RAD: RECONSTRUCTION-ANCHORED DIFFUSION PRETRAINING FOR CORRUPTION-ROBUST POINT CLOUD RECOGNITION

Anonymous authors
Paper under double-blind review

ABSTRACT

Corruption-robust point-cloud recognition has become an essential task for the realm of 3D vision. Current globally conditioned diffusion pretraining typically optimizes the encoder for whole-shape denoising through a pooled global condition. Although effective in transferring features for clean recognition, this approach leaves masked-patch geometry without an explicit prediction objective. In this work, we propose a framework, named as RAD, for corruption-robust point-cloud recognition, via complementing whole-shape generative pretraining with an explicit geometric prediction task. Specifically, RAD consists of two key modules, the Local Geometry Reconstructor and the Conditional Flow Denoiser. The former module, the Local Geometry Reconstructor, is designed to reconstruct masked patches from visible encoder tokens under Chamfer distance, thus supplying the missing geometric prediction objective. The Conditional Flow Denoiser retains whole-shape denoising as an auxiliary pretraining objective, thereby keeping the generative branch. Through the integration of these two modules, RAD jointly pretrains a single encoder that is transferred alone, effectively leaving the downstream classifier and fine-tuning protocol unchanged. Extensive experiments across diverse corruption benchmarks demonstrate that adding geometric reconstruction reduces mean Corruption Error relative to the diffusion baseline.

1 INTRODUCTION

Three-dimensional point-cloud recognition is a cornerstone of many applications such as autonomous driving and robotics. Prevailing techniques have been designed and evaluated mainly on clean clouds, such as ModelNet40 (Wu et al., 2015) and ScanObjectNN (Uy et al., 2019), overlooking sensor noise, occlusion-induced dropout, non-uniform density, and rigid misalignment that deployed clouds routinely exhibit in corruption benchmarks (Ren et al., 2022b; Wang et al., 2023; Ren et al., 2022a). To obtain transferable 3D features under this setting, diffusion-based self-supervised pretraining has emerged as a powerful route, with globally conditioned diffusion (Zheng et al., 2024) becoming a mainstream choice for encoder pretraining. Therefore, improving the corruption robustness of such pretrained encoders remains a significant yet challenging task.

Current globally conditioned diffusion pretraining (Zheng et al., 2024) groups the cloud, embeds masked patches, and supports a generative denoiser through a pooled global condition, discarding the denoiser upon downstream transfer. Although effective for clean recognition, this approach leaves masked patches without an explicit geometric objective, because the encoder only optimizes whole-shape denoising (Fig. 1a–b). Thus, complementing whole-shape generative pretraining with explicit geometric prediction for 3D perception under corruption remains the challenge we address.

To address the aforementioned limitation, we present a reconstruction-anchored diffusion framework, named as RAD, for corruption-robust point-cloud recognition, via complementing whole-shape generative pretraining with an explicit geometric prediction task. Specifically, RAD consists of two key modules, the Local Geometry Reconstructor and the Conditional Flow Denoiser, on a shared Mask Encoder (Fig. 2). The former, the Local Geometry Reconstructor, is designed to supply the missing geometric prediction objective. It reconstructs masked patches from visible encoder tokens under Chamfer distance, so that the encoder is asked to recover held-out local coordinates

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

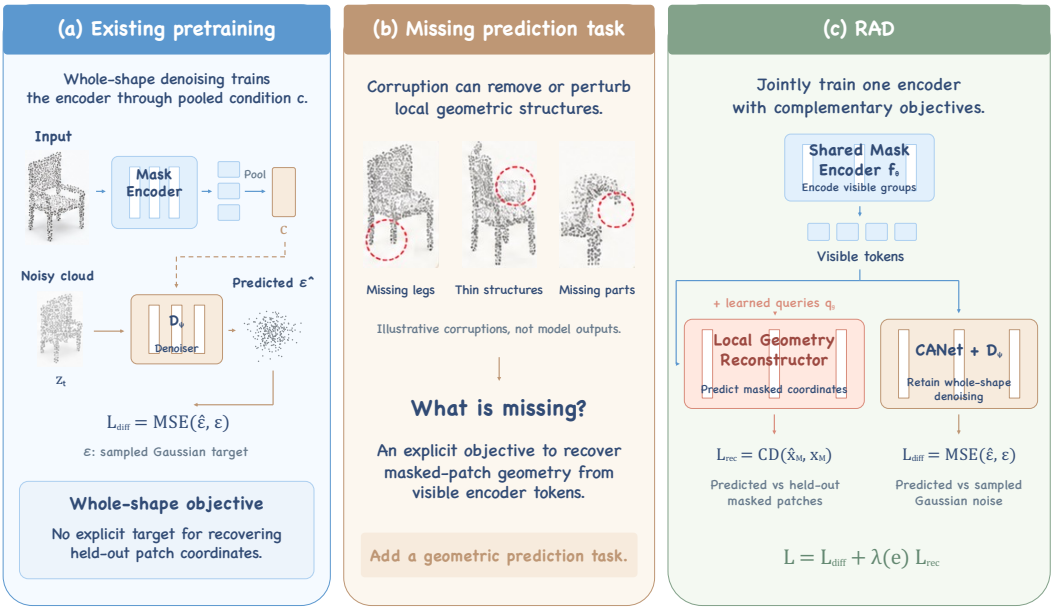


Figure 1: Motivation of RAD. (a) Existing diffusion pretraining relies on whole-shape denoising, lacking explicit target supervision for masked patch coordinates. (b) Corruptions damage local geometric structures, motivating explicit recovery of held-out geometry. Examples depict corruptions, not model outputs. (c) RAD incorporates a Local Geometry Reconstructor alongside whole-shape denoising, transferring only the shared encoder downstream.

rather than only to support whole-shape denoising. Meanwhile, the Conditional Flow Denoiser retains whole-shape denoising as an auxiliary pretraining objective. It pools the same encoder tokens into a global condition and predicts noise on a flow-matching interpolant, thereby keeping the generative branch through which the encoder is still trained. Through the integration of these two modules, RAD jointly pretrains a single encoder that is transferred alone, effectively leaving the downstream classifier and fine-tuning protocol unchanged.

We extensively validate the effectiveness of RAD on ModelNet-C and more challenging real-scan scenarios on ScanObjectNN-C. Our model outperforms previous models with multiple evaluation metrics on these benchmarks. Our contributions can be summarized as follows:

- We identify that globally conditioned diffusion pretraining optimizes the encoder for whole-shape denoising without an explicit masked-patch geometry objective, and we study this missing prediction task as a route to corruption-robust transfer.
- We propose RAD, which complements the inherited Conditional Flow Denoiser with a Local Geometry Reconstructor and transfers only the jointly trained encoder.
- We demonstrate the effectiveness of RAD across diverse corruption benchmarks, substantially reducing corruption errors without altering downstream protocols.

2 Related Work

Generative and diffusion pretraining for point clouds. Generative pretext objectives learn representations by conditioning a decoder or denoiser on encoder features. PointDif (Zheng et al., 2024) instantiates this idea for point clouds by using the encoder as the conditioner of a conditional denoising model, and it therefore serves as the diffusion-pretraining backbone of RAD. Although such globally conditioned pretraining transfers well for clean recognition, it optimizes the encoder for whole-shape denoising without an explicit masked-patch geometry objective. Related work explores several ways to combine point-cloud representations with diffusion. PreDifPoint (Qiu et al., 2025) aggregates backbone features to condition denoising, while Point-MaDi (Xiao et al., 2025) combines center-coordinate diffusion with conditional patch diffusion. Our focus is corruption-robust representation learning through joint pretraining: RAD combines whole-shape generative learning

with deterministic masked-patch coordinate reconstruction on encoder tokens before global pooling. Both pretraining heads are discarded during downstream transfer.

Masked point modeling. Masked point modeling provides a complementary route to point-cloud pretraining. Point-BERT (Yu et al., 2022) tokenizes local patches through a learned codebook, while Point-MAE (Pang et al., 2022) reconstructs masked-patch coordinates under Chamfer distance; subsequent methods then scale or restructure this recipe, including Point-M2AE (Zhang et al., 2022), PointGPT (Chen et al., 2023), and PCP-MAE (Zhang et al., 2024). The Local Geometry Reconstructor deliberately borrows this reconstruction form, yet it is not introduced as a stand-alone masked autoencoder. Instead, it acts as a local-geometry anchor inside diffusion pretraining, so that the encoder is asked to recover held-out patches while still supporting whole-shape denoising. Complete-target reconstructors take a different route: OcCo (Wang et al., 2021) learns transferable representations through completion of view-occluded point clouds, and Point-DAE (Zhang et al., 2025) reconstructs a complete cloud from a corrupted input. These methods train an encoder–decoder model to reconstruct a complete point cloud, whereas LGR supervises masked patches of the same grouped cloud while the shared encoder also supports whole-shape generative learning.

Corruption robustness for point clouds. ModelNet-C (Ren et al., 2022b) and ScanObjectNN-C (Wang et al., 2023) provide benchmarks for point-cloud corruption robustness. We adopt the mean corruption error protocol of ModelNet-C, which normalizes errors by those of DGCNN (Wang et al., 2019). Dedicated robustness methods intervene in the recognizer or its training: RPC (Ren et al., 2022b) redesigns the recognition pipeline, AdaptPoint (Wang et al., 2023) learns sample-adaptive adversarial augmentation, and APCT (Wang et al., 2024) introduces adversarial-significance mechanisms. We additionally include PointGST (Liang et al., 2025), which applies spectral adaptation to a frozen backbone, as a parameter-efficient fine-tuning baseline. These approaches modify the recognition pipeline or downstream training. Our reconstruction-anchored framework, RAD, instead changes only the pretraining objective and transfers the same downstream encoder.

3 Methodology

3.1 Overview

Problem formulation. Given an unlabelled point-cloud corpus, our goal is to pretrain a Mask Encoder f_θ whose features transfer to corruption-robust recognition, while leaving the downstream classifier and its training protocol unchanged. Current globally conditioned diffusion pretraining (Zheng et al., 2024) groups the cloud, embeds masked patches, and supports a generative denoiser through a pooled global condition. Although effective for clean recognition, this paradigm optimizes the encoder only to support whole-shape denoising and therefore leaves masked-patch geometry without an explicit prediction objective. Because only f_θ is loaded at fine-tune, this limitation must be resolved in the pretraining objective rather than by redesigning the recognizer.

Framework overview. The key insight is that this missing geometric prediction task can complement whole-shape generative pretraining, rather than replace it, so that the transferred encoder encodes both held-out local coordinates and a whole-shape generative condition. As illustrated in Fig. 2, RAD consists of two key modules on a shared Mask Encoder. The **Local Geometry Reconstructor** is designed to recover held-out patch coordinates from visible tokens, thereby supplying the missing geometric prediction objective. Meanwhile, the inherited **Conditional Flow Denoiser** retains whole-shape denoising as an auxiliary pretraining objective on the same tokens. Through the integration of these two modules, a single encoder is jointly trained to encode both held-out local geometry and a whole-shape generative condition, and is transferred alone, leaving the downstream classifier and fine-tuning protocol unchanged.

3.2 Shared Mask Encoder

Both modules operate on a shared encoder representation, as only this backbone transfers downstream. This design ensures that reconstruction gradients directly shape the representation subsequently transferred to the classifier. We therefore share a Mask Encoder f_θ inherited from PointDif, so that both geometric prediction and whole-shape denoising optimize identical tokens.

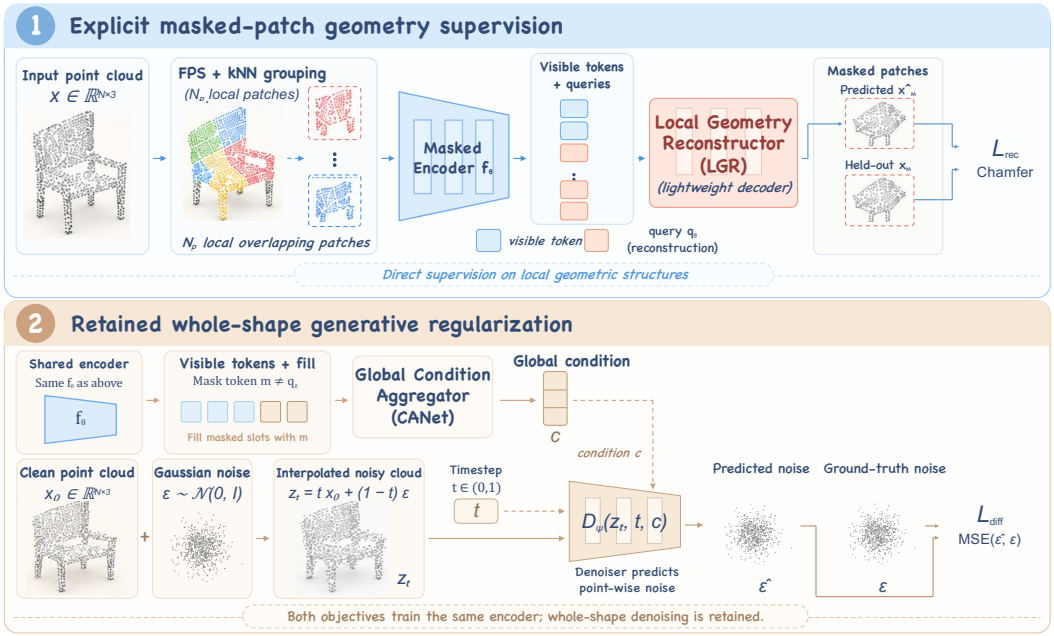


Figure 2: The overall architecture of RAD. An input point cloud is patchified and masked, where a shared Mask Encoder embeds visible patches into tokens. The *Local Geometry Reconstructor* combines visible tokens with learned queries to predict held-out patch coordinates under reconstruction loss $\mathcal{L}_{\text{rec}}$. Meanwhile, the *Conditional Flow Denoiser* pools tokens into a global condition for whole-shape denoising under diffusion loss $\mathcal{L}_{\text{diff}}$. Both objectives jointly optimize the shared encoder, which is transferred alone downstream while auxiliary heads are discarded.

Patchify and mask. Given an input point cloud $x \in \mathbb{R}^{N\times 3}$, we first partition the raw coordinates into localized geometric primitives to decouple visible context from held-out prediction targets. Specifically, Farthest Point Sampling (FPS) selects G representative center points, around each of which an S-point local neighborhood is gathered via k-Nearest Neighbors (kNN), yielding a set of local patches $\{x_g\}_{g=1}^{G}$ with $x_g \in \mathbb{R}^{S\times 3}$. To construct the pretext supervision, a random mask $\mathcal{M} \subset \{1,\dots,G\}$ designates a fraction of groups as held-out targets, while the complementary index set $\mathcal{V}$ retains the visible context. This spatial partition fulfills two synchronized purposes: the visible patches $\mathcal{V}$ supply contextual evidence for representation learning, whereas the held-out patches $\mathcal{M}$ establish an explicit coordinate ground truth for subsequent geometric reconstruction. These raw grouped coordinates, however, cannot directly interact through Transformer attention or condition the diffusion denoiser without feature transformation.

Visible-token encoding. To address this, we map only the visible groups into latent feature space, preventing trivial coordinate leakage while establishing high-dimensional representations for subsequent modules. Specifically, each visible patch x_g ($g \in \mathcal{V}$) is first embedded by a mini-PointNet projection and combined with the positional encoding of its cluster center, after which standard Transformer encoder blocks f_θ extract contextual tokens $\{t_g\}_{g\in\mathcal{V}}$. To ensure that the robustness improvements are attributable to our pretraining objective, we directly adopt the Transformer encoder architecture from PointDif (Zheng et al., 2024) and keep the grouping and tokenization parameters unchanged. The resulting visible tokens $\{t_g\}_{g\in\mathcal{V}}$ serve as the unified substrate shared across both downstream heads; while they readily supply contextual evidence for whole-shape denoising, they are not yet incentivized to recover the missing local geometry of the masked slots.

3.3 Local Geometry Reconstructor

While the visible tokens produced by the encoder provide contextual cues for downstream denoising, they are never constrained to recover the fine-grained metric coordinates of the unobserved geometry. To address this deficiency without altering the downstream classifier architecture or discarding generative pretraining, we introduce an explicit geometric prediction task on the exact same token sequence. Specifically, the Local Geometry Reconstructor is activated exclusively during pre-

training: it deploys learnable queries at masked locations, decodes fine-grained geometric details conditioned on visible tokens, and aligns the decoded point sets directly with the ground-truth held-out patches. We unfold this reconstructive process through three progressive steps: masked-slot querying, visible-context decoding, and Chamfer matching.

Masked-slot querying. To formulate a reconstruction objective for the held-out geometry, each masked patch index $g \in \mathcal{M}$ must serve as an active spatial probe rather than an uninformative pooling slot. We therefore introduce a shared learnable reconstructor query $q \in \mathbb{R}^C$ and combine it with the positional embedding of each group center $p_g \in \mathbb{R}^3$, generating patch-specific queries $q_g = q + \mathrm{PE}_{\mathrm{dec}}(p_g)$ for every $g \in \mathcal{M}$, using a decoder positional MLP separate from the encoder. Critically, these reconstructor queries $\{q_g\}$ are strictly decoupled from the diffusion mask token m utilized in the denoiser branch, which merely indicates masked positions for global pooling. Although these queries explicitly encode the spatial whereabouts of the missing patches, they remain devoid of surface geometry without attending to the visible context.

Visible-context decoding. To incorporate visible context, we concatenate the visible encoder tokens with the shared learnable mask tokens and add the corresponding center positional embeddings. A lightweight Transformer block applies self-attention to this full sequence, allowing masked slots to exchange information with visible tokens and other masked slots. After layer normalization, the masked-slot outputs pass through a linear coordinate prediction head to produce reconstructed patches $\hat{x}_g \in \mathbb{R}^{S\times 3}$ in center-relative coordinates. An explicit point-set reconstruction objective then supervises these predictions against the corresponding held-out local patches.

Chamfer matching. We supervise predictions exclusively on the held-out masked groups $\mathcal{M}$, compelling the encoder to infer missing local structures from surrounding context rather than reconstructing visible surfaces. Denoting predicted and ground-truth coordinate sets by $\hat{x}_{\mathcal{M}} = \{\hat{x}_g\}_{g\in\mathcal{M}}$ and $x_{\mathcal{M}} = \{x_g\}_{g\in\mathcal{M}}$, the geometric reconstruction loss is formulated under Chamfer Distance:

$$\mathcal{L}_{\mathrm{rec}} \;=\; \frac{1}{|\mathcal{M}|}\sum_{g\in\mathcal{M}} \mathrm{CD}\big(\hat{x}_g,\, x_g - p_g\big), \tag{1}$$

where for two finite point subsets $P, Q \subset \mathbb{R}^3$, the symmetric Chamfer Distance is defined as:

$$\mathrm{CD}(P,Q) = \frac{1}{|P|}\sum_{p\in P}\min_{q\in Q}\|p-q\|_2 + \frac{1}{|Q|}\sum_{q\in Q}\min_{p\in P}\|q-p\|_2. \tag{2}$$

This coordinate-level formulation propagates backpropagation gradients directly through the decoder attention back into the visible encoder tokens $\{t_g\}_{g\in\mathcal{V}}$, forcing f_θ to encode local surface continuity and metric fidelity. We retain the whole-shape generative objective alongside local coordinate reconstruction, so that both pretraining tasks supervise the shared encoder.

3.4 Conditional Flow Denoiser

To preserve the generative representation benefits that safeguard against global corruptions, RAD retains whole-shape denoising as a complementary pretraining objective rather than replacing it with reconstruction alone. Concretely, this module aggregates the token representations into a holistic shape condition and trains the backbone by predicting noise ε on a continuous flow-matching probability trajectory. This generative branch unfolds through three synchronized steps: condition pooling, flow-matching trajectory interpolation, and conditional noise prediction.

Condition pooling. While the generative branch requires an expressive condition describing global topology, it must remain functionally separate from localized reconstructor queries q_g. To achieve this decoupling, masked slots are filled with a learned mask token $m \in \mathbb{R}^C$ where $m \neq q_g$, yielding a full-length sequence of size $G \times C$. A Condition Aggregation Network then pools this sequence through two stages of pointwise convolutions and soft pooling into a global condition vector $c \in \mathbb{R}^{768}$. This condition c serves as a macro-level shape descriptor for generative denoising, complementing the patch-level Chamfer objective without interfering with local coordinates. To optimize the encoder via this condition, we establish a whole-shape generative target path.

Flow-matching interpolant. To establish a holistic generative objective, we define a continuous diffusion interpolant over the complete point cloud, providing a global denoising target that complements local patch reconstruction. Given a clean sample $x_0 \in \mathbb{R}^{N\times 3}$ and standard Gaussian noise

$\varepsilon \sim \mathcal{N}(0, I)$, we construct the intermediate state z_t via linear flow interpolation:

$$z_t \;=\; t\, x_0 + (1-t)\, \varepsilon, \tag{3}$$

where the continuous timestep $t \in [0, 1]$ is sampled from a logit-normal distribution concentrated around intermediate noise levels where generative gradients are most discriminative. With the interpolant z_t and condition c formulated, the network is equipped to execute conditional denoising.

Conditional ε-prediction. We use a pointwise MLP denoiser D_ψ based on PointDif (Zheng et al., 2024), with scale-and-shift conditioning and no layer normalization in its modulation layers. Conditioned on the global vector c and the time embedding of t, D_ψ processes the noisy point cloud z_t to predict the injected perturbation ε, optimized via mean-squared error:

$$\mathcal{L}_{\text{diff}} \;=\; \big\| D_\psi(z_t,\, c,\, t) - \varepsilon \big\|_2^2. \tag{4}$$

This generative loss trains the Mask Encoder f_θ to preserve global shape semantics and manifold structure, forming a dual-objective training synergy with the geometric prediction task.

3.5 Joint objective and transfer

The Local Geometry Reconstructor and Conditional Flow Denoiser establish complementary supervision on shared encoder tokens: metric Euclidean geometry at the patch level, and global shape consistency across the cloud. Below, we formulate their joint objective and transfer protocol.

Pretraining objective. To integrate geometric reconstruction into diffusion pretraining without destabilizing representations, we optimize the shared encoder under a unified objective:

$$\mathcal{L} \;=\; \mathcal{L}_{\text{diff}} + \lambda(e)\, \mathcal{L}_{\text{rec}}, \tag{5}$$

where e denotes the zero-based training epoch, and $\lambda(e) = \lambda_0 \cdot \min\big(1,\, e/E_{\text{w}}\big)$ implements a linear warmup over E_{w} epochs ($\lambda_0 = 1.0, E_{\text{w}} = 30$). Importantly, whole-shape denoising $\mathcal{L}_{\text{diff}}$ remains active throughout pretraining from $e = 0$; this weighting schedule acts as a stabilization mechanism rather than a disjoint curriculum, ensuring the encoder is never exposed to reconstruction in isolation.

Encoder-only transfer. Because both auxiliary heads serve purely to enrich pretraining representations, they are completely discarded after pretraining. During downstream fine-tuning and inference, the classifier instantiates only the pretrained Mask Encoder f_θ as its backbone. This strict protocol ensures that robustness gains cannot be confounded by auxiliary parameters, inference overhead, or classifier modifications. Downstream classification follows the standard fine-tuning protocol.

4 Experiments

We first detail the experimental setup in §4.1. Then, we evaluate the corruption robustness of RAD on the synthetic ModelNet-C benchmark in §4.2 and on the real-world ScanObjectNN-C benchmark in §4.3. Finally, we conduct comprehensive ablative analyses in §4.4 to examine the reconstruction decoder, loss weighting, and the roles of the two pretraining branches.

4.1 Experimental Setup

Datasets and corruption benchmarks. We evaluate point-cloud recognition under corruptions that simulate realistic sensor degradations, structural occlusions, and spatial perturbations:

- **ShapeNet** (Chang et al., 2015) serves as the dataset for self-supervised pretraining. It consists of 51,127 clean synthetic 3D CAD models spanning 55 common object categories. Following standard pretraining protocols (Pang et al., 2022; Zheng et al., 2024), each object is represented by 1,024 points uniformly sampled from its mesh surface without normal or color attributes.
- **ModelNet-C** (Ren et al., 2022b) evaluates corruption robustness on synthetic CAD objects across 40 categories. It subjects the test set to seven corruption families: Scale, Jitter, Drop-Global (Drop-G), Drop-Local (Drop-L), Add-Global (Add-G), Add-Local (Add-L), and Rotate. Each corruption spans five severity levels, yielding 35 standardized corruption scenarios.

Table 1: Classification results on the ModelNet-C dataset, mCE ($\%, \downarrow$) is reported, the best performance is **bold**. † denotes method designed specifically against corruption.

Method	Refer	mCE	Seven Corruption Families (CE ↓)						
			Sca.	Jit.	Drop-G	Drop-L	Add-G	Add-L	Rot.
DGCNN (Wang et al., 2019)	[TOG'19]	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
PointNet (Qi et al., 2017a)	[CVPR'17]	142.2	126.6	**64.2**	50.0	107.2	298.0	159.3	190.2
PointNet++ (Qi et al., 2017b)	[NeurIPS'17]	107.2	87.2	117.7	64.1	180.2	61.4	99.3	140.5
GDANet (Xu et al., 2021)	[AAAI'21]	89.2	**83.0**	83.9	79.4	89.4	87.1	103.6	**98.1**
PCT (Guo et al., 2021)	[CVM'21]	92.5	87.2	87.0	52.8	100.0	78.0	138.5	104.2
RPC† (Ren et al., 2022b)	[ICML'22]	86.3	84.0	89.2	49.2	79.7	92.9	101.1	107.9
PointNeXt (Qian et al., 2022)	[NeurIPS'22]	85.6	90.4	129.7	84.7	95.7	25.1	27.6	146.0
Point-M2AE (Zhang et al., 2022)	[NeurIPS'22]	83.9	95.7	159.2	53.2	83.6	30.2	30.5	134.9
PointGPT (Chen et al., 2023)	[NeurIPS'23]	83.4	108.5	102.5	58.5	98.1	29.2	32.7	154.4
PointMamba (Liang et al., 2024)	[NeurIPS'24]	87.2	102.1	121.8	56.5	103.9	30.5	31.6	163.7
PCP-MAE (Zhang et al., 2024)	[NeurIPS'24]	91.2	123.4	112.7	68.1	119.8	33.2	36.0	145.1
MAP (Liu & Yi, 2025)	[CVPR'25]	99.8	192.6	105.7	94.0	111.1	32.9	36.4	126.0
PointGST (GPT-S) (Liang et al., 2025)	[TPAMI'25]	84.0	91.5	167.4	57.7	**69.1**	24.1	**25.8**	152.6
PointGST (MAE) (Liang et al., 2025)	[TPAMI'25]	75.6	84.0	96.8	56.9	94.7	**23.7**	26.2	147.0
APCT (Wang et al., 2024)	[NeurIPS'24]	72.2	94.7	88.3	46.8	85.0	28.5	29.8	132.6
PointDif (Zheng et al., 2024)	[CVPR'24]	73.8	92.6	108.2	41.5	76.8	28.8	30.5	138.1
RAD (ours)		**65.2**	87.2	87.0	**41.1**	72.5	26.8	28.7	113.0
vs. prev. SoTA	–	***↓7.0***				-			

- **ScanObjectNN-C** (Wang et al., 2023) evaluates robustness on real-world object point clouds derived from RGB-D scans, using the PB-T50-RS split (Uy et al., 2019) (15 categories). It applies the same seven synthetic corruption families across five severity levels to scanned objects with incomplete geometry and background clutter.

Implementation details. We describe the pretraining and downstream fine-tuning settings below:

- **Pretraining protocols.** We pretrain RAD on ShapeNet using AdamW with a total batch size of 256, a base learning rate of 2×10^{-3}, and weight decay of 0.05. We use a 300-epoch cosine learning-rate schedule with a 10-epoch linear warmup. Each input contains 1,024 points.
- **Architecture and supervision.** FPS and kNN form $G = 64$ patches of $S = 32$ points. The random masking ratio is set to 60%, yielding $|\mathcal{M}| = \lfloor 0.6G \rfloor = 38$. The encoder has 12 Transformer blocks with dimension $C = 384$ and six attention heads. LGR applies one self-attention block to the concatenated visible tokens and learnable mask tokens, followed by layer normalization and a linear coordinate prediction head. Its loss weight follows $\lambda(e) = \min(1, e/30)$, while the denoising objective remains active throughout pretraining.
- **Downstream transfer.** We transfer the encoder weights to the classification architecture used in PointDif (Zheng et al., 2024), discarding both pretraining heads and CANet. We fine-tune on the original ModelNet40 and ScanObjectNN PB-T50-RS training sets using AdamW with batch size 32, base learning rate 5×10^{-4}, weight decay 0.05, and the same learning-rate schedule. Both datasets use 1,024 input points and 32 points per patch, with 64 patches on ModelNet40 and 128 on ScanObjectNN. We apply random rotation about the y-axis when fine-tuning on ScanObjectNN. No RAD-specific modules are added to the downstream classifier.

Evaluation metric. Following the standardized ModelNet-C protocol (Ren et al., 2022b), we employ the Mean Corruption Error (mCE, $\%, \downarrow$) as our primary metric to assess corruption robustness.

4.2 Evaluation on Synthetic CAD Corruptions: ModelNet-C

As summarized in Tab. 1, we report the quantitative classification performance across all seven corruption families on the synthetic ModelNet-C benchmark. RAD establishes advanced and highly competitive performance, achieving an overall Mean Corruption Error of **65.2%**. This performance decisively outperforms representative supervised backbones (PointNet++ at 107.2% and PointNeXt at 85.6%), hierarchical masked autoencoders (Point-M2AE at 83.9%), autoregressive and state-space architectures (PointGPT at 83.4% and PointMamba at 87.2%), and pure generative diffusion pretraining (PointDif at 73.8%). When compared against specialized robustness architectures, RAD demonstrates a marked enhancement of 7.0 mCE-point reduction over the leading adversarial token-dropping model APCT (Wang et al., 2024) (72.2%), and outpaces the parameter-efficient fine-tuning baseline PointGST-MAE (Liang et al., 2025) (75.6%) by 10.4 mCE points. Crucially,

Table 2: Classification results on the ScanObjectNN-C dataset, mCE (%, ↓) is reported, the best performance is **bold**. † denotes method designed specifically against corruption.

Method	Refer	mCE	Seven Corruption Families (CE ↓)						
			Sca.	Jit.	Drop-G	Drop-L	Add-G	Add-L	Rot.
DGCNN (Wang et al., 2019)	[TOG'19]	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
PointNet (Qi et al., 2017a)	[CVPR'17]	132.8	141.2	**88.4**	78.3	125.3	139.2	200.0	156.9
PointNet++ (Qi et al., 2017b)	[NeurIPS'17]	96.9	89.7	110.3	55.0	127.7	94.7	90.5	110.7
RPC† (Ren et al., 2022b)	[ICML'22]	132.6	131.7	107.3	145.5	130.5	114.2	158.7	140.2
PointNeXt (Qian et al., 2022)	[NeurIPS'22]	92.1	80.3	107.9	80.7	94.2	94.4	87.5	99.5
Point-M2AE (Zhang et al., 2022)	[NeurIPS'22]	78.2	**68.5**	119.3	55.3	83.2	35.7	71.4	114.2
PointGPT (Chen et al., 2023)	[NeurIPS'23]	84.7	119.9	104.0	58.5	76.9	36.1	72.2	125.1
PointMamba (Liang et al., 2024)	[NeurIPS'24]	80.2	104.0	112.5	52.6	71.9	33.7	67.4	119.1
PCP-MAE (Zhang et al., 2024)	[NeurIPS'24]	88.4	117.3	103.3	65.9	89.1	38.9	78.0	126.2
MAP (Liu & Yi, 2025)	[CVPR'25]	75.3	100.9	104.5	51.6	70.6	32.0	63.4	104.3
PointGST (GPT-S) (Liang et al., 2025)	[TPAMI'25]	74.7	83.2	118.8	52.4	66.0	29.1	58.1	115.4
PointGST (MAE) (Liang et al., 2025)	[TPAMI'25]	69.1	87.9	93.9	50.3	66.3	28.0	57.7	99.6
APCT (Wang et al., 2024)	[NeurIPS'24]	74.2	104.5	98.9	48.9	67.0	30.0	63.0	107.1
PointDif (Zheng et al., 2024)	[CVPR'24]	71.6	96.7	103.7	47.1	65.0	29.1	57.7	101.9
RAD (ours)		**64.1**	91.2	89.5	**41.3**	**56.4**	**27.2**	**53.3**	**89.5**
vs. prev. SoTA	–	***↓5.0***				-			

while competing robustness frameworks intervene directly on downstream classifier architectures or enforce adversarial perturbation loops during supervised training, RAD secures its resilience entirely through self-supervised pretraining objectives. Upon downstream deployment, the pretrained encoder transfers seamlessly into a standard classification head with zero architectural overhead, completely circumventing the inference latency penalties associated with specialized classifiers.

The per-corruption results in Tab. 1 show that RAD achieves lower CE than the original PointDif baseline across all seven corruption families. For example, Drop-Local CE decreases from 76.8% to 72.5%, Jitter from 108.2% to 87.0%, and Rotate from 138.1% to 113.0%. These results demonstrate improved robustness across both missing-point and coordinate perturbations.

4.3 Evaluation on Real-World Object Scans: ScanObjectNN-C

Unlike synthetic CAD models, object point clouds extracted from real-world RGB-D scans contain incomplete surfaces, occlusions, and background clutter. As reported in Tab. 2, we evaluate robustness to synthetic corruptions applied to these scanned objects on ScanObjectNN-C. On this realistic testbed, RAD achieves an advanced score of 64.1% mCE, establishing clear margins over leading competitors across different design paradigms, such as PointGST-MAE at 69.1%, generative conditional diffusion PointDif at 71.6%, and adversarial token-dropping APCT at 74.2%. Compared to classical supervised backbones like PointNet at 132.8% and DGCNN at 100.0%, RAD achieves dramatic reductions of 68.7 and 35.9 mCE points, demonstrating improved classification robustness on corrupted real-world object scans. Furthermore, RAD achieves lower CE than the original PointDif baseline across all seven corruption families, with Jitter CE dropping by 14.2 points and Rotate by 12.4 points, demonstrating that the learned geometric representations transfer effectively to corrupted real-world object scans.

4.4 Ablation Study

To systematically dissect the algorithmic components of RAD and substantiate the underlying physical mechanisms driving corruption robustness, we conduct extensive ablative experiments across both synthetic ModelNet-C and real-world ScanObjectNN-C benchmarks.

Effect of core pretraining modules. To evaluate the core pretraining objectives in RAD, we ablate the generative diffusion and reconstruction branches in Tab. 3. Relying solely on generative diffusion optimizes the encoder for whole-shape denoising through a globally pooled condition; lacking token-level geometric supervision on local patches, this branch yields higher corruption errors of 73.8% on ModelNet-C and 71.6% on ScanObjectNN-C. Conversely, training solely with the Local Geometry Reconstructor increases error to 69.1% on ModelNet-C, demonstrating that the continuous velocity field learned by whole-shape diffusion remains indispensable for regularizing coordinate noise. Unifying both objectives in RAD achieves the best overall robustness across both domains, reducing mCE to 65.2% on ModelNet-C and 64.1% on ScanObjectNN-C. These results

Table 3: Effect of core pretraining modules.

Diffusion	LGR	MN40-C	Scan-C
✓	–	73.8	71.6
–	✓	69.1	64.4
✓	✓	**65.2**	**64.1**

Table 4: Effect of reconstruction decoder depth.

Depth	MN40-C	Scan-C
0	71.2	73.2
1	**65.2**	**64.1**
2	67.7	66.2

Table 5: Effect of reconstruction loss weight ratio λ.

λ	MN40-C	Scan-C
0.5	66.7	64.5
1.0	**65.2**	**64.1**

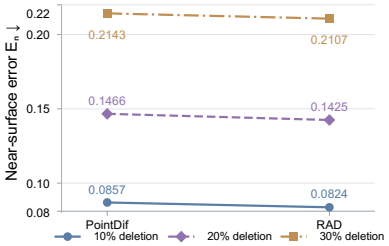


Figure 3: Near-surface readout error $E_N \downarrow$ from frozen PointDif and RAD encoders under local deletion.

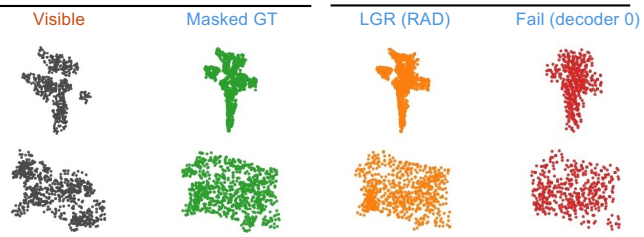


Figure 4: LGR visualisation on held-out ShapeNet groups (same FPS+kNN and mask in every column): visible points, masked ground truth, RAD prediction, decoder-depth-0 fail.

demonstrate that generative diffusion and local coordinate reconstruction provide complementary inductive biases across synthetic and scanned geometries, while both pretraining heads are discarded upon downstream transfer without altering the classifier architecture or fine-tuning protocol.

Geometry readout under local deletion. To assess local geometric decodability, we freeze the pretrained encoders and independently train readers of identical architecture to predict truncated nearest-neighbor distances to the clean point set. As shown in Fig. 3, RAD consistently achieves lower near-surface error than PointDif across 10%, 20%, and 30% local deletion, reducing error from 0.0857 to 0.0824, 0.1466 to 0.1425, and 0.2143 to 0.2107, respectively. These results indicate that local geometric information is more readily predicted from RAD's pretrained representation under the shared reader protocol at these deletion levels.

Effect of reconstruction decoder depth. We investigate the configuration of the reconstruction decoder by evaluating different self-attention depths in Tab. 4. Setting the decoder depth to zero, which maps learnable mask queries to coordinates through a linear layer without cross-token attention, causes performance to collapse to 71.2% on ModelNet-C and 73.2% on ScanObjectNN-C. This degradation of 6.0 points on ModelNet-C and 9.1 points on ScanObjectNN-C relative to the single-block default indicates that spatial attention between visible tokens and masked queries is indispensable for inferring continuous geometry. As illustrated in Fig. 4, the single-block decoder faithfully reconstructs fine local structures on held-out ShapeNet patches, whereas the zero-depth variant collapses into uninformative point clusters. Conversely, stacking two self-attention blocks yields 67.7% on ModelNet-C and 66.2% on ScanObjectNN-C, showing that deeper decoders provide diminishing returns while adding pretraining overhead.

Sensitivity to reconstruction loss weighting. We next examine the impact of the reconstruction loss weight on representation quality in Tab. 5. Halving the reconstruction weight from the default 1.0 to 0.5 increases corruption errors to 66.7% on ModelNet-C and 64.5% on ScanObjectNN-C, confirming that a balanced weighting of 1.0 is suitable. This indicates that the default loss magnitude provides sufficient gradient signals to anchor local geometry without overpowering the generative diffusion objective. Because the reconstruction module is discarded after pretraining, this pretraining loss schedule enriches encoder representations without incurring downstream computational cost.

5 Conclusion

In this work, we introduce a new pretraining framework, named as RAD, tailored for corruption-robust 3D point cloud recognition. Our framework integrates a Local Geometry Reconstructor and a Conditional Flow Denoiser directly within the pretraining stage. The former reconstructs masked patch coordinates under Chamfer distance, while the latter retains whole-shape generative denoising. Because all auxiliary heads are discarded upon downstream transfer, RAD enriches geometric representations without altering classifier architectures or adding inference compute. Experimental evaluations on comprehensive benchmarks manifest its superiority.

AI USE STATEMENT

In this work, we have not used generative AI tools to implement methods, to generate synthetic data sets, to develop theoretical models or conceptual frameworks, to propose or refine hypotheses, to design or provide feedback on research methodology or experiments, to clean or reformat data sets, to support qualitative or thematic data analysis, or to interpret results; formulating mathematical claims, providing ingredients for or assisting in the writing of proofs, and assisting with translation are not applicable to this work. Additionally, we used generative AI tools to create or edit software code, to edit the paper to improve readability, and to create or modify scientific figures. We have reviewed all AI-assisted work: AI-generated code was read, verified and tested by the authors before use, every passage of AI-edited text was checked by the authors against the underlying experiments, and AI-assisted figures were checked against the data they display. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies robustness of point-cloud recognition models on public academic benchmarks (ShapeNet, ModelNet40, ScanObjectNN, and their corruption variants). It does not involve human subjects, private data, or dual-use applications beyond standard 3D perception research.

REPRODUCIBILITY STATEMENT

Pretraining and fine-tuning protocols, architectures, and hyperparameters are specified in Sec. 3 and Sec. 4. We will release code and pretrained checkpoints upon acceptance.

REFERENCES

Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An information-rich 3D model repository. Technical Report arXiv:1512.03012, Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 6

Guangyan Chen, Meiling Wang, Yi Yang, Kai Yu, Li Yuan, and Yufeng Yue. PointGPT: Auto-regressively generative pre-training from point clouds. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023. 3, 7, 8, 16

Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R. Martin, and Shi-Min Hu. PCT: Point cloud transformer. *Computational Visual Media*, 7(2):187–199, 2021. doi: 10.1007/s41095-021-0229-5. 7, 16

Dingkang Liang, Xin Zhou, Wei Xu, Xingkui Zhu, Zhikang Zou, Xiaoqing Ye, Xiao Tan, and Xiang Bai. PointMamba: A simple state space model for point cloud analysis. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024. 7, 8, 16

Dingkang Liang, Tianrui Feng, Xin Zhou, Yumeng Zhang, Zhikang Zou, and Xiang Bai. Parameter-efficient fine-tuning in spectral domain for point cloud learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 47(12):10949–10966, 2025. doi: 10.1109/TPAMI.2025.3594749. 3, 7, 8, 16, 17

Yunze Liu and Li Yi. MAP: Unleashing hybrid mamba-transformer vision backbone's potential with masked autoregressive pretraining. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9676–9685, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Liu_MAP_Unleashing_Hybrid_Mamba-Transformer_Vision_Backbones_Potential_with_Masked_Autoregressive_CVPR_2025_paper.html. 7, 8, 16

Yatian Pang, Wenxiao Wang, Francis E. H. Tay, Wei Liu, Yonghong Tian, and Li Yuan. Masked autoencoders for point cloud self-supervised learning. In *European Conference on Computer Vision (ECCV)*, pp. 604–621. Springer, 2022. doi: 10.1007/978-3-031-20086-1_35. 3, 6, 16

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep learning on point sets for 3d classification and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 652–660, 2017a. 7, 8, 16

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pp. 5099–5108, 2017b. URL https://papers.nips.cc/paper/7095-pointnet-deep-hierarchical-feature-learning. 7, 8, 16

Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Abed Al Kader Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. PointNeXt: Revisiting PointNet++ with improved training and scaling strategies. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, 2022. 7, 8, 16

Chang Qiu, Feipeng Da, and Zilei Zhang. Feature extraction and representation of pre-training point cloud based on diffusion models. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 26559–26568, 2025. doi: 10.1109/ICCV51701.2025.02465. 2

Jiawei Ren, Lingdong Kong, Liang Pan, and Ziwei Liu. PointCloud-C: Benchmarking and analyzing point cloud perception robustness under corruptions. Project page, 2022a. URL https://pointcloud-c.github.io. 1, 13, 14

Jiawei Ren, Liang Pan, and Ziwei Liu. Benchmarking and analyzing point cloud classification under corruptions. In *International Conference on Machine Learning (ICML)*, volume 162 of *PMLR*, pp. 18559–18575, 2022b. URL https://proceedings.mlr.press/v162/ren22c.html. 1, 3, 6, 7, 8, 13, 14, 16

Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1588–1597, 2019. doi: 10.1109/ICCV.2019.00167. 1, 7

Hanchen Wang, Qi Liu, Xiangyu Yue, Joan Lasenby, and Matt J. Kusner. Unsupervised point cloud pre-training via occlusion completion. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9782–9792, 2021. URL https://openaccess.thecvf.com/content/ICCV2021/html/Wang_Unsupervised_Point_Cloud_Pre-Training_via_Occlusion_Completion_ICCV_2021_paper.html. 3

Jie Wang, Lihe Ding, Tingfa Xu, Shaocong Dong, Xinli Xu, Long Bai, and Jianan Li. Sample-adaptive augmentation for point cloud recognition against real-world corruptions. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 14330–14339, 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Wang_Sample-adaptive_Augmentation_for_Point_Cloud_Recognition_Against_Real-world_Corruptions_ICCV_2023_paper.html. 1, 3, 7, 13, 14, 16

Jie Wang, Tingfa Xu, Lihe Ding, and Jianan Li. Target-guided adversarial point cloud transformer towards recognition against real-world corruptions. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/b57b9aafccf6bf6a76d01079e316e14d-Abstract-Conference.html. 3, 7, 8, 14, 16

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 38(5):146:1–146:12, 2019. doi: 10.1145/3326362. 3, 7, 8, 14, 16

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1912–1920, 2015. doi: 10.1109/CVPR.2015.7298801. 1

Xiaoyang Xiao, Runzhao Yao, Zhiqiang Tian, and Shaoyi Du. Point-MaDi: Masked autoencoding with diffusion for point cloud pre-training. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/4809dd4b628b6253d0aad0154014f7a3-Abstract-Conference.html. 2

Mutian Xu, Junhao Zhang, Zhipeng Zhou, Mingye Xu, Xiaojuan Qi, and Yu Qiao. Learning geometry-disentangled representation for complementary understanding of 3d object point cloud. In *AAAI Conference on Artificial Intelligence*, volume 35, pp. 3056–3064, 2021. doi: 10.1609/aaai.v35i4.16414. 7, 16

Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-BERT: Pre-training 3d point cloud transformers with masked point modeling. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19313–19322, 2022. URL https://openaccess.thecvf.com/content/CVPR2022/html/Yu_Point-BERT_Pre-Training_3D_Point_Cloud_Transformers_With_Masked_Point_Modeling_CVPR_2022_paper.html. 3

Renrui Zhang, Ziyu Guo, Peng Gao, Rongyao Fang, Bin Zhao, Dong Wang, Yu Qiao, and Hongsheng Li. Point-M2AE: Multi-scale masked autoencoders for hierarchical point cloud pre-training. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 27061–27074, 2022. 3, 7, 8, 16

Xiangdong Zhang, Shaofeng Zhang, and Junchi Yan. PCP-MAE: Learning to predict centers for point masked autoencoders. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024. 3, 7, 8, 16

Yabin Zhang, Jiehong Lin, Ruihuang Li, Kui Jia, and Lei Zhang. Point-DAE: Denoising autoencoders for self-supervised point cloud learning. *IEEE Transactions on Neural Networks and Learning Systems*, 36(9):16279–16293, 2025. doi: 10.1109/TNNLS.2025.3557055. 3

Xiao Zheng, Xiaoshui Huang, Guofeng Mei, Yuenan Hou, Zhaoyang Lyu, Bo Dai, Wanli Ouyang, and Yongshun Gong. Point cloud pre-training with diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22935–22945, 2024. doi: 10.1109/CVPR52733.2024.02164. 1, 2, 3, 4, 6, 7, 8, 16, 17

A APPENDIX

The supplementary material provides additional technical formulations, architectural specifications, hyperparameter tables, baseline descriptions, per-corruption diagnostic analyses, and qualitative visualizations to complement the primary manuscript:

- §A.1: **Detailed Corruption Taxonomy and Formulations.** Mathematical definitions, severity parameters, and geometric interpretations for all seven corruption families.
- §A.2: **Detailed Evaluation Metric and Normalization Formulation.** Mathematical formulations of DGCNN-normalized Corruption Error (CE_c) and Mean Corruption Error (mCE).
- §A.3: **Detailed Network Architectures.** Layer-by-layer specifications of the Mask Encoder, Local Geometry Reconstructor, Condition Aggregation Network, and Conditional Flow Denoiser.
- §A.4: **Comprehensive Pretraining and Fine-tuning Hyperparameters.** Complete tabular overview comparing pretraining and downstream fine-tuning configurations.
- §A.5: **Extended Baseline Descriptions.** Detailed taxonomy, characteristics, and comparison contexts of all evaluated baselines.
- §A.6: **Per-Corruption Diagnostic Analysis.** Per-corruption comparisons between original PointDif and RAD, and corruption fingerprint visualizations.
- §A.7: **Pretrained Geometry Readout.** Controlled local-deletion protocol, five-level readout comparison, and category-wise analysis of original PointDif and RAD.
- §A.8: **Limitations and Future Work.** Discussion of current evaluation scope, scaling directions to large-scale scenes, and downstream computational confirmation.

A.1 DETAILED CORRUPTION TAXONOMY AND FORMULATIONS

Point-cloud corruption benchmarks test sensitivity to controlled changes in coordinates, point availability, and spurious observations. Both benchmarks use seven corruption families and five severity levels, yielding 35 conditions per benchmark. The definitions below follow the ModelNet-C (Ren et al., 2022b;a), with severity index $k \in \{1, \ldots, 5\}$; N denotes the input point count. We evaluate the released ScanObjectNN-C data (Wang et al., 2023) using its benchmark-specific normalization constants, rather than regenerating it from these formulas.

1. **Scale.** Independently sample three axis factors $a_j \sim \mathcal{U}(1/s_k, s_k)$, where $s_k \in \{1.6, 1.7, 1.8, 1.9, 2.0\}$. Apply $\mathrm{diag}(a_1, a_2, a_3)$ to the coordinates, then recenter and normalize the cloud to unit radius. This changes relative axis scales rather than only object size.
2. **Jitter.** Add independent Gaussian coordinate noise, $p_i' = p_i + \epsilon_i$, with $\epsilon_i \sim \mathcal{N}(0, \sigma_k^2 I)$ and $\sigma_k = 0.01k$. The released ModelNet-C generator does not clip this noise.
3. **Drop-Global (Drop-G).** Randomly permute the points and retain $\lfloor N(1 - r_k) \rfloor$ points, where $r_k \in \{0.25, 0.375, 0.5, 0.625, 0.75\}$. The retained count is fixed for each severity.
4. **Drop-Local (Drop-L).** Remove $100k$ points across a randomly chosen number of clusters between one and seven. The removal budget is randomly allocated across clusters; each step samples an anchor from the remaining points and removes its nearest neighbors. The anchors are not selected by FPS.
5. **Add-Global (Add-G).** Append $10k$ points sampled uniformly inside the unit sphere, giving 10–50 added points across the five levels.
6. **Add-Local (Add-L).** Append $100k$ points distributed across one to seven randomly chosen local clusters. Each cluster perturbs an input-point anchor with Gaussian noise whose standard deviation is sampled from $[0.075, 0.125]$; the implementation constrains added points to the unit sphere.
7. **Rotate.** Sample three Euler angles independently from $[-\theta_k, \theta_k]$, with $\theta_k = k\pi/30$, and apply the composed rotation.

A.2 DETAILED EVALUATION METRIC AND NORMALIZATION FORMULATION

To provide a calibrated assessment of corruption robustness across disparate corruption families, we detail the normalization scheme for the Mean Corruption Error (mCE) metric:

DGCNN-Normalized Corruption Error (CE_c)**.** Different corruption families have different reference error rates; normalization uses a separate DGCNN error for each family and benchmark. To prevent corruptions with naturally higher error rates from skewing the overall evaluation, the Corruption Error for each corruption category c ($c \in \{1, \ldots, 7\}$) is calibrated against the standard, unaugmented DGCNN (Wang et al., 2019) baseline, following ModelNet-C (Ren et al., 2022b):

$$\mathrm{CE}_c \;=\; \frac{1 - \mathrm{OA}^{(c)}_{\mathrm{method}}}{1 - \mathrm{OA}^{(c)}_{\mathrm{DGCNN}}}, \tag{6}$$

where $\mathrm{OA}^{(c)}_{\mathrm{method}}$ and $\mathrm{OA}^{(c)}_{\mathrm{DGCNN}}$ denote the Top-1 Overall Accuracy of the evaluated model and the DGCNN anchor on corruption c, averaged across all five severity levels. Under this formulation, $\mathrm{CE}_c < 1.0$ indicates that the method demonstrates superior resilience relative to DGCNN, whereas $\mathrm{CE}_c > 1.0$ indicates higher susceptibility.

Mean Corruption Error (mCE). The overall benchmark robustness metric, Mean Corruption Error (mCE), is computed as the unweighted arithmetic mean across all seven corruption categories:

$$\mathrm{mCE} \;=\; \frac{1}{7}\sum_{c=1}^{7} \mathrm{CE}_c. \tag{7}$$

Consistent with the published literature (Ren et al., 2022b;a; Wang et al., 2023; 2024), we report $100 \times \mathrm{mCE}$ rounded to one decimal place, where a lower numerical value signifies stronger overall corruption resilience.

A.3 Detailed Network Architectures

The remainder of this section specifies the pretraining network used by RAD. We first describe how an input cloud of $1{,}024$ points is partitioned into overlapping local patches and embedded by a shared Mini-PointNet. We then give the layer widths of the shared Mask Encoder, the Local Geometry Reconstructor, the Condition Aggregation Network, and the Conditional Flow Denoiser, including the positional embeddings, mask tokens, and the form of each training loss. Optimizer settings, epoch schedules, loss weights, and the downstream grouping choices are collected separately in Tab. 6. In particular, fine-tuning keeps 32 points per patch but uses 64 patches on ModelNet40 and 128 patches on ScanObjectNN; the reconstructor, CANet, and denoiser are not part of the classifier.

Patch partition and Mini-PointNet. Given an input point cloud with $N = 1,024$ points, Farthest Point Sampling (FPS) selects $G = 64$ cluster centers $p_g \in \mathbb{R}^3$. For each center, k-Nearest Neighbors (kNN) groups $S = 32$ points into a local patch $x_g \in \mathbb{R}^{32\times 3}$. Points in each patch are normalized to local coordinates $\tilde{x}_g = x_g - p_g$. The kNN neighborhoods can overlap; masking is applied to groups and does not guarantee disjoint underlying point sets. The shared Mini-PointNet first applies pointwise convolutions with channel widths $3 \to 128 \to 256$, with batch normalization and ReLU between the two convolutions. A max-pooled patch feature is broadcast and concatenated with each point feature, producing 512 channels. A second pointwise stack, $512 \to 512 \to 384$, with batch normalization and ReLU between its convolutions, followed by max pooling, yields one 384-dimensional patch embedding. The cluster center coordinates p_g are mapped to positional embeddings $\mathrm{PE}(p_g)$ via a 2-layer MLP $[3 \to 128 \to 384]$ with GELU activations. The encoder adds the center positional embedding before each Transformer block, rather than only once at the input.

Mask Encoder f_θ. We adopt the standard Vision Transformer architecture with 12 sequential encoder blocks and Pre-Layer Normalization. The latent embedding dimension is $C = 384$, with 6 self-attention heads (head dimension 64) and an MLP expansion ratio of 4 (feed-forward dimension 1536). We apply GELU activations with attention and MLP dropout set to 0.0. Stochastic-depth rates increase linearly from 0 to 0.1 across the 12 encoder blocks. The Transformer blocks process the 26 visible tokens remaining after masking $\lfloor 0.6 \times 64 \rfloor = 38$ patches.

Local Geometry Reconstructor (LGR). The decoder uses its own positional MLP $\mathrm{PE}_{\mathrm{dec}}$, with widths $3 \to 128 \to 384$ and a GELU activation, separate from the encoder positional MLP. For each masked index $g \in \mathcal{M}$, its input is $q_g = q + \mathrm{PE}_{\mathrm{dec}}(p_g)$, where $q \in \mathbb{R}^{384}$ is a shared learnable vector. We concatenate the 26 visible encoder tokens and 38 copies of q, add the decoder positional embeddings once, and process the resulting 64-token sequence with one Transformer self-attention

Table 6: Comprehensive hyperparameter specifications for RAD across self-supervised pretraining on ShapeNet and downstream fine-tuning on ModelNet40 and ScanObjectNN PB-T50-RS.

Configuration	**Pretraining**	**Fine-tuning (ModelNet40)**	**Fine-tuning (ScanObjectNN)**
Dataset	ShapeNet	ModelNet40	ScanObjectNN (PB-T50-RS)
Input points (N)	1,024	1,024	1,024
Number of patches (G)	64	64	128
Points per patch (S)	32	32	32
Mask ratio	60% ($\lvert\mathcal{M}\rvert = 38$)	—	—
Pretraining schedule (epochs)	300	—	—
Fine-tuning schedule (epochs)	—	300	300
Batch size	256	32	32
Optimizer	AdamW	AdamW	AdamW
AdamW β_1, β_2	0.9, 0.999	0.9, 0.999	0.9, 0.999
Base learning rate	2×10^{-3}	5×10^{-4}	5×10^{-4}
Weight decay	0.05	0.05	0.05
LR scheduler	Cosine annealing	Cosine annealing	Cosine annealing
Warmup epochs	10	10	10
LGR reconstructor depth	1	Discarded	Discarded
Reconstructor loss weight λ_0	1.0	Discarded	Discarded
LGR warmup epochs E_{w}	30	Discarded	Discarded
Denoiser loss weight	1.0	Discarded	Discarded
Downstream classifier head	—	MLP $[768, 256, 256, 40]$	MLP $[768, 256, 256, 15]$
Classifier dropout	—	0.5	0.5

block (384 channels, six heads, MLP ratio 4). After layer normalization, we retain the masked-slot outputs and apply $\mathrm{Linear}(384, 96)$, reshaping each prediction to $\hat{x}_g \in \mathbb{R}^{32\times3}$. The loss averages the symmetric Chamfer distance over corresponding masked patches in center-relative coordinates, using Euclidean distances without squaring.

Condition Aggregation Network (CANet). In the generative branch, masked slots are filled with a separate shared mask token $m \in \mathbb{R}^{384}$ ($m \neq q$), restoring a full-length sequence of $G = 64$ tokens. CANet first applies two 1×1 convolutions with channel widths $384 \to 512 \to 512$, each followed by ReLU. SoftPool aggregates across tokens; the resulting global feature is broadcast and concatenated with the token features. A second pair of 1×1 convolutions, $1024 \to 512 \to 768$, each followed by ReLU, and a final SoftPool produce $c \in \mathbb{R}^{768}$.

Conditional Flow Denoiser D_ψ. The denoiser is a six-layer pointwise MLP with widths $3 \to 128 \to 256 \to 512 \to 256 \to 128 \to 3$ and LeakyReLU activations after the first five layers. We encode $1000t$ with a 256-dimensional sinusoidal embedding followed by an MLP $256 \to 128 \to 128$ with SiLU, then concatenate it with c to form an 896-dimensional condition. Each layer uses this condition to predict channel-wise scale and shift, applying $h \odot (1+s)+b$ to its linear features without layer normalization; the modulation projections are zero-initialized. A residual connection adds the noisy input coordinates to the final MLP output to predict ε. Timesteps follow $t = \mathrm{sigmoid}(u)$ with $u \sim \mathcal{N}(-0.8, 0.8^2)$, and the loss is the mean squared error against the sampled Gaussian noise.

A.4 Comprehensive Pretraining and Fine-tuning Hyperparameters

As detailed in Tab. 6, we provide the comprehensive hyperparameter configurations across self-supervised pretraining and downstream classification fine-tuning.

The classifier concatenates the class token and max-pooled patch features before the three linear layers; each hidden layer uses batch normalization, ReLU, and dropout. CANet, the denoiser, and LGR are omitted from this downstream architecture. During ScanObjectNN fine-tuning, we apply random rotation about the y-axis with an angle sampled uniformly from $[0, 2\pi)$. The reconstruction targets are the center-relative coordinates of the input patches. Pretraining uses the 300-epoch cosine learning-rate schedule with a 10-epoch linear warmup. The reconstruction weight follows the schedule in Sec. 3, $\lambda(e) = \lambda_0 \cdot \min(1,\, e/E_{\mathrm{w}})$, with $\lambda_0 = 1.0$ and $E_{\mathrm{w}} = 30$. Whole-shape denoising remains active from the first epoch.

A.5 Extended Baseline Descriptions

As summarized across Tabs. 1 and 2, we benchmark RAD against representative supervised backbones (PointNet (Qi et al., 2017a), PointNet++ (Qi et al., 2017b), DGCNN (Wang et al., 2019), GDANet (Xu et al., 2021), PCT (Guo et al., 2021), PointNeXt (Qian et al., 2022)), self-supervised and generative pretraining models (Point-M2AE (Zhang et al., 2022), PCP-MAE (Zhang et al., 2024), PointMamba (Liang et al., 2024), PointGPT (Chen et al., 2023), MAP (Liu & Yi, 2025), PointDif (Zheng et al., 2024)), corruption robustness architectures (RPC (Ren et al., 2022b), APCT (Wang et al., 2024)), and parameter-efficient fine-tuning frameworks (PointGST (Liang et al., 2025)). Point-MAE (Pang et al., 2022), which reconstructs masked coordinates with a Chamfer loss, and AdaptPoint (Wang et al., 2023), which learns sample-adaptive adversarial augmentation, provide additional related-work context; neither has a standalone result row in these tables.

1. **Supervised Point Cloud Backbones.** These entries use supervised classification training rather than self-supervised pretraining:
   - **PointNet** (Qi et al., 2017a) applies point-wise MLPs and symmetric max-pooling to achieve permutation invariance, without explicit local neighborhood aggregation.
   - **PointNet++** (Qi et al., 2017b) introduces hierarchical ball querying and set abstraction levels, improving local feature learning over PointNet.
   - **DGCNN** (Wang et al., 2019) constructs dynamic kNN graphs in feature space via EdgeConv and serves as the normalization anchor for the corruption metrics used here.
   - **GDANet** (Xu et al., 2021) disentangles point cloud geometry into contour and flat components.
   - **PCT** (Guo et al., 2021) introduces Transformer self-attention to point cloud coordinate processing.
   - **PointNeXt** (Qian et al., 2022) revisits PointNet++ with modernized architectural improvements and training strategies.
2. **Self-Supervised and Generative Pretraining Methods.** These approaches learn transferable representations from unlabeled point clouds via masked autoencoding, autoregressive sequence modeling, or conditional diffusion. They provide direct comparisons for assessing the joint use of local coordinate reconstruction and whole-shape denoising in RAD:
   - **Point-M2AE** (Zhang et al., 2022) extends masked autoencoding through hierarchical reconstruction with progressive downsampling.
   - **PCP-MAE** (Zhang et al., 2024) predicts positional embeddings for masked patch centers and uses the predicted centers in place of the provided masked-center embeddings during reconstruction.
   - **PointGPT** (Chen et al., 2023) adapts autoregressive generative sequence modeling to point clouds, sorting patches along spatial trajectories.
   - **PointMamba** (Liang et al., 2024) combines a state-space backbone with serialized point tokens and masked modeling for pretraining.
   - **MAP** (Liu & Yi, 2025) uses masked autoregressive pretraining with a hybrid Mamba–Transformer backbone.
   - **PointDif** (Zheng et al., 2024) learns representations through globally conditioned point-cloud recovery from Gaussian noise.
3. **Dedicated Corruption Robustness Frameworks.** These methods address corruption robustness through specialized classifier design or robust supervised training strategies, whereas RAD introduces its additional supervision purely during pretraining without modifying the downstream classification architecture:
   - **RPC** (Ren et al., 2022b) combines kNN grouping, frequency-based aggregation, and self-attention based on architectural choices systematically studied in the ModelNet-C benchmark.
   - **APCT** (Wang et al., 2024) uses an Adversarial Significance Identifier and a Target-guided Promptor to guide significance-based token dropping within self-attention during supervised training.
4. **Parameter-Efficient Fine-Tuning.** These frameworks explore parameter-efficient adaptation by freezing the pretrained backbone and inserting lightweight trainable adapters, providing comparisons with an alternative downstream adaptation strategy:

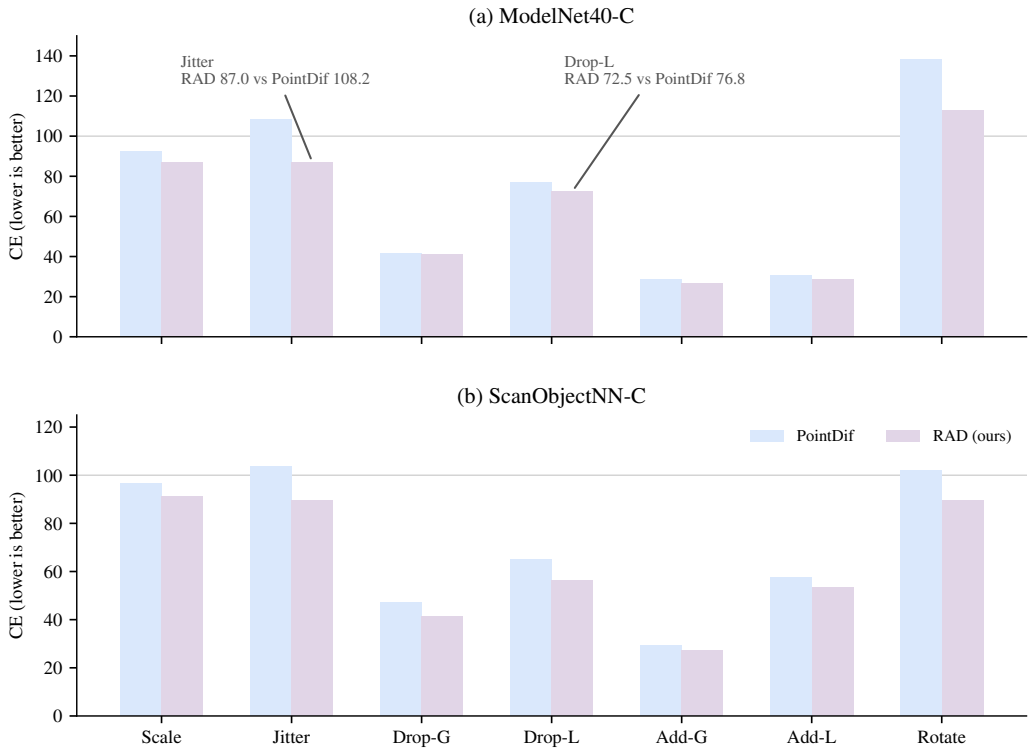


Figure 5: Per-corruption CE for original PointDif (Zheng et al., 2024) and RAD; lower is better. The bars report the seven corruption families on ModelNet-C and ScanObjectNN-C.

- **PointGST** (Liang et al., 2025) introduces trainable spectral adapters while keeping the pre-trained backbone frozen. The main tables include both its Point-MAE and PointGPT-S adapted variants.

A.6 Per-Corruption Diagnostic Analysis

Tabs. 1 and 2 compare the original PointDif recipe with RAD under the reported benchmark protocols. The per-corruption discussion below uses those two methods only.

Per-corruption performance on synthetic ModelNet-C. As summarized in Tab. 1 and visualized in Fig. 5, RAD achieves consistently lower corruption error than the original PointDif baseline across all seven corruption families. The largest margins occur for Rotate and Jitter, where Rotate error drops by 25.1 points from 138.1% to 113.0%, and Jitter error decreases by 21.2 points from 108.2% to 87.0%. Furthermore, RAD exhibits steady resilience against topological degradation and noise insertion, reducing Drop-Local error from 76.8% to 72.5%, Add-Global from 28.8% to 26.8%, Add-Local from 30.5% to 28.7%, and Drop-Global from 41.5% to 41.1%, alongside a 5.4-point reduction on Scale from 92.6% to 87.2%. These comprehensive gains confirm that localized coordinate reconstruction effectively complements whole-shape diffusion smoothing, stabilizing synthetic CAD representations against both global pose variations and localized metric distortions.

Per-corruption performance on scanned ScanObjectNN-C. On ScanObjectNN-C, which applies synthetic corruptions to object point clouds derived from RGB-D scans, RAD outperforms original PointDif across every corruption category in Tab. 2. On this scanned benchmark, Jitter and Rotate again yield the largest improvements, reducing CE by 14.2 points from 103.7% to 89.5%, and by 12.4 points from 101.9% to 89.5%, respectively. Under synthetic point deletion, Drop-Local error decreases by 8.6 points from 65.0% to 56.4%, and Drop-Global declines by 5.8 points from 47.1% to 41.3%. Together with reductions of 5.5 points on Scale from 96.7% to 91.2%, 4.4 points on Add-Local from 57.7% to 53.3%, and 1.9 points on Add-Global from 29.1% to 27.2%, these results

demonstrate improved classification robustness to the benchmark's synthetic corruptions on scanned objects.

Corruption fingerprint diagnostics. The grouped bar visualization in Fig. 5 profiles the complete corruption fingerprint across both benchmarks, providing a holistic comparison between original PointDif and RAD. RAD has lower CE in all seven families on both datasets, so its improvement over original PointDif is not confined to one corruption type. This comparison establishes the performance difference between the complete pretraining recipes; the CE profiles alone do not identify the mechanism underlying that difference.

A.7 Local Geometry Readout from Pretrained Encoders

To assess accessible local geometry beyond classification mCE, we freeze the pretrained encoders from original PointDif and RAD and independently train readers of identical architecture to predict truncated distances to the clean point set. The encoders receive all groups of the current input without additional token masking. This diagnostic compares the two pretraining recipes under a shared reader protocol; it measures geometric decodability rather than an isolated effect of LGR. The readout depends jointly on encoder features, the reader's capacity and training, and the observed patch centers; it is not a direct measure of intrinsic encoder fidelity.

Experimental protocol. We partition the official ModelNet training set into 8,856 reader-training objects and 984 validation objects. Evaluation uses 2,404 test shapes, excluding the 64 official-test objects inspected during exploratory reader validation. This evaluation directly tests the frozen pretrained representations rather than fine-tuned classifiers. Each object comprises 1,024 sampled coordinates and two fixed corruption instances. For each instance, a shared anchor defines nested local deletions of 10%, 20%, 30%, 40%, and 50% of the original points, identically across methods. The first three levels correspond to Fig. 3 in the main text; the last two extend the analysis to larger deletions. These are controlled diagnostic inputs, distinct from the released ModelNet-C corruptions. The reader consumes final encoder tokens alongside patch centers computed from the current corrupted input, while clean ground-truth targets are utilized solely for supervision and evaluation. Architecturally, a single cross-attention block maps token representations to spatial grid queries with 128 hidden channels, four attention heads, and a feed-forward dimension of 256, followed by a scalar sigmoid prediction head. All readers are optimized on clean point clouds for 50 epochs using AdamW with an initial learning rate of 10^{-3}, weight decay of 0.05, batch size of 16, a five-epoch linear warmup, and cosine decay. Training samples 2,048 spatial queries per object and minimizes weighted absolute error, with weights computed from full-grid region counts to balance $d_X(q) < 0.2r_\star$ against $d_X(q) \geq 0.2r_\star$. This training partition differs from the narrower evaluation band $d_X(q) \leq 0.05r_\star$ used for E_N. Model checkpoints are selected according to the minimum balanced loss on the clean validation grid.

Distance field formulation. For a clean discrete point set X, the prediction target at spatial query q is its normalized, truncated nearest-neighbor distance:

$$t_X(q) = \frac{\min\{d_X(q), \tau\}}{\tau}, \qquad d_X(q) = \min_{x \in X} \|q - x\|_2, \tag{8}$$

where the truncation threshold is set to $\tau = 0.2r_\star$, and queries span a uniform 24^3 grid over $[-1.2r_\star, 1.2r_\star]^3$. The reference scale $r_\star$ represents the median bounding radius computed over reader-training shapes and remains fixed across corrupted test instances. We report the mean absolute prediction error in the band $d_X(q) \leq 0.05r_\star$ around the clean point set, denoted as E_N (near-surface error). The target is defined by discrete reference points, so this metric does not directly evaluate reconstruction of a continuous surface. These values represent normalized geometric distance discrepancies rather than percentage ratios. We first average the two instances within each object and then average across objects. The near-surface metric uses 2,401 test objects: the same three objects have no queries in the fixed near-surface band for either method and are excluded by this model-independent rule.

Readout across deletion levels. Fig. 6 extends the main-text comparison to all five deletion levels. RAD has lower mean near-surface error than original PointDif at every level, including reductions from 0.2871 to 0.2846 at 40% deletion and from 0.3648 to 0.3634 at 50%.

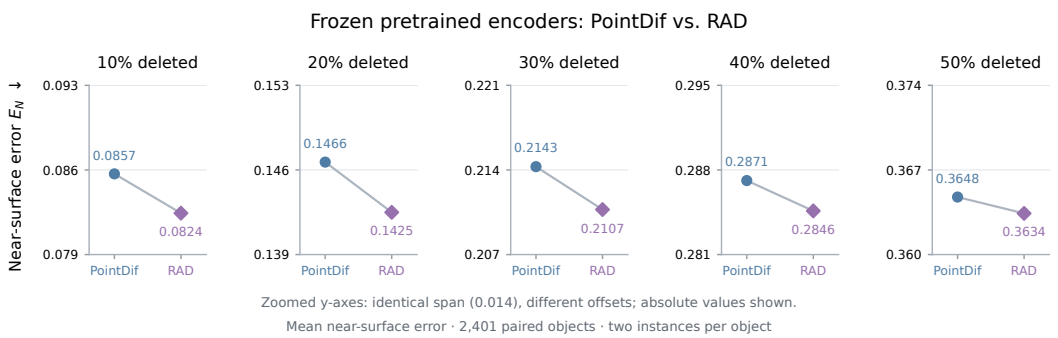


Figure 6: Near-surface readout error ($E_N \downarrow$) for frozen pretrained PointDif and RAD encoders at five controlled local-deletion levels. Each point averages two instances per object and 2,401 paired objects. All panels use a zoomed vertical range of 0.014 with different offsets; labels report absolute errors.

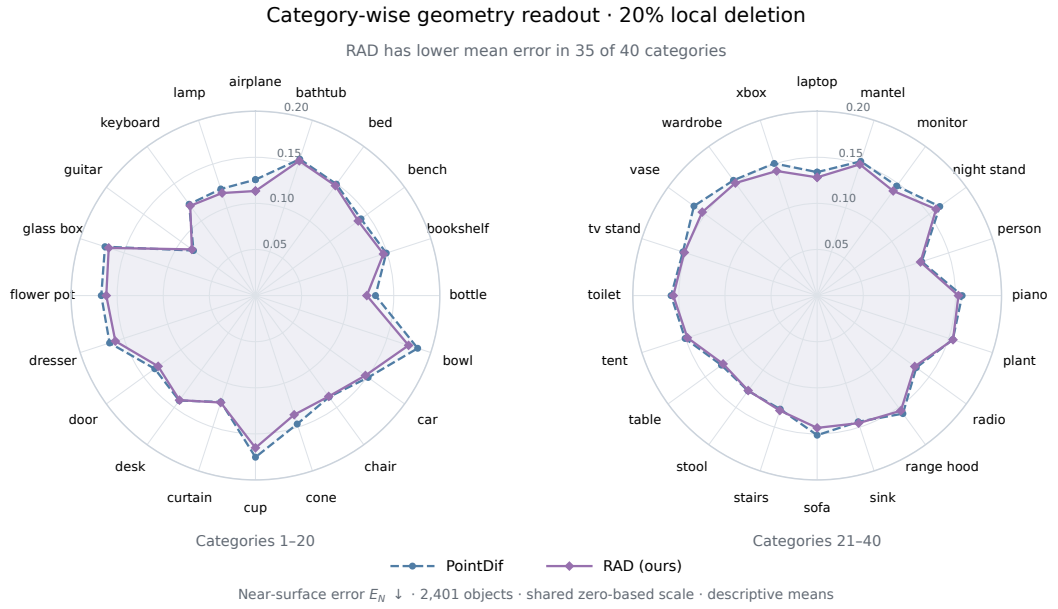


Figure 7: Category-wise near-surface readout error ($E_N \downarrow$) at 20% local deletion. Both panels use the same zero-based radial scale and together include all 40 categories. Means average instances within objects and then objects within each category; RAD has lower mean error in 35 categories. This is an exploratory breakdown at one fixed deletion level.

Category-wise readout. Fig. 7 breaks down the 20% deletion condition by all 40 ModelNet categories, using the same paired objects and reader checkpoints as the aggregate comparison. RAD has lower mean error in 35 categories, with the largest absolute reductions on airplane (0.0123), vase (0.0112), cone (0.0106), bowl (0.0104), and cup (0.0102). PointDif has lower mean error on curtain, guitar, plant, sink, and stairs, so the aggregate advantage is widespread but not universal. These are descriptive category means at a common severity, rather than tests of category-wise statistical significance.

Together, the severity and category views support improved local-geometry readout from RAD's pretrained representations under this protocol. They do not establish accurate surface completion or a causal explanation for the downstream classification gains.

A.8 Limitations and Future Work

While RAD demonstrates marked robustness across synthetic CAD models and authentic scanned objects, our current evaluation focuses primarily on object-level point cloud recognition, which represents the established benchmark testbed in corruption robustness literature. An appealing future direction is extending reconstruction-anchored diffusion pretraining to large-scale, unbounded outdoor LiDAR scenes in autonomous driving and mobile robotics. Such outdoor environments present vast spatial extents, non-uniform sensor range distributions, and varying point densities, which can motivate future explorations into multi-scale voxel tokenization and sparse coordinate decoders.