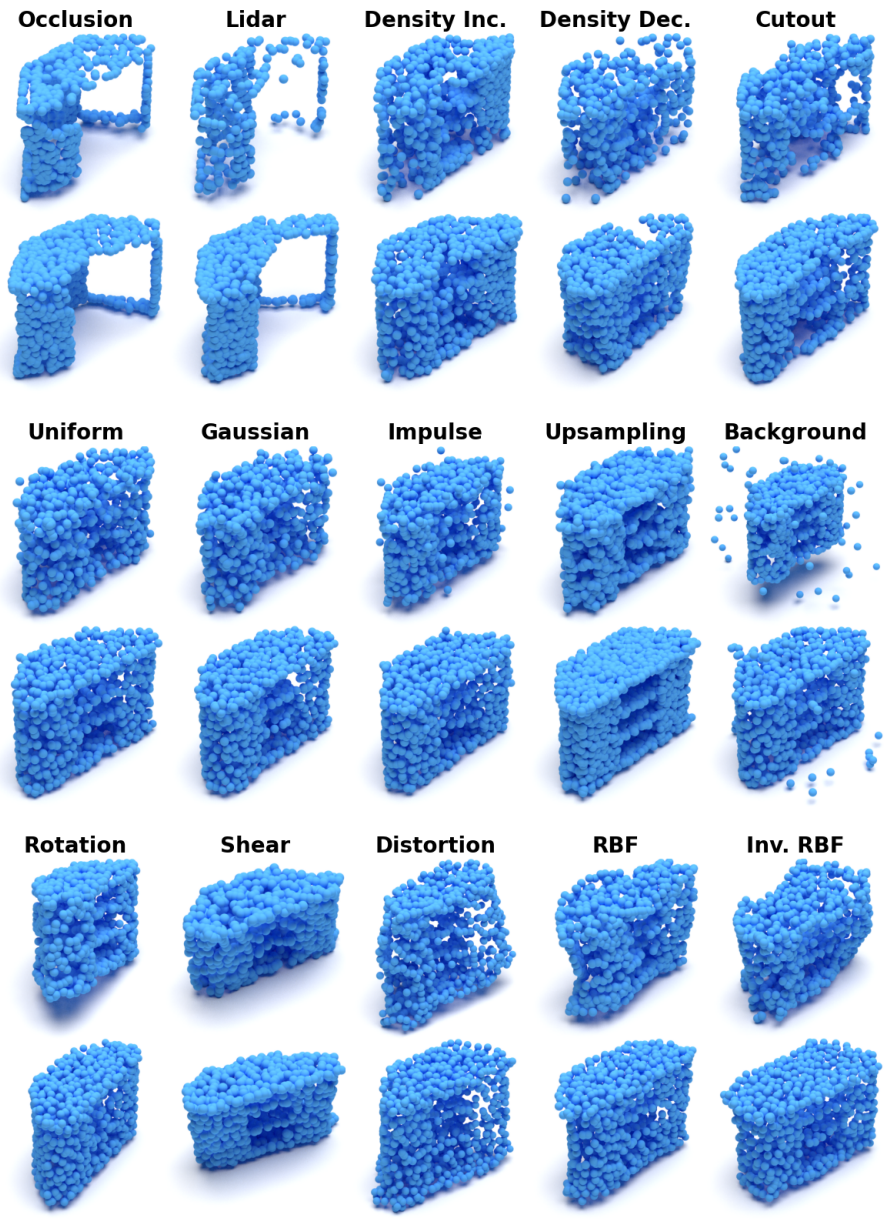

Fig. J5: Comparison between the corrupted and adapted point clouds following the application of CloudFixer to 15 different corruption types. The example pertains to a single original point cloud belonging to the table class in the test set of ModelNet40-C.

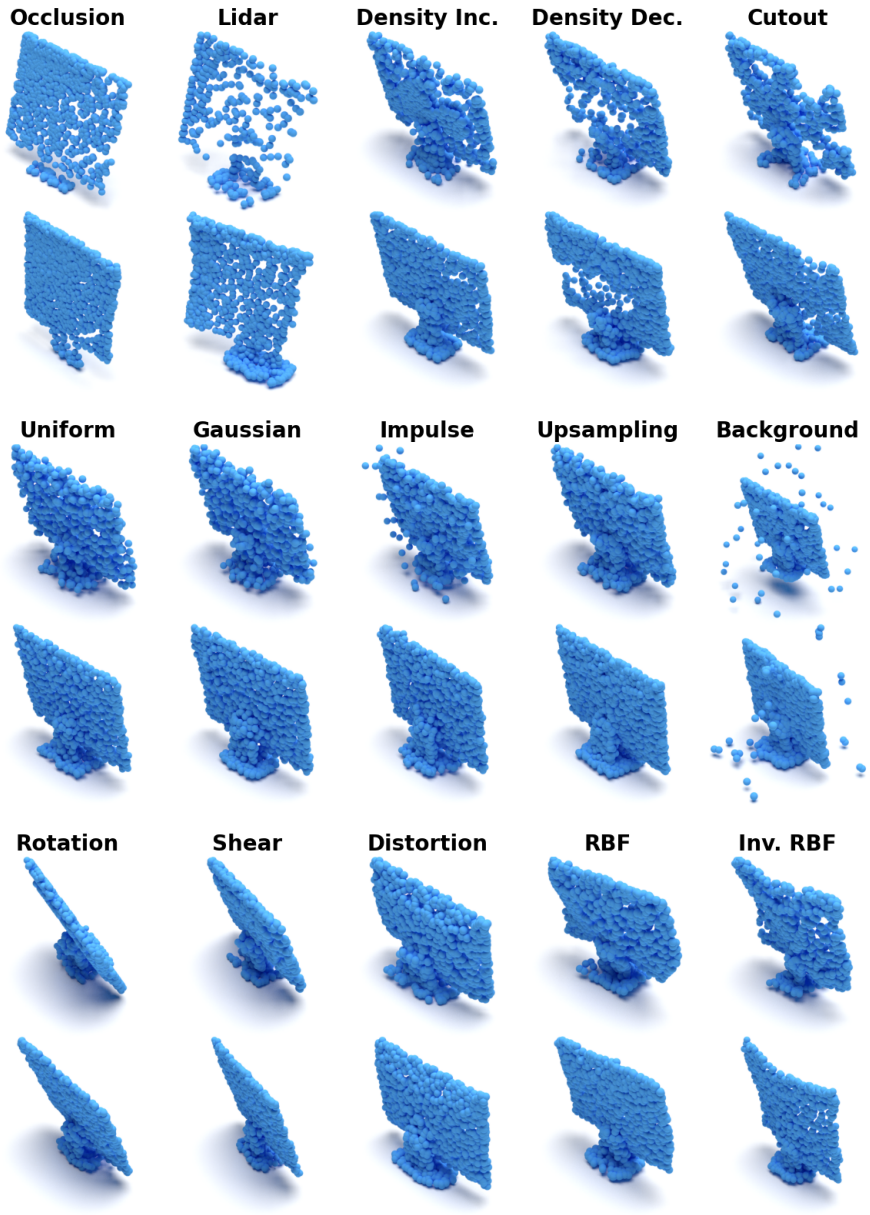

Fig. J6: Comparison between the corrupted and adapted point clouds following the application of CloudFixer to 15 different corruption types. The example pertains to a single original point cloud belonging to the monitor class in the test set of ModelNet40-C.

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. Hoogeboom, 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., Thickstun, 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 implicits. 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)