Naïve Difficulty Conditioning Fails on ShapeNet-55: A Matched-Control Audit

Anonymous authors
Paper under double-blind review

Abstract

We audit a naïve difficulty-conditioned extension of AdaPoinTr and show that the apparent +14%–21% per-bin CD-ℓ_2 gains we observed for it on ShapeNet-55 are evaluation-discipline artefacts rather than method effects. Warm-starting from a fully-trained checkpoint inflates apparent per-bin gains across Simple/Moderate/Hard irrespective of method, across two independent training runs (a pilot recorded per-bin at the time and a seed-matched rerun whose aggregate is released in the bundle)—an inflation visible only under a matched warm-start control. We propose a *stratified matched-control protocol* that pins warm-start, schedule, crop path, and pre-specified decision thresholds. Applied as a bounded case study to a natural DiffCond extension, the protocol uncovers a 5.48% Hard and 4.56% Mean CD-ℓ_2 regression across three seeds, missing the pre-specified acceptance threshold on every bin. Four diagnostic probes localise the failure to a dead ranking-delta branch (architecturally non-differentiable) and to an auxiliary cross-entropy loss as the largest *identified* contributor. A data-side probe (Hard-Specialist) gives an *exploratory point estimate* of $+7.38\% \pm 3.82$pp on Hard across three seeds, with a 95% CI that crosses zero; a broader Easy-side probe gives no Simple-bin signal. We report this contrast descriptively, not as a tested general principle. A cross-backbone replication on UpTrans (4 seeds) yields no (mode, metric) cell clearing the pre-specified +3% method-bar (largest paired mean +1.79%, 95% CI [−2.72%, +6.29%]); a 2-seed PCN sanity check (underpowered) holds all deltas within $\sim$1.1%. Under matched control, neither backbone reproduces the apparent double-digit signature: warm-start mismatch, not difficulty conditioning, drove the original gain.

1 Introduction

Point cloud completion—the task of recovering a full 3D shape from a partial observation—has progressed rapidly with transformer architectures such as PoinTr (Yu et al., 2021) and AdaPoinTr (Yu et al., 2023), which report averaged Chamfer distances that are routinely compared across papers as evidence of method-level progress. The accompanying stratified Simple/Moderate/Hard protocol, however, is vulnerable to training-time and pipeline choices that are routinely left uncontrolled—and small choices among them can flip the sign of an apparent improvement. While auditing a natural difficulty-conditioned extension of AdaPoinTr in our own pilot experiments, we observed an apparent +14%–21% improvement across S/M/H bins under uncontrolled training-time settings. The same extension instead *regressed* by $\sim 5\%$ once these confounds were controlled—a sign-flip on identical trained weights, robust across two independent training runs (the follow-up rerun aggregate is released in the bundle).

The size of this swing motivates our central claim: in current practice, averaged ShapeNet-55 reporting can attribute discipline artefacts to method, and a stratified matched-control protocol is needed to separate them. This paper makes four contributions, expanded in the *Contributions* paragraph below: (i) a stratified matched-control protocol that exposes the three confounds; (ii) a mechanistic failure analysis of a representative naïve difficulty-conditioned extension (DiffCond) on AdaPoinTr that, under 3-seed matched control, regresses on every bin, including a 5.48% Hard-bin CD-ℓ_2 degradation; (iii) a constructive data-side counterexample (Hard-Specialist) together with an Easy-side null (EasyWide-Specialist); and (iv) a cross-backbone replication on UpTrans (4 seeds) that finds no +3%-bar improvement on any (mode, metric) cell.

We claim neither a new state-of-the-art nor a general principle of difficulty conditioning; our contribution is a methodological discipline together with a bounded empirical regularity on AdaPoinTr/UpTrans and ShapeNet-55.

Three pitfalls in current practice. ShapeNet-55 (Yu et al., 2023) evaluates completion under three difficulty levels (Simple, Moderate, Hard) defined by the fraction of points removed, using 8 fixed viewpoints per test sample. During our case-study audit we identified three confounds, each capable of inflating apparent gains by several percentage points in the absence of any methodological change, all reproducible under the official PoinTr-repository defaults:

1. **Random-crop proxy bias.** Training-time validation uses a random crop ratio drawn uniformly from $[0.25, 0.75]$, whereas official evaluation uses three fixed crop ratios at 8 deterministic viewpoints. We show that a method exhibiting 2.15% improvement on the random-crop proxy achieves no meaningful improvement on any bin of the official Simple/Moderate/Hard protocol (§3.1).

2. **Warm-start inflation.** Initialising from a fully-trained baseline and fine-tuning for 30 epochs can yield large double-digit ($\sim$+14%–21%) apparent gains across *all* difficulty bins—a signature of the warm-start advantage, not the method itself (§3.2).

3. **Crop-path mismatch.** Nominally equivalent crop pipelines—per-sample list-crop versus batch-level integer-crop with farthest-point sampling (FPS)—produce measurably different augmentation distributions, introducing latent confounders into comparisons even when model weights and hyper-parameters are shared (§3.3).

Stratified matched-control protocol. To neutralise these confounds we adopt the following discipline: (i) pin the warm-start source and fine-tuning schedule across all candidates; (ii) share a single crop-path pipeline; (iii) pre-specify decision thresholds *before* running the final comparison; and (iv) report per-bin metrics (F-Score@1%, CD-ℓ_1, CD-ℓ_2) rather than overall averages alone.

Case study: naïve difficulty conditioning. As a concrete case study we implement a representative "DiffCond" extension of AdaPoinTr that adds a difficulty embedding, a per-query ranking delta, a gated decoder injection, and a lightweight auxiliary predictor. Under 3-seed matched control, DiffCond regresses on every difficulty bin: Hard-bin CD-ℓ_2 by 5.48%, Moderate by 3.09%, Simple by 4.18%, and Mean by 4.56% relative to a matched vanilla control. Across the three bins $\times$ two inference settings (Oracle and Predict), all six paired-Δ entries are negative with no seed producing a positive outcome—we report this as a descriptive sign-consistency note rather than as six independent significance tests, since Oracle and Predict share the same trained checkpoint and the predictor tracks the oracle to within 0.004 CD-ℓ_2 per cell (supplementary). We trace the failure to four converging pieces of evidence: (i) the per-query ranking-delta branch is dead by construction (`argsort`+`gather` zeroes its gradient; the last-layer weight is identically zero after 30 epochs); (ii) an epoch-0 official-eval probe confirms that the DiffCond model is mathematically equivalent to vanilla at initialisation (within 0.001 CD-ℓ_2 on Simple, bit-identical on Moderate and Hard; no bitwise audit), so the regression is consistent with a training-time effect rather than with an init-disparity explanation; (iii) the $\sim$2.8$\times$ apparent training-loss gap at epoch 0 is largely accounted for by a $0.1 \cdot \ln 3$ auxiliary cross-entropy (CE) term folded into the reconstruction column; and (iv) test-time ensembling over difficulty conditions cannot recover matched-vanilla quality (even per-sample oracle selection over three conditionings leaves a 9.86% residual gap against the matched vanilla on a Hard-bin ensemble probe, §5.4). A *single-seed* 2$\times$2 ablation (rank $\times$ CE) attributes the largest identified share of the regression to the auxiliary CE loss as an *exploratory mechanism hypothesis*, not as a settled causal decomposition; the ranking path is architecturally dead by construction (it carries no learned signal because the downstream `argsort`+`gather` operator is non-differentiable w.r.t. the ranking scores, see §6), so its contribution to the matched-control regression is at most a structural side-effect rather than a learned-weight performance driver. We do not claim a fully-settled causal decomposition under the single-seed ablation data.

Contributions.

- We propose a *stratified matched-control evaluation protocol* with pre-specified thresholds, applicable to any point cloud completion method, that systematically exposes the three pitfalls above (§3).
- We present a mechanistic *failure analysis* of naïve difficulty conditioning. Four independent lines of evidence implicate two distinct DiffCond components: (i) the auxiliary cross-entropy loss is the largest *identified* driver of the regression under our single-seed ablation matrix (we report it as the leading contributor rather than as a fully settled causal decomposition); and (ii) the non-differentiable ranking-delta path is *architecturally dead*—zero gradient flows through the `argsort`+`gather` discrete-selection operator (a well-documented failure mode for differentiation through discrete operations (Kool et al., 2019)), and the last-layer weight remains identically zero after 30 epochs of training. The latter is therefore a design defect rather than a learned-weight contributor to the regression (§5, §6).
- We report an *exploratory data-side probe* testing whether the same difficulty axis that resists architectural conditioning is exploitable through the training data: a Hard-range fine-tune (Hard-Specialist) gives an exploratory point estimate of $+7.38\% \pm 3.82$pp on Hard across three seeds (95% CI crosses zero), while an analogous wider Easy-side probe yields no Simple-bin signal ($-0.25\% \pm 2.03$pp across the same three seeds). We report this contrast descriptively, as a Hard-side point estimate paired with an Easy-side null on a broader probe on AdaPoinTr/ShapeNet-55, not as a tested general principle of target-range specialisation (§5.6).
- We provide a *cross-backbone replication* on a different transformer-completion backbone (UpTrans) under the same matched-control discipline. The largest observed paired CD-ℓ_2 improvement is $+1.79\%$ on Hard with a 95% CI of $[-2.72\%, +6.29\%]$—below the pre-specified $+3\%$ method-bar lower bound and well short of the double-digit AdaPoinTr inflation signature, supporting the cautious reading that no large architecture-general gain emerges from a naïve DiffCond extension across the two transformer decoders we test (§5.7).

2 Related Work

Point cloud completion. Learning-based completion has progressed from PointNet (Qi et al., 2017)-based decoders (PCN (Yuan et al., 2018), TopNet (Tchapmi et al., 2019)) to folding and voxel-gridded generators (FoldingNet (Yang et al., 2018), GRNet (Xie et al., 2020)). Transformer architectures (PoinTr (Yu et al., 2021), AdaPoinTr (Yu et al., 2023)) and seed-based upsampling (SeedFormer (Zhou et al., 2022), SnowflakeNet (Xiang et al., 2021)) currently dominate the S/M/H ShapeNet-55 benchmark. A concurrent study (Pathak et al., 2025) questions benchmark adequacy via a real-world dataset, addressing the synthetic-vs-real gap rather than within-benchmark methodological confounds that are our focus.

Conditional learning and evaluation methodology. Feature-wise linear modulation (FiLM (Perez et al., 2018)) and cross-attention conditioning are standard in image generation and NLP; in 3D completion, conditioning on observed-point statistics or task difficulty has been explored implicitly through curriculum learning and loss reweighting, but, to our knowledge, explicit difficulty-level conditioning via learned embeddings injected into the decoder has not been systematically evaluated under matched control. Critiques of benchmark practice are well established in 2D vision but less explored in 3D completion, where ShapeNet-55's stratified protocol (Yu et al., 2023) is often reported only through its overall average, masking per-bin regressions.

Reproducibility and methodological critique. Methodological reality-checks are an established genre. Specific instances exist for reinforcement learning (Henderson et al., 2018), metric learning (Musgrave et al., 2020), and ML benchmarking practice broadly—checklisted protocol reporting (Pineau et al., 2021), principled experimental presentation (Dodge et al., 2019), and explicit benchmark-variance accounting (Bouthillier et al., 2021). The recurring finding is that uncontrolled training-time and pipeline choices can dominate the nominal method gap, so apples-to-apples comparison requires pinning these factors. In statistics, matched designs have long served as the corresponding tool for separating pipeline-induced effects from treatment

effects (Rosenbaum, 2002; Stuart, 2010). We bring this discipline to point-cloud completion at the *within-benchmark* granularity: a same-dataset critique that operates beneath the averaged CD figure rather than against a different distribution. The most closely related point-cloud work, Pathak et al. (2025), probes completion benchmark adequacy through a real-world dataset contrast; we instead localise three *synthetic-benchmark* confounds in the existing ShapeNet-55 protocol, and propose an evaluation-discipline protocol to neutralise them (§3). The two readings are complementary: their work establishes that synthetic-benchmark training does not transfer cleanly to real-world inputs, while ours establishes that within-synthetic-benchmark method comparisons may themselves be confounded.

Curriculum and difficulty-conditioned learning. Difficulty-aware training has been studied extensively as curriculum learning (Bengio et al., 2009; Hacohen & Weinshall, 2019), typically via sample scheduling, and as loss reweighting that emphasises hard examples (focal loss (Lin et al., 2017), online hard example mining (Shrivastava et al., 2016)), both of which act through the loss surface rather than through model-internal conditioning. In parallel, conditioning-by-embedding (FiLM (Perez et al., 2018), cross-attention) is a standard generation primitive. Our DiffCond architecture (§4) is the *combination of these two*: a difficulty token injected into the completion decoder. It represents the most direct reduction of curriculum-style ideas to modern architectural conditioning, and is precisely the design that an evaluation protocol should be able to audit under matched control.

3 Stratified Matched-Control Protocol

The ShapeNet-55 evaluation protocol is precisely specified, yet the surrounding training and reporting practices remain underspecified, and small choices among them can flip the sign of an apparent improvement. During our case study we identified three confounds, each capable of inflating an apparent improvement by several percentage points in the absence of any methodological change. Statistically, the three pitfalls correspond to measurement-error / proxy-validity (Pitfall 1), classical confounding (Pitfall 2), and pipeline-induced selection bias (Pitfall 3). Figure 1 visualises the three pitfalls; the five-principle discipline that neutralises them is enumerated in §3.4 and formalised in the rest of this section.

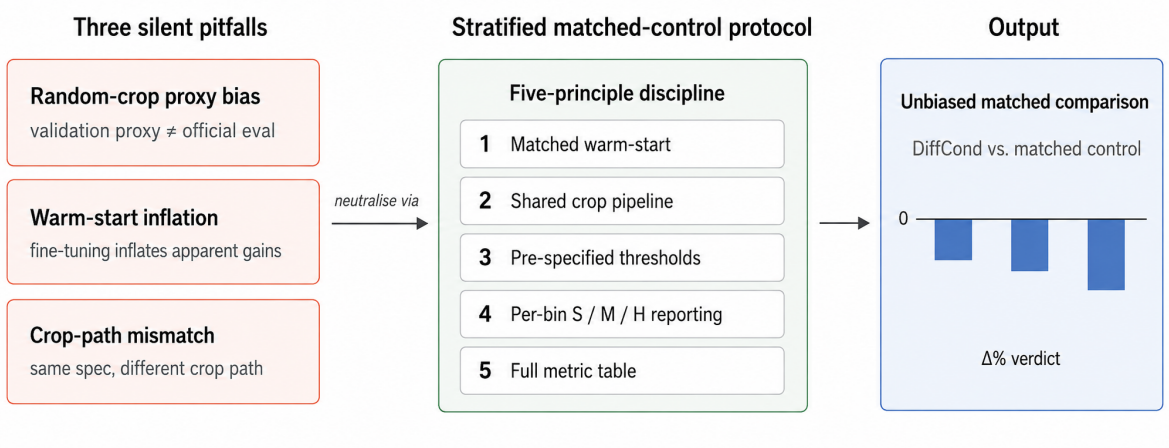

Figure 1: Three commonly overlooked ShapeNet-55 evaluation pitfalls, each capable of inflating apparent gains by several percentage points without any methodological change. Pitfall 1 *(random-crop proxy bias)*: a per-sample random-crop validation signal does not transfer to the official stratified Simple/Moderate/Hard bins (§3.1). Pitfall 2 *(warm-start inflation)*: fine-tuning from a pretrained checkpoint inflates apparent per-bin CD-ℓ_2 gains irrespective of method (§3.2). Pitfall 3 *(crop-path mismatch)*: nominally identical crop specifications produce measurable CD-ℓ_2 differences depending on whether a per-sample list-crop or batch int-crop+FPS path is used (§3.3). The five-principle stratified matched-control protocol that neutralises all three, its pre-specified ±3% method-bar, and its GO / NO-GO / REVERSE-SEVERE verdict are defined in §3.4; quantitative effects appear in §3.1–§3.3.

3.1 Pitfall 1: Random-Crop Proxy Bias

The proxy distribution used for training-time validation differs structurally from the distribution used for official evaluation, yet the two are routinely treated as exchangeable. The training-time validation in the public PoinTr code path draws a crop ratio $r \sim \mathcal{U}(0.25, 0.75)$ per sample through a per-sample list-based routine; the official protocol instead uses eight *fixed* viewpoints per sample, three fixed crop ratios, and a deterministic batch-level integer-crop pipeline followed by FPS resampling to 2048 points. A gain on the former does not entail a gain on the latter.

We encountered this exact effect when auditing an error-aware self-refinement extension: random-crop validation showed a 2.15% improvement (best val CD-ℓ_2 0.7815 at epoch 8 vs. baseline 0.7987), yet the epoch-45 checkpoint of the same training run, evaluated under the official S/M/H protocol, showed no meaningful improvement on any bin. Two consistency probes on the ep-45 checkpoint corroborated the spurious signal: oracle gating was the *worst* of four gate variants on Hard (1.390 vs. 1.350), and iterative self-refinement monotonically *degraded* Hard CD-ℓ_2 across iteration counts 1 through 5—both contrary to the behaviour expected of a correct refinement operator. These numbers are reported descriptively as a motivating development observation; the headline reproducibility evidence in this paper centres on the matched-control protocol of §3.4 and its application in §5.

3.2 Pitfall 2: Warm-Start Inflation

Fine-tuning from a fully-trained checkpoint and training from scratch are not directly comparable settings, yet fine-tuning vs. from-scratch comparisons are easy to mix during method development; we hit this exact pitfall during our own audit. Our headline evidence for warm-start inflation is the reproducibility rerun released as `data/results/from_scratch_baseline_smh.json`: it repeats the from-scratch baseline at seed 42 and computes seed-matched paired gains against the DiffCond Oracle (also at seed 42), yielding +17.67%, +18.68%, and +14.92% on Simple/Moderate/Hard CD-ℓ_2 respectively. An earlier pilot recorded at the time (raw aggregate not retained in the bundle, only the per-bin numbers were kept) gave roughly +18%, +21%, and +17% on the same bins; the bundled rerun corroborates the pilot's direction and order of magnitude, and the minor differences (Simple bit-near-exact; Moderate and Hard roughly 2 pp lower) are within the `cudnn.benchmark=True` nondeterminism envelope accumulated over a 30-epoch from-scratch trajectory and are documented in the artefact's provenance block. Across both runs the pattern is a near-uniform double-digit warm-start inflation across all three difficulty bins. The signature is diagnostic: warm-start dominates any method-specific contribution. A matched warm-start control (same checkpoint, schedule, and seed) eliminates this inflation and reveals the true comparison direction (Table 1).

3.3 Pitfall 3: Crop-Path Mismatch

A subtler confound lies in the data path itself. Within the PoinTr code path, two implementations that both nominally apply $\mathcal{U}(0.25, 0.75)$ random crop followed by FPS-to-2048 can still produce different evaluation results, because their crop *granularity* differs: the stock vanilla baseline draws a crop ratio *per sample* via a list routine before the internal FPS step, whereas the DiffCond path draws one crop ratio *per batch* and integer-indexes the crop count before FPS-to-2048. The two paths produce measurable CD-ℓ_2 differences attributable to crop-path granularity alone—differences that remain invisible without a byte-for-byte pipeline comparison.

We adopt the DiffCond path (batch-level int-crop + FPS) as the shared pipeline for the matched-control comparisons that follow, since it is the more restrictive of the two. Like the Pitfall 1 audit above, the crop-path mismatch is reported as a development-stage observation that motivated the protocol of §3.4; the headline reproducibility evidence is the matched-control comparison itself.

3.4 The Protocol

Building on these three observations, we define the *stratified matched-control protocol* used in the remainder of the paper. Throughout, we estimate a within-seed paired treatment effect $\Delta = (V - D)/V$, where V is the

matched vanilla CD-ℓ_2 and D is the DiffCond CD-ℓ_2, with seed, initialisation, schedule, and pipeline pinned as matched covariates; we do not bound unmeasured-confounder sensitivity (no Rosenbaum Γ analysis) and do not claim transportability beyond the two backbones tested.

1. **Matched warm-start.** All candidates in a comparison share the same pre-trained checkpoint, the same optimiser, the same learning-rate schedule, the same number of fine-tuning epochs, and the same random seed. Differences from the shared configuration are enumerated explicitly in the experiment setup.

2. **Shared crop pipeline.** One crop-path is selected before the comparison and used unchanged across all candidates. Any deviation is treated as an additional independent variable, not as an implementation detail.

3. **Pre-specified decision thresholds.** Before running the matched comparison we declare quantitative GO / NO-GO / REVERSE thresholds on the headline metric, in order to structurally preclude post-hoc rationalisation. *Throughout the paper, "pre-specified" means declared in advance of the final comparison run as a property of this protocol; we did not file a public preregistration.* Our thresholds for Section 5 are:

   - **Main-comparison verdict** (§5.2, DiffCond vs. Vanilla-FT): *GO* if Mean CD-$\ell_2 \geq +3\%$, or (Hard $\geq +3\%$ and Mean ≥ 0); *NO-GO* if $|\text{Hard}| < 2\%$ and $|\text{Mean}| < 2\%$; *REVERSE-SEVERE* if Mean $\leq -3\%$ or Hard $\leq -3\%$. The asymmetry between the GO bar and the REVERSE-SEVERE bar is intentional: the positive GO bar requires no mean harm in addition to a per-bin gain, whereas the negative REVERSE-SEVERE bar flags harm conservatively under either a Mean or a Hard regression alone.
   - **Secondary-method bar** (§5.4, §5.6): any add-on or data-side method must achieve a $\geq +1.5\%$ headline-bin improvement over its corresponding matched baseline to count as a positive contribution. The lower threshold reflects the narrower comparison (fixed model, varying intervention).

4. **Per-bin reporting.** Results are reported at each of the three difficulty bins separately; averaged-only reporting can hide a large per-bin regression behind a flat mean.

5. **Full metric table.** F-Score@1%, CD-ℓ_1 and CD-ℓ_2 are reported for every bin and every candidate. Averages across bins are an auxiliary summary, not the headline.

The discipline incurs no additional GPU cost beyond that of the comparison itself, and it controls the three named confounds in the present study. We position it as an evaluation-discipline contribution rather than as a claim that other, unnamed confounds are excluded. The remainder of the paper applies this protocol to a case study.

4 Naïve Difficulty Conditioning: Architecture

We describe the difficulty-conditioned extension of AdaPoinTr that forms the case study for the rest of the paper. The architecture is a natural combination of well-established conditioning primitives (learned discrete embedding, FiLM-style residual, gated injection, auxiliary classifier); we do not claim novelty and treat it as the "obvious" extension an evaluation protocol should be able to audit.

We choose this extension as our case study for three reasons: (a) it is the most direct architectural reduction of curriculum-style difficulty awareness to the modern transformer-completion setting; (b) AdaPoinTr is the strongest published ShapeNet-55 transformer backbone at the time of writing, so any positive result would plausibly generalise to follow-up work; and (c) the extension's expected benefit is concentrated on the Hard bin, making it a particularly informative test of stratified matched-control sensitivity. Figure 2 gives the full pipeline; the dead ranking-delta branch (red double-bar "no grad" marker) that later localises the failure is visible by inspection.

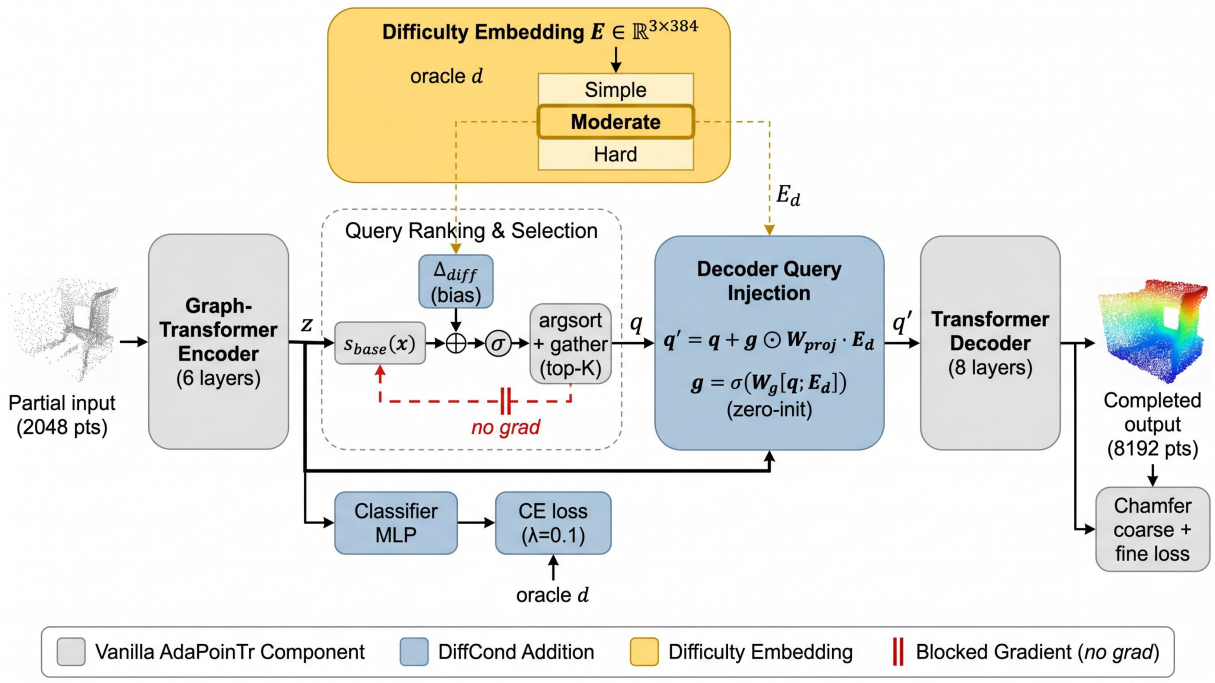

Figure 2: DiffCond architecture. Vanilla AdaPoinTr components in gray (graph-transformer encoder, transformer decoder, per-query ranking, Chamfer losses); the blue DiffCond additions form three groups (ranking residual Δ_{diff} inside the dashed Query Ranking & Selection box, gated injection into decoder queries, and an auxiliary CE classifier head); and the yellow difficulty embedding $\mathbf{E}$. The red double-bar "no grad" marker on `argsort+gather` marks the non-differentiable step that blocks gradient flow to Δ_{diff}; empirically this leaves the branch at zero-weight after 30 epochs (§6.1).

4.1 Architectural Additions

Starting from AdaPoinTr (denoted `base_model`) with the PoinTr-repository default ShapeNet-55 hyperparameters (§5.1) we add four modules, each prefixed `diff_cond_` (Fig. 2): a difficulty embedding $\mathbf{E} \in \mathbb{R}^{3\times384}$ (one vector per level {Simple, Moderate, Hard}), a per-query ranking residual, a gated decoder injection, and an auxiliary classifier.

Per-query ranking delta. The vanilla `query_ranking` scores each candidate $\mathbf{x}$ by $s_{\text{base}}(\mathbf{x})$ and selects the top-K queries via `argsort+gather`. We add a FiLM-style residual:

$$s(\mathbf{x}, d) = \sigma\big(s_{\text{base}}(\mathbf{x}) + \Delta_{\text{diff}}([\mathbf{x}; \mathbf{E}_d])\big), \tag{1}$$

with zero-initialised last layer, so $s(\cdot, d) = \sigma(s_{\text{base}})$ at $t = 0$ for every d. The downstream `argsort+gather` is non-differentiable, blocking all gradient to Δ_{diff} (§6.1).

Gated decoder injection. A difficulty-aware feature is injected at the decoder-query stage as:

$$\mathbf{q}' = \mathbf{q} + \mathbf{g}(\mathbf{q}, \mathbf{E}_d) \odot \mathbf{W}_{\text{proj}}\mathbf{E}_d, \qquad \mathbf{g}(\mathbf{q}, \mathbf{E}_d) = \sigma\big(\mathbf{W}_g[\mathbf{q}; \mathbf{E}_d]\big), \tag{2}$$

with $\mathbf{W}_g$ trunc-normal-initialised (zero bias) and $\mathbf{W}_{\text{proj}} \in \mathbb{R}^{384\times384}$ zero-initialised, so $\mathbf{q}' = \mathbf{q}$ at $t = 0$ regardless of d (mathematical equivalence to vanilla at warm-start; verified empirically in §5.3 to within 0.001 CD-ℓ_2 on Simple and bit-identically on Moderate/Hard, without a bitwise audit).

Difficulty predictor. A lightweight two-layer MLP on the encoder's global feature produces a difficulty logit, trained with cross-entropy against the oracle crop-ratio-derived label. It both yields an inference-time

prediction (obviating the need for a test-time oracle) and injects a weak self-supervision signal into the encoder during training.

4.2 Training Regime and Matched Vanilla Control

All DiffCond variants are warm-started from the Baseline AdaPoinTr checkpoint (§5.1) using a builder-side remap that copies `query_ranking` weights into s_{base} and leaves every `diff_cond_*` module at construction-time initialisation; the model is then fine-tuned for 30 epochs (PoinTr-repository `max_epoch=30` convention) with AdamW (lr 1×10^{-4}, weight decay 5×10^{-4}, batch size 48, PoinTr-repository default `LambdaLR` schedule with `lr_decay`$= 0.9$ every 21 epochs) through the shared crop pipeline (batch-level $r \sim \mathcal{U}(0.25, 0.75)$, integer-crop, FPS to 2048). The total loss is

$$\mathcal{L} = \mathcal{L}_{\text{coarse}} + \mathcal{L}_{\text{fine}} + \lambda\, \mathcal{L}_{\text{CE}}, \qquad \lambda = 0.1, \tag{3}$$

where $\mathcal{L}_{\text{coarse}}$ and $\mathcal{L}_{\text{fine}}$ are the standard Chamfer-distance losses on the coarse and dense outputs; the oracle difficulty is quantised from the crop ratio (<0.375: Simple; $[0.375, 0.625)$: Moderate; ≥ 0.625: Hard) and fed to both the ranking-delta and injection paths. At inference we report oracle-fed (**Oracle**) and predictor-fed (**Predict**).

The *matched vanilla control* (**Vanilla-FT**) is plain AdaPoinTr (no `diff_cond_*` modules) warm-started from Baseline and fine-tuned for the same 30 epochs, same optimiser, matched seed, and *through the DiffCond crop path*. Within the matched training recipe (shared warm-start, seed, schedule, optimiser, and crop pipeline), any difference between Vanilla-FT and the DiffCond candidate (**DiffCond**) is therefore attributable to the additions of §4.1 rather than to warm-start dynamics or the crop pipeline; seed-induced variance is partially controlled within each seed-matched comparison, and across-seed variance is characterised in §5.2. We do not bound the contribution of training-time non-determinism (`cudnn.benchmark=True`) under this seed budget.

5 Experiments

5.1 Setup

Dataset and metrics. We evaluate on the ShapeNet-55 test split (10,518 models across 55 categories) under the official eight-viewpoint stratified protocol: each sample is cropped at three fixed ratios (Simple: 25%, Moderate: 50%, Hard: 75%) from eight deterministic viewpoints, for a total of 24 partial inputs per test sample across the three bins (i.e., 8 per difficulty bin). We report F-Score@1%, CD-ℓ_1 and CD-ℓ_2 (the CD metrics scaled by 10^3) per bin, per candidate. Category-balanced means are computed as in AdaPoinTr.

Candidates. All three candidates share warm-start source, fine-tune schedule, seed, and crop path (see §4.2):

- **Vanilla-FT** (matched vanilla): plain AdaPoinTr, 30 epochs.
- **DiffCond Oracle**: the DiffCond model of §4, inference fed ground-truth difficulty.
- **DiffCond Predict**: same checkpoint, inference fed the predictor's argmax.

Seed selection. Seeds $\{42, 43, 44\}$ are sequential integers continuing the PoinTr-repository default seed 42; the cross-backbone replication in §5.7 adds seed 45 under the pre-specified one-step stop-rule documented in the supplement.

Baseline definition and caveat. We refer to our vanilla AdaPoinTr reference checkpoint as **Baseline**: 300-epoch training from scratch on ShapeNet-55 under the PoinTr-repository default configuration (batch size 48, LR decay 0.9 every 21 epochs, 512 queries). This differs from the published AdaPoinTr configuration (batch size 64, LR decay 0.76 every 20 epochs, 256 queries) and reaches category-mean CD-ℓ_2 $= 0.869$ versus the published 0.81. All comparisons below are *internal* to this re-implementation; they do not claim

state-of-the-art scores, only a matched-control contrast. Because each comparison is internal and matched in this way, absolute baseline strength is not a confound for the within-benchmark contrast—the very property that cross-paper comparisons lack (Henderson et al., 2018; Musgrave et al., 2020). Whether matched-control conclusions transfer to a stronger warm-start (matching the published 0.81 mean CD-ℓ_2) is a natural next step; our framing implicitly assumes that the $\sim 7\%$ warm-start gap is not load-bearing for the matched-control sign-flip—an assumption our data cannot directly verify.

5.2 Main Result: Matched-Control Comparison

Table 1 reports the full eight-viewpoint S/M/H evaluation for Vanilla-FT versus DiffCond under both inference settings. The pre-specified decision rule from §3.4 is applied directly.

Table 1: Matched-control comparison on ShapeNet-55 (official 8-viewpoint S/M/H protocol). CD-$\ell_2 \times 10^3$, lower is better; best in bold. All candidates share warm-start, 30-epoch schedule, seed set $\{42, 43, 44\}$, and crop pipeline; Δ is the sign-aware improvement $(V-D)/V$ on the raw 3-seed aggregate (so positive Δ means DiffCond improves over Vanilla-FT on the lower-is-better CD-ℓ_2 metric). Table values are rounded to 3 decimals; small differences between Δ and back-computation from rounded entries are expected. Across-seed paired-Δ standard deviations are 1.60pp (Simple), 1.26pp (Moderate), 0.89pp (Hard), and 0.54pp (Mean); under Student's $t_{0.975} = 4.30$ at df $= 2$ the corresponding 95% CIs are $[-8.16, -0.20]$% (Simple), $[-6.22, +0.04]$% (Moderate), $[-7.69, -3.27]$% (Hard), and $[-5.89, -3.23]$% (Mean). The Hard CI upper bound (-3.27%) clears the pre-specified -3% rejection threshold (§3.4); the Simple, Hard, and Mean intervals exclude zero, while Moderate marginally crosses zero. DiffCond Predict values deferred to supplementary (tracks Oracle within 0.004 CD-ℓ_2 per bin across all 9 seed$\times$bin pairs).

Mode	Vanilla-FT	DiffCond Oracle	Δ
Simple	$\mathbf{0.574 \pm 0.007}$	0.598 ± 0.010	-4.18%
Moderate	$\mathbf{0.820 \pm 0.015}$	0.846 ± 0.009	-3.09%
Hard	$\mathbf{1.552 \pm 0.040}$	1.637 ± 0.051	-5.48%
Mean	$\mathbf{0.982 \pm 0.021}$	1.027 ± 0.022	-4.56%

Result. Across every bin and every seed, DiffCond underperforms Vanilla-FT on CD-ℓ_2. Mean CD-ℓ_2 worsens by 4.56% and Hard by 5.48%, both clearing our pre-specified negative threshold. The same DiffCond checkpoint, evaluated instead against a *from-scratch* 30-epoch vanilla baseline as in the warm-start pitfall of §3.2, would have shown apparent *gains* of roughly +14%–21% across all bins. On identical trained weights, swapping the from-scratch baseline for a matched warm-start one flips the apparent direction of the comparison: roughly +20% becomes ~ -5%. This sign-flip—a swing of nearly 25pp on the same checkpoint—is the most direct evidence that matched-control reporting materially affects the conclusion drawn from the experiment, and it motivates the four mechanistic probes of §6.

5.3 Epoch-0 Equivalence Probe

To rule out an initialisation-time source, we construct a DiffCond epoch-0 checkpoint by applying the builder's warm-start remap (`query_ranking`$\rightarrow s_{\text{base}}$, all `diff_cond_*` modules zero-initialised) without any gradient step. On the official S/M/H protocol this untrained checkpoint produces CD-ℓ_2 0.519/0.734/1.352 (both Oracle and Predict), matching the Baseline reference (0.520/0.734/1.352) to within 0.19% on Simple and bit-identically on Moderate and Hard.

Oracle and Predict are identical at ep 0 because $\mathbf{W}_{\text{proj}} = \mathbf{0}$ makes $\mathbf{q}' = \mathbf{q}$ for every d: DiffCond is therefore *mathematically* equivalent to vanilla at initialisation, and the small residual 0.001 CD-ℓ_2 drift on Simple (vs. bit-identical Moderate and Hard) is consistent with floating-point ordering in the additional zero-initialised layers rather than with a genuine init disparity, though we do not provide a bitwise audit. Combined with Table 1, this yields a strong conclusion: the observed Hard regression is a training-time effect, not a remap bug, not a warm-start transcription error, and not an init-disadvantage. It constitutes a robust init-effect rule-out within our setup, and it localises diagnosis to training dynamics (§5.5) and mechanism (§6).

Table 2: Ablation matrix. "Ranking cond.": whether the difficulty enters the `diff_delta` ranking path (✓) or uses the vanilla s_{base} only (✗). "CE λ": auxiliary cross-entropy weight. All four candidates warm-start from Baseline and fine-tune for 30 epochs with seed 42 through the DiffCond crop path. CD-ℓ_2 $\times 10^3$ on the official S/M/H protocol. Single-seed ablation; seed-stability of the 2×2 matrix was not assessed.

Candidate	Rank. cond.	CE λ	Simple	Moderate	Hard	Mean	ΔMean
Vanilla-FT (vanilla)	✗	0	**0.568**	**0.810**	**1.529**	**0.969**	0.00%
DiffCond v1 (full)	✓	0.1	0.601	0.836	1.619	1.019	−5.13%
v1-λ0 (CE off, rank on)	✓	0	0.573	0.823	1.580	0.992	−2.37%
v2b (CE off, rank off)	✗	0	0.596	0.854	1.550	1.000	−3.20%

5.4 Test-time Ensembling Does Not Rescue on the Hard-bin Probe

Is the regression a *wrong-conditioning* problem that test-time routing can repair? We evaluate seven decision-time routing (DTR) strategies on a Hard-bin probe (100 samples ×2 viewpoints) using the trained DiffCond checkpoint. The six non-oracle strategies cluster within 1.4% of `cond_hard` (CD-ℓ_2 values 1.662–1.696, with `cond_hard` itself at 1.673); the per-sample upper bound `oracle_best_cdl2` reaches 1.571, a 6.10% gain over `cond_hard` yet still 9.86% worse than Vanilla-FT (ep-30) on the identical subset (CD-ℓ_2 1.430). Even this theoretical upper bound of discrete-conditioning routing is 9.86% worse than Vanilla-FT, far below the +1.5% method-bar against the matched vanilla; the residual gap is larger than in the full-test comparison, providing consistent evidence that, under the tested discrete conditionings and ensemble strategies on this probe, DiffCond does not recover the representation quality Vanilla-FT exhibits. We treat these subset gaps as descriptive of the 100×2 Hard probe rather than as inferential point estimates; per-sample artefacts are bundled but bootstrap CIs across samples were not pre-specified for this probe.

5.5 Targeted Ablations and Training-loss Artefact

Before isolating components, we resolve a potentially misleading training-time observation. Per-epoch training logs of DiffCond vs. Vanilla-FT show an apparent ∼2.8× gap in the dense Chamfer training loss (henceforth "DenseLoss") at epoch 0 (118.0 vs. 42.6, a difference of ~ 75 in DenseLoss units). This is a logger accounting artefact rather than training instability: the runner folds the auxiliary CE term from Eq. (3) (with weight λ), which contributes $\lambda \cdot H_{\text{prior}} \cdot 10^3$ in DenseLoss units at random initialisation (where H_{prior} is the entropy of the difficulty-label prior). With $\lambda = 0.1$ and a near-uniform 3-class prior, this contribution is on the order of ~ 100, which dominates the observed ~ 75 gap; the residual trace differences originate in the zero-initialised DiffCond modules at $t = 0$. We flag this explicitly to forestall misinterpretation of the raw training curves: the apparent training-time gap is bookkeeping, not regression.

To isolate which DiffCond component drives the regression we run two targeted ablations: (i) **v1-λ0** disables only the CE loss but keeps the ranking conditioning active, and (ii) **v2b** additionally disables the ranking-path conditioning (reverting to s_{base}). Combined with Vanilla-FT (both off) and DiffCond (both on), this yields the 2×2 matrix in Table 2.

Removing CE (DiffCond→v1-λ0) recovers +2.76pp of the Mean regression—an *exploratory single-seed* attribution of CE as the largest identified driver, broadly consistent with the static audit (§6.1). Further removing the ranking path (v1-λ0 →v2b) changes Mean only marginally ($\Delta = -0.83$pp) and flips per-bin sign, consistent with the ranking path being architecturally inert. The residual −3.20% v2b Mean regression is attributable to the remaining additions (embedding and gated injection). The 2×2 matrix is, however, single-seed (seed 42); we therefore explicitly frame the CE driver attribution as an exploratory mechanism hypothesis, not a settled causal decomposition, and flag a multi-seed replication of this ablation as the most informative follow-up. We note one cross-row tension: v1-λ0 and v2b differ by roughly 4pp on Simple and Moderate (with Hard mildly reversed) despite the ranking-delta path being architecturally dead in v1-λ0. We attribute this swing to seed-level cascade effects under single-seed ablation but cannot rule out a forward-path interaction that the static audit (§6.1) does not detect.

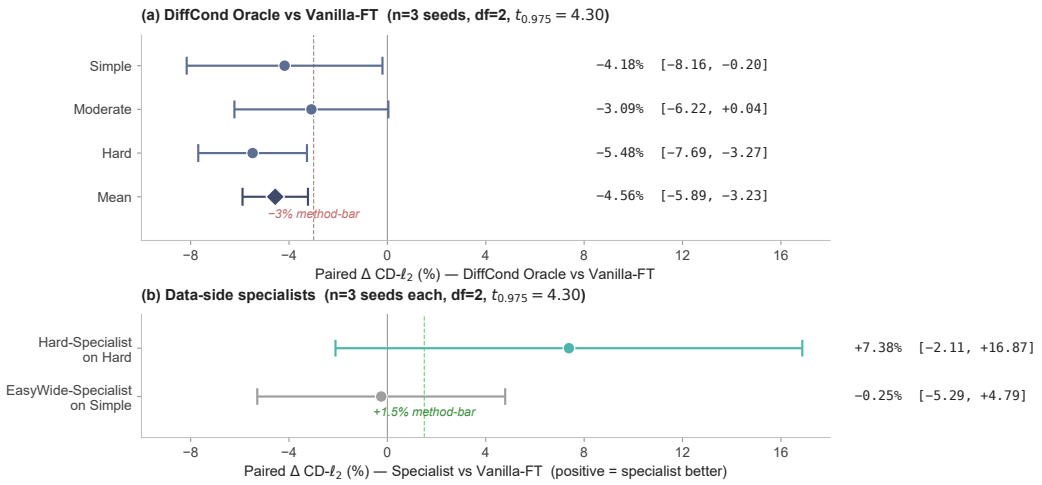

Figure 3: Forest plot of 3-seed matched-control deltas. Each row: dot is the mean paired $\Delta\%$, horizontal whisker is the 95% CI (df $= 2$, $t_{0.975} = 4.30$); vertical solid line marks 0%; vertical dashed lines mark the pre-specified method-bar thresholds. **(a)** DiffCond Oracle vs Vanilla-FT regresses on every bin and clears the -3% REVERSE-SEVERE threshold on Hard and Mean (both CIs lie entirely beyond -3%). **(b)** The Hard-Specialist paired mean ($+7.38\%$ on Hard) is above the $+1.5\%$ method-bar, although the 95% CI $[-2.11\%, +16.87\%]$ is wide and crosses zero; EasyWide-Specialist shows no Simple-bin gain (-0.25%, null). Descriptive figure; no general law of specialisation is claimed.

5.6 Data-Side Counterexample: Asymmetric Benefit of Target-Range Specialisation

The result so far is one-sided: *architectural* difficulty conditioning fails under matched control. A natural follow-up question is whether the failure is *about difficulty conditioning itself*, or specifically about *architectural* conditioning. We test this by asking the simplest possible alternative: can a purely *data-side* intervention—no new modules, no new loss term, only a change in the training crop distribution—recover a Hard-bin gain? The answer turns out to be yes, but only on the Hard bin, which makes the result a constructive counterpoint rather than a refutation of the main finding. We fine-tune a vanilla AdaPoinTr under the identical Vanilla-FT recipe, except that the crop ratio is drawn from $\mathcal{U}(0.625, 0.75)$ (the Hard bin only); we denote this candidate **Hard-Specialist**. Across three seeds ($\{42, 43, 44\}$), Hard-Specialist reaches Hard CD-ℓ_2 of 1.437 ± 0.038 versus the matched vanilla baseline's 1.552 ± 0.040; the per-seed paired deltas ($+8.04\%$, $+10.83\%$, $+3.27\%$; mean $+7.38\% \pm 3.82$pp) are all positive and the point estimate clears the $+1.5\%$ method-bar, although the 95% CI (df $= 2$, $t_{0.975} = 4.30$) is $[-2.11\%, +16.87\%]$ and crosses zero. We therefore treat this as an *exploratory descriptive signal* consistent with the data rather than as a formal statistical rejection of the null. The Simple and Moderate bins regress substantially (0.772 ± 0.030 and 0.920 ± 0.041; not shown as improvements) because those samples fall outside the Hard-only training crop range—an expected specialisation cost.

If target-range specialisation were the operative mechanism, an analogous Easy-side intervention should also recover a Simple-bin gain. We test this with **EasyWide-Specialist**, trained on the widened range $\mathcal{U}(0.25, 0.5)$—a broader Easy-side probe than the strict mirror $\mathcal{U}(0.25, 0.375)$ of Hard-Specialist's upper 0.125 slice, which we did not extend to a 3-seed submission run. Across the same three seeds, EasyWide-Specialist reaches Simple-bin CD-ℓ_2 of 0.575 ± 0.006 versus matched-vanilla's 0.574 ± 0.007 (per-seed paired deltas -1.06%, $+2.06\%$, -1.75%; mean $-0.25\% \pm 2.03$pp), well below the $+1.5\%$ method-bar; Moderate and Hard regress as the expected specialisation cost. The juxtaposition (Fig. 3) of Hard-Specialist ($+7.38\%$ on Hard, all three seeds positive, 95% CI crosses zero) and EasyWide-Specialist ($\sim 0\%$ on Simple, three seeds spanning both signs) is consistent with the reading that target-range fine-tuning helps where the baseline has slack (Hard) but not where it is already close to ceiling (Simple). Because the Easy-side probe is broader than a strict mirror, the contrast is best read as a Hard-side positive point estimate paired with an Easy-side

null on AdaPoinTr/ShapeNet-55, not as a general principle. A strict-mirror test together with a per-bin specialist router remain natural follow-ups.

5.7 Cross-Backbone Replication: UpTrans-Native DiffCond

Why this section. Every result so far is specific to AdaPoinTr (Yu et al., 2023), which raises an obvious objection: perhaps the failure is an idiosyncrasy of that one decoder, and a different transformer head would behave differently. We therefore replicate the experiment on **UpTrans**, the upsampling-transformer decoder used by ODGNet (Cai et al., 2024) and shipped as an alternative head in the PoinTr code base, under the same matched-control discipline: same warm-start source, same 30-epoch schedule, same crop pipeline, same seed set $\{42, 43, 44, 45\}$, same official 8-view S/M/H protocol. We chose UpTrans over alternatives such as SeedFormer or SnowflakeNet because the matched-control discipline requires byte-identical pre-training and crop pipeline, and UpTrans is the only published alternative that meets this constraint within the PoinTr code base. The aim is not to establish a universal claim about difficulty conditioning—two backbones still constitute a small sample—but to test whether the AdaPoinTr verdict is at least *not* an isolated artefact of one architecture.

UpTrans-native DiffCond as a role-level analogue. UpTrans's decoder design differs from AdaPoinTr in three relevant respects, so the difficulty-conditioning primitives must be re-instantiated: (i) the per-query ranking-delta path uses a soft dictionary attention rather than AdaPoinTr's hard `argsort+gather`; (ii) the conditioning injection uses FiLM-style scale–shift modulation in place of AdaPoinTr's gated additive query injection; and (iii) UpTrans does not expose AdaPoinTr's `predict` inference mode—its predictor branch is integrated into the conditioning, so we report Oracle inference only. These are role-level analogues of the same four primitives (difficulty embedding, ranking-delta, gated/FiLM injection, auxiliary classifier). The cross-backbone result therefore tests whether a *conceptually equivalent* naïve DiffCond extension produces a similar large gain on a different transformer architecture, rather than whether the AdaPoinTr-specific implementation ports verbatim. Specifically, the cross-backbone replication is informative about whether the original double-digit AdaPoinTr signature re-emerges on a different architecture under matched control; it does *not* test whether the AdaPoinTr-specific dead-branch failure (§6.1) is causally responsible for the AdaPoinTr matched-control regression, because UpTrans's soft dictionary attention replaces precisely the non-differentiable `argsort+gather` path that is the locus of the architectural dead-ness on AdaPoinTr.

Result. Across four seeds on the official S/M/H protocol, the UpTrans-native DiffCond does not meet the pre-specified +3% method-bar on any (mode, metric) cell, and we therefore find no support for a large architecture-general effect under this protocol. Per-seed paired deltas (DiffCond–Vanilla-FT, sign-aware: lower is better for CD; higher for F-Score) are reported in Table 3 for the focal metric CD-ℓ_2; full F-Score / CD-ℓ_1 / CD-ℓ_2 per seed appear in the supplementary (cross-backbone full per-seed table).

Table 3: UpTrans-native DiffCond vs. matched Vanilla-FT on ShapeNet-55 S/M/H, paired per-seed CD-ℓ_2 (lower is better; $\times 10^3$). Per-seed paired % improvement is computed seed-by-seed ((Van−Diff)/Van×100%); mean / std / 95% CI are taken across the four seeds via Student's t at df = 3, $t_{0.975} = 3.18$. "Beats +3%?" tests whether the CI lower bound clears the pre-specified method-bar of §3.4. Per-seed values are released with the submission bundle; paired statistics use `tools/paired_stats.py`.

Mode	Per-seed paired %	Mean paired %	95% CI	Beats +3%?
Simple	$\{+1.41,\ -1.64,\ +1.38,\ -0.39\}$	+0.19%	$[-2.17,\ +2.55]$%	no
Moderate	$\{+2.30,\ -0.69,\ +0.68,\ -0.95\}$	+0.33%	$[-2.04,\ +2.71]$%	no
Hard	$\{+0.33,\ +5.87,\ +1.41,\ -0.48\}$	+1.79%	$[-2.72,\ +6.29]$%	no

Statistical reading. The largest observed paired mean is +1.79% on Hard CD-ℓ_2, with a 95% CI $[-2.72\%, +6.29\%]$ that crosses zero and falls below the pre-specified +3% method-bar lower bound. The Hard improvement is driven primarily by a single seed (`seed 43` contributes +5.87% while seeds 42, 44, and 45 contribute only +0.33%, +1.41%, and −0.48% respectively), inflating the across-seed variance. F-Score

deltas across all three modes are similarly statistically null (all four-seed means lie between -0.12% and -0.11%, all CIs span zero; full table in supplementary). We therefore characterise the cross-backbone result as: naïve DiffCond on UpTrans *does not reproduce a large architecture-general gain*, and on the most discriminative bin (Hard) the largest observed signal is small relative to seed variance and below the pre-specified acceptance threshold. We deliberately stop short of stronger claims; the data are equally consistent with (a) a small genuine improvement that would require many more seeds to detect, and (b) no architecture-general effect at all. Quantitatively, the Hard 95% CI half-width is approximately ± 4.5pp at $n = 4$ (pooled SD $\approx$ 2.83pp, $t_{0.975} = 3.18$ at df $= 3$), so a true effect below the pre-specified $+3\%$ method-bar is not statistically distinguishable from null under our sample size; only a substantially larger seed set could rule out such a small effect (the supplement reports the retrospective minimum-detectable effect). Requiring a candidate to clear the bar on every relevant (mode, metric) cell, rather than a single favourable one, is the conjunctive criterion endorsed for comparisons over few cells (Bouthillier et al., 2021); the breadth of the null across all nine cells therefore reflects a principled test rather than accumulated noise.

Contrast with the AdaPoinTr double-digit inflation pattern. The original AdaPoinTr inflation pattern attributed to naïve DiffCond (under the *warm-start inflation* configuration of §3.2) was a near-uniform $+14\%$–21% across all three difficulty bins, matched in magnitude across both Oracle and Predict inference. The UpTrans replication (matched-control configuration, same seed set) shows neither uniformity across bins (-0.48% to $+5.87\%$ per-seed range on Hard) nor any single bin near that magnitude. Together with the AdaPoinTr matched-control regression (§5.2), this is consistent with two compatible interpretations: the AdaPoinTr double-digit inflation was either an evaluation-discipline artefact of the warm-start pitfall (§3.2) or specific to AdaPoinTr's particular decoder design (or both); the UpTrans data alone cannot decisively separate the two. What it does show is that the *architecture-general* reading—"naïve difficulty conditioning produces $\sim +20\%$ Hard-bin gains across transformer completion backbones"—is not supported.

PCN sanity check (supplementary). As a secondary cross-benchmark sanity check, we also run a 2-seed UpTrans Vanilla / DiffCond pair on PCN (Yuan et al., 2018) with a nearest-neighbour difficulty proxy (PCN does not provide native S/M/H labels; the proxy is computed once and archived under `data/PCN/difficulty_proxy.json`). The 2-seed mean deltas on the official PCN test split are $+0.06\%$ F-Score, $+0.3\%$ CD-ℓ_1, $+1.1\%$ CD-ℓ_2, all well below the $+3\%$ method-bar. We position this as a supplementary sanity check rather than core cross-backbone evidence: the proxy difficulty has no native ground-truth, only 2 seeds were run, and PCN's category distribution differs substantially from ShapeNet-55. The full per-seed PCN table appears in the supplementary (PCN sanity check section).

6 Mechanistic Diagnostics

Matched-control comparison tells us *that* DiffCond regresses; it does not tell us *why*. We therefore probe the failure from four independent angles, each designed to rule in or out a specific mechanism: a static weight audit (§6.1) checks whether the conditioning modules are even active at convergence; an epoch-0 official-eval probe (§5.3) checks whether DiffCond starts disadvantaged before any training occurs; a CE-accounting analysis (§5.5) checks whether the apparent training-loss gap is a real reconstruction-loss gap or a logger artefact; and a decision-time routing test (§5.4) checks whether the regression can be rescued at inference time without re-training. Read together, the four probes converge on one established architectural defect (a dead ranking-delta branch) and one best-supported training contributor (the auxiliary cross-entropy loss) under the current single-seed ablation evidence.

6.1 Static Weight Audit of the Trained Checkpoint

We directly inspect the DiffCond ep-30 checkpoint. Table 4 reports summary statistics for every DiffCond-specific module as well as aggregate drift from Baseline on the shared backbone.

Finding 1: ranking-delta branch dead by construction. The residual Δ_{diff} from Eq. (1): its last-layer weight and bias remain identically zero after 30 epochs—an architectural rather than an optimisation failure: the downstream `argsort`+`gather` is non-differentiable with respect to the scores and produces no gradient

Table 4: Static audit of the DiffCond ep-30 checkpoint. Drifts are Frobenius-norm distances to the corresponding Baseline parameters. Shared backbone drift is computed over every parameter present in both Baseline and Vanilla-FT/DiffCond.

Module / quantity	Value	Interpretation
`diff_delta[-1].{weight,bias}`	exactly 0	Branch dead
`diff_cond_proj.weight` ($\|\cdot\|_F$, $\|\cdot\|_\infty$)	4.61, 0.065	Learned, small
`diff_cond_gate.weight` norm	11.81	Trained
Embedding norms (S, M, H)	0.67, 0.21, 0.98	Distinct
Embedding cos (S-H, S-M, M-H)	−0.95, +0.57, −0.74	Antipodal S/H
Shared-backbone drift DiffCond / Vanilla-FT	0.1065 / 0.1065	Identical overall
`base_model` core drift DiffCond / Vanilla-FT	23.8% / 21.6%	+2.2pp in DiffCond

to Δ_{diff} (Fig. 2). The pattern generalises: any learned conditioning residual placed inside a discrete-selection operation will exhibit the same behaviour in the absence of a soft surrogate. The architectural deadness conclusion formally holds at 30 epochs; while we cannot rule out a saddle-point regime that a longer schedule might escape, the underlying non-differentiability of `argsort`+`gather` is structural rather than schedule-dependent, so additional epochs alone would not be expected to introduce gradient flow without a soft surrogate.

Finding 2: injection active but small; predictor near-perfect. For the gated injection (Eq. (2)), embeddings are clearly separated (Simple–Hard cos −0.95, Simple–Moderate +0.57); $\mathbf{W}_{\text{proj}}$ has norm 4.61 and $\ell_\infty = 0.065$ (non-zero, rules out gate-off); gate weight norm is 11.81 with near-zero bias. The injection path is active but small. A direct calibration probe shows the predictor is near-perfect on this probe: 99.88% top-1 over 12,000 predictions, Brier 0.0021, ECE 0.0051. We do not directly probe which feature the predictor uses, but the near-perfect accuracy is consistent with exploitation of low-level geometric cues of the cropped input (e.g., the spatial extent / compactness of the remaining cluster, which differs systematically across the three crop ratios and survives the FPS resampling to 2048 points), rather than with a generalisable difficulty estimator. A direct geometric-cue ablation is left to future work; in either case the official S/M/H bins are defined by deterministic crop ratios, so the post-FPS geometric footprint carries enough bin information for a head with access to the encoder global feature to be near-perfect. Together these resolve the Oracle≈Predict closeness in Table 1: Predict tracks Oracle because the predictor almost always picks the correct label, not because the model ignored the difficulty signal. The regression therefore occurred while the model received correct difficulty information, narrowing failure attribution to the training dynamics of the CE head and the injection path rather than to a labelling or routing issue.

Finding 3: shared transformer trunk drifts more under DiffCond. Shared-backbone drift is identical between DiffCond and Vanilla-FT (0.1065), but the `base_model` core (encoder + decoder transformer trunk, 30M shared parameters) shows +2.2 pp more drift in DiffCond, consistent with the extra DiffCond-exclusive gradient pressure (auxiliary CE loss on the encoder global feature plus the active gated injection into the decoder queries). The $\lambda = 0$ ablation (§5.5) is the most direct controlled evidence pointing to CE as the dominant contributor; we flag its single-seed limitation.

6.2 Takeaways

Three lessons for future conditioning-style extensions: (T1) a learned residual placed inside a discrete-selection operation (`argsort`+`gather`) is architecturally dead and requires a soft relaxation (Gumbel-top-K, temperature softmax) or a post-selection residual; (T2) an auxiliary classification loss sharing features with the primary reconstruction objective must be validated against a $\lambda = 0$ ablation; (T3) stratified per-bin reporting exposes profile information that averaged reporting can hide at no extra evaluation cost—v2b's −3.20% Mean sits atop a −1.37% Hard, −5.43% Moderate, and −4.93% Simple profile that the mean obscures.

7 Discussion, Limitations, and Future Work

7.1 Scope and Limitations

Our claims are deliberately local. We do *not* assert that difficulty conditioning can never succeed—soft-differentiable selection, a post-selection residual, and a smaller CE weight λ are plausible recovery paths that our data do not rule out. Three practical bounds apply: (i) three seeds for the matched-control, Hard-Specialist, and EasyWide contrasts, with single-seed 2×2 ablations; (ii) a single fine-tuning schedule (30 epochs), a single backbone (AdaPoinTr under the PoinTr-repository default configuration), and a single dataset (ShapeNet-55); (iii) a Baseline approximately 7% weaker than the paper-reported AdaPoinTr due to the repository default configuration, with `cudnn.benchmark=True` active during training, so strict cross-hardware reproduction is not guaranteed. Whether matched-control conclusions transfer to a stronger warm-start matching the published 0.81 mean CD-ℓ_2 is the natural follow-up; our discipline-artefact framing implicitly assumes that the $\sim$7% warm-start gap is not load-bearing for the matched-control sign-flip (§5.1), an assumption our data cannot directly verify. Beyond the bounded UpTrans cross-backbone replication and 2-seed PCN sanity check shown in §5.7, broader backbone and dataset coverage is the natural next step.

Acknowledged design sensitivities. The DiffCond construction combines four primitives that each have known sensitivities: (i) the `argsort`+`gather` ranking-delta path is non-differentiable, a well-documented failure mode for gradient flow through discrete operations (Kool et al., 2019); (ii) an auxiliary classification loss sharing encoder features with the primary regression task can compete for capacity; (iii) zero-initialised injection paths must overcome warm-start inertia within a finite fine-tuning budget; and (iv) FiLM-style modulation conditioned on a 3-class discrete signal can remain inertial under matched initialisation. The observed -5% regression is therefore consistent with the joint effect of these sensitivities under the chosen 30-epoch budget. We frame our headline claim accordingly: *this specific naïve DiffCond on AdaPoinTr/ShapeNet-55 fails under matched control*. A Gumbel-soft variant, a smaller CE weight λ, or a from-scratch schedule are plausible recovery paths and constitute the natural follow-up tests. Concretely, we would regard a difficulty-conditioning claim as credible if, under matched warm-start and a shared crop pipeline, it cleared the +3% Hard-bin bar across ≥ 5 seeds with a 95% CI excluding zero and reproduced on a second backbone.

7.2 Candidate Mechanisms Behind the Hard-Specialist Gain

Our data do not isolate a single mechanism. The most parsimonious interpretation is *optimisation headroom*: the Baseline reaches CD-ℓ_2 of 0.520 on Simple versus 1.352 on Hard, so the warm-start has little remaining slack on Simple and substantial slack on Hard. A non-exclusive second interpretation is *information-content asymmetry*: sparser inputs admit a wider completion space, favouring a specialist prior. A from-scratch control and per-bin intrinsic-variance measurement are left to future work.

7.3 Implications and Adoption

The five-principle protocol (§3.4) is not specific to ShapeNet-55 or to point-cloud completion. Its three ingredients—matched warm-start, shared data pipeline, and pre-specified per-bin thresholds—we conjecture extend to other benchmarks with stratified difficulty (e.g., PCN with proxy difficulty, KITTI with viewpoint-derived bins, or non-completion benchmarks such as ImageNet-C corruption tiers and stratified visual question answering), though we do not test this directly. Of the three pitfalls we identified, only Pitfall 2 (warm-start inflation) is fully benchmark-agnostic; Pitfalls 1 (random-crop proxy bias) and 3 (crop-path mismatch) are point-cloud-pipeline specifics whose analogues elsewhere are not guaranteed. Adopting the discipline requires no additional evaluation pass when the matched baseline has already been run: matched-control reuses the same training and evaluation budget, only constraining *which baseline* the candidate is compared against and *how* per-bin metrics are reported. Concretely, we suggest three groups adopt the discipline by default: authors of completion follow-ups, who can use the protocol to audit their own proposed extensions before submission; benchmark maintainers, who can require matched-control reporting in submission templates; and reviewers, who can verify that headline numbers reflect the same warm-start and pipeline as their declared baseline.

Recommendations. Distilled to a checklist, a within-benchmark completion comparison should: (1) warm-start every candidate from the same checkpoint, schedule, and seed; (2) share a single crop pipeline across candidates; (3) pre-specify GO/NO-GO/REVERSE thresholds before the final run; (4) report each difficulty bin separately rather than an average alone; and (5) tabulate the full metric set (F-Score@1%, CD-ℓ_1, CD-ℓ_2). None costs additional GPU time once the matched baseline has been trained.

7.4 Future Directions

Natural follow-ups include replacing `argsort`+`gather` with Gumbel-top-K (Kool et al., 2019) or a temperature softmax, extending the PCN sanity check (§5.7) to a multi-seed study with refined difficulty bins, replicating on Projected-ShapeNet, and scaling the data-side study into a per-bin specialist router.

8 Conclusion

We identified three commonly overlooked confounds in point-cloud completion reporting (Fig. 1) and proposed a *stratified matched-control protocol* that neutralises them. Applied to a natural difficulty-conditioned AdaPoinTr extension (DiffCond), the protocol turns an apparent +14%–21% per-bin gain into a 5.48% Hard CD-ℓ_2 regression across three seeds; a cross-backbone replication on UpTrans yields no (mode, metric) cell clearing the pre-specified +3% method-bar. Four diagnostic probes localise the AdaPoinTr failure to a dead ranking-delta branch and an auxiliary cross-entropy loss; a data-side Hard-Specialist intervention produces a positive Hard-bin result while a broader Easy-side probe does not, and a 2-seed PCN sanity check finds no large effect.

Read together, the AdaPoinTr matched-control regression and the UpTrans null recast the original double-digit per-bin signature as an evaluation-discipline artefact—warm-start mismatch under uncontrolled training—rather than a transferable architectural gain. The broader lesson is that within-benchmark method comparisons on ShapeNet-55 require matched warm-start and matched data pipelines, and that a transparently reported negative audit contributes methodological evidence even when no positive method emerges. We expect the protocol to generalise to other stratified benchmarks where training-time and pipeline confounds can inflate averaged reporting; broader backbone and dataset coverage is the natural next step.

The anonymised reproducibility bundle accompanies this submission; full raw checkpoints and training logs will be released upon acceptance.

Reproducibility Statement

The bundle provides *partial* reproducibility: every numerical claim in the main tables and figures is traceable to a released aggregate JSON/TSV artefact, and the full patched training pipeline (see below) reproduces the matched-control comparison end-to-end from a public PoinTr clone. Raw fine-tuned checkpoints and per-batch training logs are *not* included for size/anonymity reasons but will be released after acceptance. To support reproducibility of the headline numerical claims in the paper, the anonymised submission bundle releases: (i) the diagnostic and aggregation scripts used to compute the headline numbers (`tools/aggregate_results.py`, `tools/paired_stats.py`, `tools/compute_pcn_proxy_difficulty.py`, `tools/ckpt_static_audit.py`, `tools/eval_dtr.py`, `tools/eval_vanilla_subset.py`, `tools/eval_predictor_calibration.py`), together with a top-level `REPRO.md`; (ii) per-seed result artefacts for the matched-control comparison on ShapeNet-55 (DiffCond Oracle, Vanilla-FT, DiffCond Predict; seeds 42, 43, 44) and the cross-backbone UpTrans-native DiffCond per-seed table (4 seeds); (iii) all checkpoint configurations, including the architectural modifications used by the DiffCond variants and the matched vanilla controls; (iv) the PCN difficulty-proxy preprocessing artefact and its construction script. Additionally, the bundle releases the aggregate from-scratch 30-epoch AdaPoinTr baseline rerun artefact for the §3.2 Pitfall 2 audit (`data/results/from_scratch_baseline_smh.json`, summarising per-bin paired deltas and `cudnn` drift relative to the pilot run), providing aggregate-level reproducibility of the warm-start comparison, which remains positive and double-digit across S/M/H bins under matched-configuration replay. The corresponding raw

fine-tuned checkpoint (`ckpt-epoch-030.pth`, ≈390 MB) and per-batch training/eval logs are kept out of the anonymised bundle for size reasons and will be released together with the other raw artefacts upon acceptance. The full PoinTr training framework that these scripts assume is the publicly available upstream repository (`github.com/yuxumin/PoinTr`, commit `3f98676aeaabd9c1150dd04cd0a51f504949ab9c`); the bundle ships, on top of that commit, the patched training pipeline used to produce the matched-control runs — `tools/runner.py` (DiffCond training/validation hooks, aligned per-difficulty validation, shared DiffCond crop pipeline), `tools/builder.py` (warm-start remap `query_ranking`→ s_{base} with zero-init of the DiffCond-specific layers), the dataset files in `datasets/`, and all configurations and modified model files needed to reproduce the candidate runs; the example training command is in `REPRO.md`. Each numerical claim in Tables 1, 2, 3, and Figure 3 is traceable to a JSON or TSV artefact in the bundle, with the producing script named in the table caption or in the supplement. Raw fine-tuned checkpoints and training logs will be released upon acceptance.

References

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML*, pp. 41–48, 2009. doi: 10.1145/1553374.1553380.

Xavier Bouthillier, Pierre Delaunay, Mirko Bronzi, Assya Trofimov, Brennan Nichyporuk, Justin Szeto, Nazanin Mohammadi Sepahvand, Edward Raff, Kanika Madan, Vikram Voleti, Samira Ebrahimi Kahou, Vincent Michalski, Tal Arbel, Chris Pal, Gaël Varoquaux, and Pascal Vincent. Accounting for variance in machine learning benchmarks. In *Proceedings of Machine Learning and Systems (MLSys)*, volume 3, 2021.

Pingping Cai, Deja Scott, Xiaoguang Li, and Song Wang. Orthogonal dictionary guided shape completion network for point cloud. In *AAAI*, volume 38, pp. 864–872, 2024. doi: 10.1609/aaai.v38i2.27845.

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. Show your work: Improved reporting of experimental results. In *EMNLP-IJCNLP*, pp. 2185–2194, 2019. doi: 10.18653/v1/D19-1224.

Guy Hacohen and Daphna Weinshall. On the power of curriculum learning in training deep networks. In *ICML*, volume 97, pp. 2535–2544, 2019.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *AAAI*, volume 32, pp. 3207–3214, 2018. doi: 10.1609/aaai.v32i1.11694.

Wouter Kool, Herke van Hoof, and Max Welling. Stochastic beams and where to find them: The Gumbel-Top-k trick for sampling sequences without replacement. In *ICML*, volume 97 of *PMLR*, pp. 3499–3508, 2019.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, pp. 2980–2988, 2017. doi: 10.1109/ICCV.2017.324.

Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. A metric learning reality check. In *ECCV*, 2020.

Stuti Pathak, Prashant Kumar, Dheeraj Baiju, Nicholus Mboga, Gunther Steenackers, and Rudi Penne. Revisiting point cloud completion: Are we ready for the real-world? In *ICCV*, pp. 25388–25398, 2025. arXiv:2411.17580.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *AAAI*, volume 32, pp. 3942–3951, 2018. doi: 10.1609/aaai.v32i1.11671.

Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Larivière, Alina Beygelzimer, Florence d'Alché-Buc, Emily Fox, and Hugo Larochelle. Improving reproducibility in machine learning research (a report from the NeurIPS 2019 reproducibility program). *Journal of Machine Learning Research*, 22(164): 1–20, 2021.

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, pp. 652–660, 2017. doi: 10.1109/CVPR.2017.16.

Paul R. Rosenbaum. *Observational Studies*. Springer Series in Statistics. Springer, 2 edition, 2002. doi: 10.1007/978-1-4757-3692-2.

Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *CVPR*, pp. 761–769, 2016. doi: 10.1109/CVPR.2016.89.

Elizabeth A. Stuart. Matching methods for causal inference: A review and a look forward. *Statistical Science*, 25(1):1–21, 2010. doi: 10.1214/09-STS313.

Lyne P. Tchapmi, Vineet Kosaraju, Hamid Rezatofighi, Ian Reid, and Silvio Savarese. TopNet: Structural point cloud decoder. In *CVPR*, pp. 383–392, 2019. doi: 10.1109/CVPR.2019.00047.

Peng Xiang, Xin Wen, Yu-Shen Liu, Yan-Pei Cao, Pengfei Wan, Wen Zheng, and Zhizhong Han. SnowflakeNet: Point cloud completion by snowflake point deconvolution with skip-transformer. In *ICCV*, pp. 5499–5509, 2021. doi: 10.1109/ICCV48922.2021.00545.

Haozhe Xie, Hongxun Yao, Shangchen Zhou, Jiageng Mao, Shengping Zhang, and Wenxiu Sun. GRNet: Gridding residual network for dense point cloud completion. In *ECCV*, pp. 365–381, 2020. doi: 10.1007/978-3-030-58545-7_21.

Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. FoldingNet: Point cloud auto-encoder via deep grid deformation. In *CVPR*, pp. 206–215, 2018. doi: 10.1109/CVPR.2018.00029.

Xumin Yu, Yongming Rao, Ziyi Wang, Zuyan Liu, Jiwen Lu, and Jie Zhou. PoinTr: Diverse point cloud completion with geometry-aware transformers. In *ICCV*, pp. 12498–12507, 2021. doi: 10.1109/ICCV48922.2021.01227.

Xumin Yu, Yongming Rao, Ziyi Wang, Jiwen Lu, and Jie Zhou. AdaPoinTr: Diverse point cloud completion with adaptive geometry-aware transformers. *IEEE TPAMI*, 45(12):14114–14130, 2023. doi: 10.1109/TPAMI.2023.3309253.

Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. PCN: Point completion network. In *3DV*, pp. 728–737, 2018. doi: 10.1109/3DV.2018.00088.

Haoran Zhou, Yun Cao, Wenqing Chu, Junwei Zhu, Tong Lu, Ying Tai, and Chengjie Wang. SeedFormer: Patch seeds based point cloud completion with upsample transformer. In *ECCV*, pp. 416–432, 2022. doi: 10.1007/978-3-031-20062-5_24.