# TETRIS: SELF-SUPERVISED TASK REPRESENTATION LEARNING FOR POINT IN-CONTEXT LEARNING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Point In-Context Learning (Point ICL) aims to infer a transformation from a support pair and apply it to a query point cloud without task-specific parameter updates. However, existing protocols train and evaluate models on the same predefined task families, allowing a *task-recognition shortcut*: models may identify and execute a familiar task rather than infer the transformation from context. Such evaluations leave transfer to unseen transformation processes unresolved. To assess such transfer, we introduce PIC-Bench, a benchmark of five structured point cloud restoration tasks modeling corruptions arising in 3D acquisition and data processing, spanning both deterministic and ambiguous transformations. PIC-Bench reveals a substantial generalization gap in existing Point ICL methods, suggesting that success on familiar tasks does not readily transfer to unseen transformation processes. To address this limitation, we propose Task rEpresenTation leaRnIng through Self-supervision (TETRIS), a self-supervised task representation learning framework requiring no training pairs from downstream tasks. TETRIS uses unlabeled clean point clouds as targets and generates corresponding sources through online compositions of parameterized atomic point-set operators. Applying the same composition across different objects encourages the model to represent transformations independently of object geometry. On PIC-Bench, TETRIS achieves the best performance across all five tasks and reduces the overall Chamfer distance from $31.6$ to $8.7$ compared with the same backbone trained under the original supervised protocol. When fine-tuned on downstream ShapeNet-In-Context tasks, TETRIS pre-training further improves data efficiency, reducing the Chamfer distance from $9.8$ to $4.9$ with only $1\%$ of the labeled data.

## 1 INTRODUCTION

In-context learning (ICL) (Brown et al., 2020) enables a model to infer a task from demonstrations and perform it without updating its parameters. This paradigm is extended to many applications for generalization such as robotics (Generalist-Team, 2026; Skild-AI, 2026). Motivated by this paradigm, Point In-Context Learning (Point ICL) extends ICL to 3D point clouds: given a support pair $(x_s, y_s)$ that demonstrates a point-set transformation and a query $x_q$, the model predicts the corresponding target $y_q$ (Fang et al., 2023; Jiang et al., 2024a;b; Shao et al., 2025; Lin et al., 2026; Jiang et al., 2026; Liu et al., 2026; Zhang et al., 2026). By expressing diverse point cloud problems through a unified input–output interface, Point ICL offers a path toward generalist 3D models that follow task demonstrations without task-specific fine-tuning. Crucially, its promise goes beyond executing several predefined tasks with a shared model: because task information is conveyed through context, the model should be able to infer transformations that were not seen during training. Realizing this promise requires a representation that captures the transformation expressed by the support pair while separating it from the geometry of the particular support instance.

Despite this promise, existing Point ICL protocols (Fang et al., 2023; Jiang et al., 2024a;b; Lin et al., 2026) evaluate models within a fixed task vocabulary. Models are trained and tested on the same predefined task families—such as reconstruction, denoising, and registration—while out-of-domain evaluations (Jiang et al., 2024a; Lin et al., 2026) change the underlying point cloud distribution but retain these task definitions. Such protocols demonstrate robustness to unseen object instances and data domains, but they do not isolate generalization to new tasks. More fundamentally, they admit a *task-recognition shortcut*: because every evaluation task belongs to a family seen during training,



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.

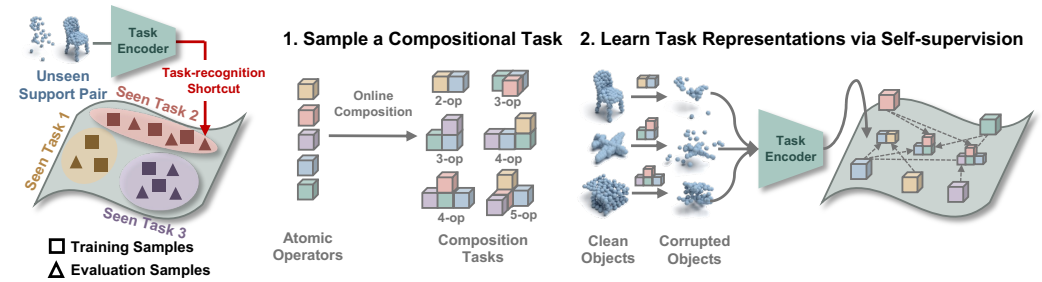



Figure 1: Existing Point ICL protocols evaluate predefined task families and can therefore admit a task-recognition shortcut. TETRIS instead constructs compositional tasks online from atomic point-set operators and learns task representations through cross-object reconstruction.

the support pair can be used to identify a familiar task and invoke the corresponding behavior learned during training, rather than to infer the transformation expressed by the context.

To evaluate whether Point ICL models can generalize to tasks governed by transformation processes unseen during training, we introduce PIC-Bench, a challenging benchmark comprising five point cloud restoration tasks—structured sensor denoising, scene clutter removal, multi-view misalignment, rendered partial completion, and physical occlusion recovery—that model structured corruptions arising during 3D acquisition and data processing. Rather than defining evaluation solely through isolated synthetic transformations, PIC-Bench models geometry-dependent processes involving sensor artifacts, scene interference, multi-view pose errors, partial visibility, and physical occlusion. These tasks retain familiar restoration objectives but realize them through more structured transformation processes, allowing us to distinguish recognition of a broad task family from transfer to a particular process. PIC-Bench spans both deterministic tasks, for which the corruption process admits a near-unique target, and ambiguous tasks, where incomplete observations permit multiple plausible reconstructions. Evaluating existing Point ICL models on PIC-Bench reveals consistently poor performance across both groups, indicating that their success on predefined tasks does not readily transfer to more complex corruption processes.

These results expose a central supervision challenge: when training pairs from downstream tasks are unavailable, how can Point ICL learn task representations that transfer beyond the predefined tasks used for training? We address this challenge with TETRIS, a Task rEpresenTation leaRnIng through Self-supervision, trained on unlabeled clean point clouds. As illustrated in Figure 1, TETRIS uses transformations as learning signals: each clean point cloud serves as its own target, while online compositions of parameterized atomic point-set operators generate the corresponding source. Applying the same sampled composition to different objects encourages the learned task representation to capture the shared transformation independently of object geometry. TETRIS thus learns from a broad, continuously generated task distribution without direct supervision from downstream tasks.

On PIC-Bench, TETRIS achieves the best performance on all five tasks, with an overall Chamfer distance of $8.7$ and consistent gains on both deterministic and ambiguous settings. The learned task representation also improves downstream adaptation: during fine-tuning on ShapeNet-In-Context, TETRIS maintains lower validation error throughout training and converges to a lower final error. Its advantage is largest when labeled data are scarce, reducing test Chamfer from $9.8$ to $4.9$ with $1\%$ of the training data, from $6.7$ to $3.3$ with $5\%$, and from $4.1$ to $2.5$ with $33\%$. Finally, t-SNE visualization shows that task embeddings group according to the applied atomic and compositional transformations across held-out objects, consistent with the embeddings retaining transformation-related information across object instances.

Our contributions are threefold:

- We highlight that existing Point ICL protocols admit a *task-recognition shortcut*: when training and evaluation share transformation processes, performance cannot distinguish task inference from recognizing and executing a familiar task.

- We introduce PIC-Bench, a benchmark comprising five deterministic and ambiguous point cloud restoration tasks defined by structured transformation processes not used for training, designed to evaluate transfer to previously unused processes.
- We propose TETRIS, which learns task representations from self-supervised atomic compositions, achieves the best performance on all five PIC-Bench tasks, and improves downstream adaptation under limited supervision.

## 2 Related Work

### 2.1 In-Context Learning

In-context learning (ICL) (Brown et al., 2020) enables pretrained models to adapt their behavior from demonstrations provided at inference time without updating model parameters. Prior work shows that demonstrations can induce compact task-related representations that modulate model predictions (Hendel et al., 2023). Such a representation, however, does not by itself reveal whether the model has inferred a new task from context or activated a behavior already acquired during pre-training. Existing analyses therefore distinguish task recognition (retrieval), which identifies a familiar task and invokes pretrained knowledge, from task learning, which infers a new input–output mapping from demonstrations (Pan et al., 2023; Lin & Lee, 2024). Controlled regression studies further show that pre-training task diversity strongly affects whether ICL remains specialized to the training task distribution or generalizes beyond it (Raventós et al., 2023; Goddard et al., 2025). This distinction also extends to visual ICL, where paired images serve as task demonstrations for a new query (Bar et al., 2022; Wang et al., 2023). Together, these findings motivate evaluating whether point ICL models infer the transformation expressed by context or merely use context to select a behavior instantiated during training.

### 2.2 Point In-Context Learning and its Generalization

Point-In-Context (PIC) (Fang et al., 2023) introduced a unified point-cloud ICL framework and the ShapeNet In-Context benchmark for reconstruction, denoising, registration, and part segmentation. PIC++ (Liu et al., 2026) extended the segmentation setting, while MICAS (Shao et al., 2025) introduced task- and query-adaptive point and prompt sampling. Subsequent work expanded Point ICL along complementary domain-generalization and adaptation axes. DG-PIC (Jiang et al., 2024a) introduced leave-one-domain-out multi-task domain generalization without parameter updates, while SADG (Jiang et al., 2026) improved structural consistency under the same protocol using a structure-aware state-space model (Gu & Dao, 2023). PCoTTA (Jiang et al., 2024b) addressed continually changing target domains through online prototype adaptation. DeformPIC (Lin et al., 2026) reformulated Point ICL as point deformation and evaluated direct transfer from ShapeNet (Chang et al., 2015) to ModelNet40 (Wu et al., 2015) and ScanObjectNN (Uy et al., 2019) without test-time adaptation. Despite these advances, existing protocols vary architectures, data domains, prompt selection, or adaptation strategies while evaluating complete transformation processes instantiated during training. PIC-Bench instead changes the task distribution by evaluating structured point-set transformation processes not instantiated as complete training tasks.

### 2.3 Self-supervised Representation Learning

Self-supervised representation learning (Chen et al., 2020; Grill et al., 2020; Caron et al., 2021; Balestriero & LeCun, 2025; Zhang et al., 2022; Lin et al., 2024; Zhang et al., 2024) has largely focused on extracting transferable data representations from unlabeled samples. Contrastive methods such as MoCo (He et al., 2020) align augmented views of the same image, encouraging invariance to the selected transformations. Masked modeling instead learns data representations by recovering missing observations: MAE (He et al., 2022) reconstructs masked image patches, whereas I-JEPA (Assran et al., 2023) predicts their latent representations without relying on hand-crafted augmentations. In 3D, Point-MAE (Pang et al., 2022) reconstructs masked local point patches to learn transferable geometric features. Despite their different objectives, these methods primarily represent the input data; transformations and masks construct the self-supervisory signal rather than constitute the representation itself. STL (Yu et al., 2024) complements this data-centric view by deriving a transformation representation from original–transformed pairs and enforcing its consistency

across different image instances. However, STL applies this representation in latent feature space to improve downstream data representations. TETRIS instead composes atomic operators online to construct a broad distribution of self-supervised tasks and learns a task representation from each support source–target pair. The inferred representation then conditions the model to execute the demonstrated transformation on a new query object.

# 3 PIC-BENCH

## 3.1 BENCHMARK DESIGN

PIC-Bench evaluates five Point ICL tasks, indexed by $c$. Each task is instantiated by a parameterized degradation process $\mathcal{D}_c$ that maps a clean point cloud to its observed input. For each episode, we sample degradation parameters $\rho$ and two clean point clouds $y_s$ and $y_q$ from different object instances. The support and query inputs are generated by

$$x_s = \mathcal{D}_c(y_s; \rho), \qquad x_q = \mathcal{D}_c(y_q; \rho), \tag{1}$$

and paired with their clean targets to form $(x_s, y_s)$ and $(x_q, y_q)$.

Support and query share the same degradation parameters $\rho$, which may specify a sensor field, a set of viewpoints and pose errors, or an occlusion configuration. The support pair therefore demonstrates how the degradation is instantiated in the current episode, and the model uses this information to recover $y_q$ from $x_q$.

## 3.2 BENCHMARK TASKS

The five tasks are grouped by the ambiguity of their recovery targets. Deterministic tasks have a near-unique clean target for each observation, whereas ambiguous tasks involve incomplete observations that admit multiple plausible reconstructions.

**Deterministic Tasks.** The deterministic tasks evaluate the correction of structured errors arising during point cloud acquisition and processing. Structured Sensor Denoising (SSD) introduces geometry-dependent sensor artifacts, including range bias, depth quantization, scanline wobble, and multipath ghost returns, and requires recovering the clean surface. Scene Clutter Removal (SCR) mixes the target object with planar surfaces, foreign object fragments, or boundary leakage, requiring the model to separate the object from its surrounding clutter. Multi-View Misalignment (MVM) fuses partial scans affected by pose errors or accumulated drift, and requires recovering a geometrically consistent point cloud.

**Ambiguous Tasks.** The ambiguous tasks evaluate shape recovery when the input omits part of the target geometry. Rendered Partial Completion (RPC) generates partial point clouds by rendering the target from one or a few viewpoints, requiring completion of the unobserved surfaces. Physical Occlusion Recovery (POR) places one or more foreground objects between the sensor and the target. The input contains visible target regions together with foreground interference; the model must remove this interference and reconstruct the hidden target geometry. RPC models missing geometry caused by limited visibility, whereas POR couples missing geometry with obstruction.

## 3.3 BENCHMARK CONSTRUCTION

We use normalized point clouds with 1,024 points from the LVIS split of OpenShape (Liu et al., 2023), with disjoint validation and test objects. Each task contains 500 validation and 2,000 test episodes, yielding 2,500 and 10,000 episodes in total, respectively. The primary protocol uses one support pair per episode. PIC-Bench contains validation and test splits only, and none of its five degradation processes is used to construct TETRIS training episodes. Each task spans four process subtypes and five difficulty levels, as detailed in Appendix A.

# 4 TETRIS

## 4.1 OVERVIEW

TETRIS learns support-conditioned point cloud restoration from unlabeled clean point clouds, without training pairs from predefined downstream tasks. It constructs training episodes online by applying the same composition of atomic point-set operators to two different objects. This exposes the model to diverse transformations without relying on a fixed vocabulary of supervised tasks.

Specifically, we sample two different clean shapes $y_s$ and $y_q$ and construct their inputs using the same sampled atomic program $a$:

$$\begin{aligned} x_s &= \mathcal{T}_a(y_s), \qquad x_q = \mathcal{T}_a(y_q), \\ z &= E_\phi(x_s, y_s), \quad \hat{y}_q = P_\theta(x_q, z). \end{aligned} \tag{2}$$

The task encoder $E_\phi$ extracts a task representation $z$ from the support pair, and the predictor $P_\theta$ uses it to recover the clean query target. The two components are trained end-to-end through query reconstruction, without task labels or training pairs from downstream tasks. We next describe how atomic compositions provide self-supervision and then present the DeformPIC$^*$ architecture that instantiates $E_\phi$ and $P_\theta$.

## 4.2 SELF-SUPERVISION THROUGH ATOMIC COMPOSITIONS

TETRIS constructs its training transformations using a library $\mathcal{A} = \{o_k\}_{k=1}^K$ of $K$ parameterized atomic point-set operators. For each episode, we sample an ordered composition of length $L \leq M$. Let $o_i \in \mathcal{A}$ and $\lambda_i$ denote the identity and parameters of its $i$-th operator. The atomic program and the resulting transformation are defined as

$$\begin{aligned} a &= \big((o_1, \lambda_1), \ldots, (o_L, \lambda_L)\big), \\ \mathcal{T}_a(y) &= \big(o_L(\,\cdot\,; \lambda_L) \circ \cdots \circ o_1(\,\cdot\,; \lambda_1)\big)(y), \end{aligned} \tag{3}$$

Operator identities, order, and parameters are resampled online for every episode. Operators with stochastic effects, such as point dropping, outlier addition, and jitter, draw their masks or noise independently each time the program is applied.

Each sampled composition is converted into episodic self-supervision by applying it to two independently sampled clean shapes. As shown in Equation 2, the support and query share the same program $a$, including its operator identities, order, and numerical parameters. Object-dependent masks, outliers, and point-wise noise are nevertheless generated independently, so the program may affect different points on the two objects while expressing the same transformation. The clean shapes $y_s$ and $y_q$ directly provide the corresponding targets, so these episodes require neither task annotations nor paired data from downstream tasks. Algorithm 1 summarizes the resulting training procedure.

**Algorithm 1** Self-supervised training of TETRIS.

```
# D: unlabeled clean point clouds, A: atomic operator library
# M: maximum composition length

for y_s, y_q in loader(D):          # independent objects
    L = randint(1, M)
    a = sample_composition(A, length=L)
    # each call draws its stochastic choices internally
    x_s = apply(a, y_s)
    x_q = apply(a, y_q)
    z = task_encoder(x_s, y_s)
    y_hat_q = predictor(x_q, z)
    loss = chamfer_l2(y_hat_q, y_q)
    optimize(loss)
```

The resulting self-supervised signal comes from transferring a transformation across objects. The task representation is extracted from the relation between $x_s$ and $y_s$, but its utility is evaluated by

reconstructing a different object $y_q$ from $x_q$. Because the object geometry and stochastic realization differ between support and query, minimizing the query reconstruction loss encourages the representation to retain the shared atomic program rather than details specific to the support example. The sampled composition is used only to generate the episode and is never provided to the model.

We jointly optimize the task encoder and predictor using the expected query reconstruction loss

$$\mathcal{L}_{\mathrm{TETRIS}}(\phi, \theta) = \mathbb{E}_{(x_s, y_s, x_q, y_q) \sim p_{\mathcal{A}}} \left[ \mathrm{CD}_{L_2}(P_\theta(x_q, E_\phi(x_s, y_s)), y_q) \right], \tag{4}$$

where $p_{\mathcal{A}}$ is the self-supervised episode distribution induced by the sampled atomic compositions. This query reconstruction objective is the sole training loss for both components.

### 4.3 Support-Conditioned Architecture

Our architecture is designed to encode the relation between the support source and target without point-wise correspondence, and to provide the query predictor with both global and token-level support information. We instantiate the task encoder $E_\phi$ and predictor $P_\theta$ by improving the DeformPIC architecture (Lin et al., 2026). The improved architecture retains the patch-based Transformer predictor of DeformPIC but redesigns how the support pair is represented and how this representation conditions query prediction. Training this architecture with the self-supervised objective in Equation 4 yields TETRIS. For controlled comparison, we also train the same architecture with the original Point ICL protocol and refer to this supervised counterpart as DeformPIC$^*$.

**Task encoder.** DeformPIC extracts support source and target patches at corresponding point indices and compresses their features into a single global condition. Our task encoder instead partitions $x_s$ and $y_s$ independently into $G$ local patches using farthest-point sampling and $k$-nearest neighbors, removing the need for point-wise correspondence between the two clouds. Let $(\mathcal{N}_{s,g}, c_{s,g})$ and $(\mathcal{N}_{t,g}, c_{t,g})$ denote the neighborhood and center of the $g$-th source and target patches, respectively. A shared point encoder $f_\phi$, a center-position encoder $p_\phi$, and learned source/target type embeddings $e_s, e_t$ produce

$$h_{s,g} = f_\phi(\mathcal{N}_{s,g}) + p_\phi(c_{s,g}) + e_s, \qquad h_{t,g} = f_\phi(\mathcal{N}_{t,g}) + p_\phi(c_{t,g}) + e_t. \tag{5}$$

We prepend $R$ learned task tokens to the source and target patch tokens and process the complete sequence with self-attention. The outputs corresponding to the learned tokens form $Z = [z_1, \ldots, z_R]$, while their mean $\bar{z} = R^{-1} \sum_{r=1}^{R} z_r$ provides a global task vector. Thus, the task representation $z$ in Equation 2 is instantiated by the pair $(Z, \bar{z})$.

**Task-conditioned predictor.** DeformPIC conditions its query Transformer through a single global vector using adaptive layer normalization. Our predictor retains this global conditioning pathway and adds token-level access to the support evidence. We independently partition $x_q$ into local patches and encode each query token using its patch feature and three-dimensional center position. Each predictor block then performs query self-attention, cross-attention from the query tokens to $Z$, and an MLP update. The global vector $\bar{z}$ generates the adaptive scale, shift, and gate parameters for all three branches through AdaLN-Zero (Peebles & Xie, 2023). Finally, a task-conditioned output head decodes each query token into a set of point coordinates, whose union forms the prediction $\hat{y}_q$.

## 5 Experiments

### 5.1 Experimental Setup

**Benchmarks and evaluation protocol.** We evaluate task generalization on PIC-Bench and downstream adaptation on the original ShapeNet-In-Context benchmark (Fang et al., 2023). PIC-Bench contains 10,000 test episodes, with 2,000 episodes for each of its five tasks. Each task spans five difficulty levels, and every point cloud contains 1,024 points. We use the standard one-shot protocol throughout the main experiments: each model receives one support source–target pair and predicts the target point cloud for an independently sampled query. For ShapeNet-In-Context, we evaluate reconstruction, denoising, and registration using the original test protocol.

Table 1: Results on PIC-Bench. Throughout the tables, Chamfer distance L2 (CD) is scaled by $10^3$; lower is better. Dataset denotes the source of training shapes.

| | | Deterministic tasks | | | | Ambiguous tasks | | | All |
|---|---|---|---|---|---|---|---|---|---|
| Method | Dataset | SSD | SCR | MVM | Avg. | RPC | POR | Avg. | Overall |
| Identity | – | 7.6 | 30.9 | 16.7 | 18.4 | 51.9 | 25.7 | 38.8 | 26.6 |
| PIC-Cat (Fang et al., 2023) | ShapeNet | 29.7 | 31.8 | 31.3 | 30.9 | 53.7 | 40.0 | 46.8 | 37.3 |
| PIC-Sep (Fang et al., 2023) | ShapeNet | 107.0 | 94.7 | 104.4 | 102.0 | 131.3 | 103.3 | 117.3 | 108.1 |
| DeformPIC (Lin et al., 2026) | ShapeNet | 90.9 | 69.3 | 87.1 | 82.4 | 137.7 | 88.5 | 113.1 | 94.7 |
| DeformPIC* (Lin et al., 2026) | ShapeNet | 24.4 | 25.5 | 24.9 | 24.9 | 55.4 | 27.8 | 41.6 | 31.6 |
| TETRIS (Ours) | ShapeNet | **5.4** | 9.1 | 10.3 | 8.3 | 6.8 | 18.1 | 12.4 | 9.9 |
| TETRIS (Ours) | OpenShape | 5.6 | **7.9** | **8.8** | **7.4** | **4.5** | **16.9** | **10.7** | **8.7** |

**Baselines.** We compare against PIC-Cat and PIC-Sep (Fang et al., 2023), which concatenate or separately encode the support and query point clouds, and DeformPIC (Lin et al., 2026), which formulates Point ICL as conditional point deformation. We additionally report an Identity baseline that directly returns the query input. To separate the effect of architecture from the training framework, we construct DeformPIC*, which uses the same improved DeformPIC backbone as TETRIS but is trained from scratch with the original supervised Point ICL protocol on reconstruction, denoising, registration, and part segmentation.

### 5.2 Implementation Details

Our main TETRIS model is trained for 100 epochs on 829,460 normalized point clouds from OpenShape (Liu et al., 2023), excluding the LVIS split used by PIC-Bench. To examine the effect of the pre-training dataset, we additionally train TETRIS from scratch for 300 epochs on 41,952 clean ShapeNet (Chang et al., 2015) training shapes. Both variants use the same architecture, atomic operator library, and self-supervised objective, without using any PIC-Bench training pairs. The operator library contains global and local point addition, global and local point removal, rotation, scaling, and jittering. Architecture configurations, operator definitions, and optimization and evaluation protocols are provided in Appendix B.

### 5.3 Generalization on PIC-Bench

**Quantitative results.** Table 1 shows that existing Point ICL models do not readily transfer from their predefined training tasks to the structured transformation processes in PIC-Bench. PIC-Cat obtains an overall Chamfer distance of 37.3, while PIC-Sep and DeformPIC reach 108.1 and 94.7, respectively. All three are worse than directly returning the corrupted query with the Identity baseline (26.6). This result exposes the limitation hidden by evaluation within familiar task families: strong performance on the original protocol does not imply that the support-conditioned behavior transfers to a new transformation process. TETRIS achieves the best result on every PIC-Bench task and reduces the overall Chamfer distance to 8.7. The improvement holds for both deterministic tasks (7.4 on average) and ambiguous tasks (10.7), rather than being driven by a single restoration setting. The comparison with DeformPIC* is particularly informative. Although DeformPIC* shares the same backbone, its original supervised training protocol yields an overall distance of 31.6. TETRIS trained on OpenShape reduces this error by 72.3%, showing that the improved architecture alone does not account for its transfer performance. Training TETRIS on ShapeNet also yields an overall CD of 9.9, compared with 31.6 for DeformPIC* trained on ShapeNet. This indicates that the transfer advantage is not solely attributable to using OpenShape as the source of training shapes.

**Qualitative results.** Figure 2 illustrates the model behaviors behind the aggregate results. PIC-Cat frequently loses the query geometry, while DeformPIC* often applies an incomplete or incorrect correction despite using the same improved backbone as TETRIS. In contrast, TETRIS more faithfully preserves the query identity while applying the denoising, clutter-removal, alignment, completion, or occlusion-recovery behavior demonstrated by the support pair.

### 5.4 Finetuning on Downstream Tasks

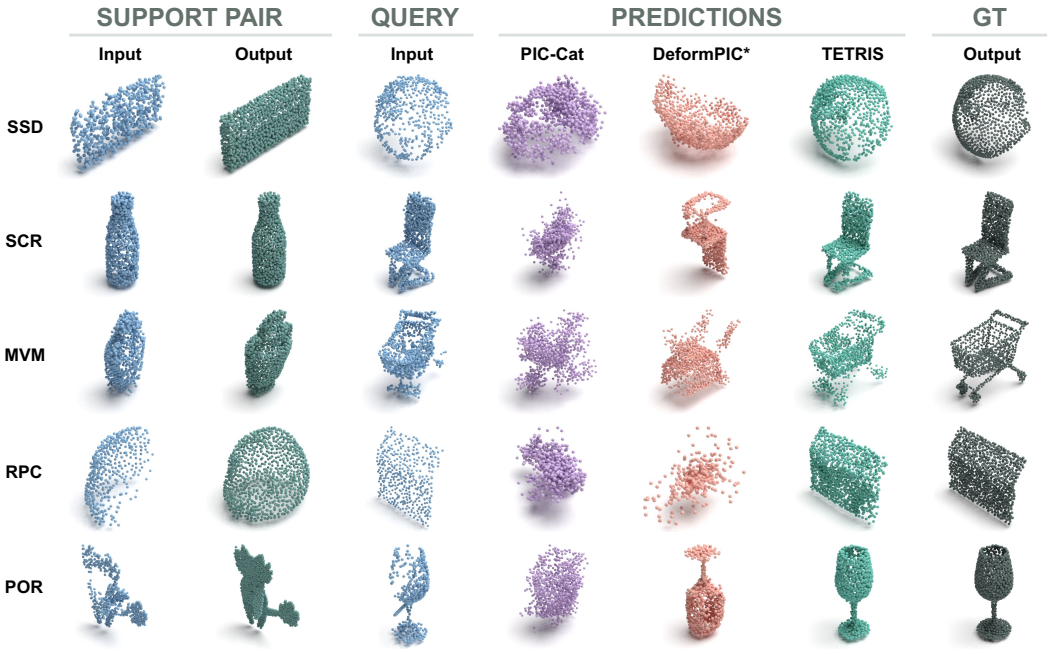



Figure 2: Qualitative comparison on PIC-Bench. We show selected level-3 examples where TETRIS succeeds while the supervised baselines produce distorted or incomplete geometry. Each row uses the same support pair and query across all methods.

We next examine whether the task representation learned by TETRIS provides a useful initialization when downstream training pairs become available. We fine-tune the TETRIS-pre-trained backbone using the original supervised Point ICL protocol and denote the resulting model as $\text{TETRIS}_{\text{f.t.}}$. DeformPIC* uses the same improved backbone and training protocol but starts from random initialization. This controlled comparison therefore isolates the effect of TETRIS pre-training. As shown in Table 2, TETRIS without downstream fine-tuning yields an average Chamfer distance of 10.8, reflecting the distribution difference between its self-supervised atomic training and the predefined benchmark tasks. Fine-tuning reduces the average to 2.3, compared with 2.4 for DeformPIC*, with the largest gains on denoising and registration. These results show that the self-supervised representation remains useful as an initialization for supervised adaptation.

Table 2: ShapeNet-In-Context results.

| Method | Rec. | Den. | Reg. | Avg. |
|---|---|---|---|---|
| PIC-Cat (Fang et al., 2023) | 4.9 | 6.0 | 14.4 | 8.4 |
| PIC-Sep (Fang et al., 2023) | 4.4 | 7.5 | 8.6 | 6.8 |
| MICAS-Cat (Shao et al., 2025) | 4.7 | 4.6 | 9.8 | 6.4 |
| MICAS-Sep (Shao et al., 2025) | 4.3 | 5.1 | 3.7 | 4.5 |
| PromptPath-Cat (Zhang et al., 2026) | 4.3 | 4.1 | 7.8 | 5.4 |
| PromptPath-Sep (Zhang et al., 2026) | 3.4 | 3.3 | 2.7 | 3.1 |
| DeformPIC (Lin et al., 2026) | 2.7 | 3.5 | **2.0** | 2.7 |
| DeformPIC* (Lin et al., 2026) | **2.2** | 2.4 | 2.5 | 2.4 |
| $\text{TETRIS}_{\text{1-shot}}$ | 4.3 | 3.7 | 24.4 | 10.8 |
| $\text{TETRIS}_{\text{f.t.}}$ | 2.3 | **2.3** | 2.3 | **2.3** |

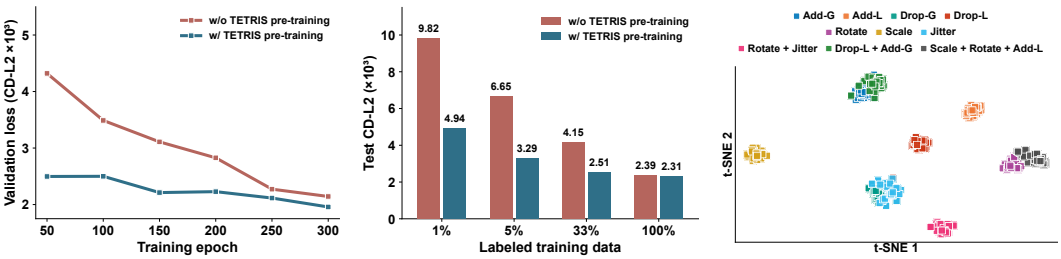
## 5.5 Analysis

**Effect of Atomic Composition.** We sample composition length uniformly from 1 to $M$, keeping other training settings fixed. Table 3 shows that $M = 3$ reduces overall CD from 12.64 ($M = 1$) to 8.73 (30.9%), supporting the value of composition over single-operator training. With $M = 5$, CD is 9.04.

Table 3: Effect of maximum composition length on PIC-Bench.

| $M$ | Det. | Amb. | Overall |
|---|---|---|---|
| 1 | 13.63 | 11.15 | 12.64 |
| 3 | **7.43** | 10.70 | **8.73** |
| 5 | 8.16 | **10.37** | 9.04 |

**Support Utilization Analysis.** We evaluate TETRIS with zeroed support clouds and support pairs shuffled across episodes. We also train a query-only




**(a) Training efficiency** **(b) Data efficiency** **(c) Task representations**

Figure 3: Task representations and adaptation efficiency of TETRIS. **(a)** Validation Chamfer distance during full-data fine-tuning of the same improved DeformPIC backbone, shown every 50 epochs. **(b)** Test Chamfer distance after 300 epochs using different fractions of labeled ShapeNet-In-Context training pairs. Lower is better in **(a)** and **(b)**. **(c)** t-SNE of task embeddings induced by atomic and compositional transformations on held-out objects.

model with a fixed zero task condition, matching TETRIS's training data, atomic compositions, predictor, and optimization schedule.

Table 4 shows that correct support reduces CD from 11.8 for the query-only model to 8.7, a 26.0% reduction, while improving F-score from 37.2% to 41.2%. Empty and shuffled support yield CD values of 11.8 and 12.8, respectively. These results show that episode-specific support improves prediction beyond the restoration prior learned from query geometry alone.

Table 4: Support-utilization analysis on PIC-Bench.

| Condition | Overall CD ↓ | F-score (%) ↑ |
|---|---|---|
| Empty support | 11.8 | 35.7 |
| Random support | 12.8 | 35.7 |
| Query-only | 11.8 | 37.2 |
| TETRIS (Ours) | **8.7** | **41.2** |

**Adaptation Efficiency** Figures 3**(a)** and **(b)** examine the utility of TETRIS pre-training for downstream adaptation. With full supervision, TETRIS pre-training produces lower validation loss throughout fine-tuning and attains a lower final validation loss under the same 300-epoch budget. Its benefit is largest when labels are scarce: it reduces test Chamfer distance from 9.8 to 4.9 with 1% of the training data and from 6.7 to 3.3 with 5%. The advantage persists at 33% (4.1 versus 2.5) and narrows only under full supervision (2.4 versus 2.3). These results show that the task representation learned by TETRIS provides a more data-efficient initialization for supervised adaptation.

**Task Representation Analysis** Figure 3**(c)** visualizes task embeddings extracted from different held-out objects. The embeddings group according to the applied atomic and compositional transformations. This observed grouping is consistent with the embeddings retaining transformation-related information across object instances.

## 6 Conclusion

This work revisits Point ICL by examining whether support-conditioned prediction transfers beyond the transformation processes used for training. We highlight that existing evaluation protocols admit a task-recognition shortcut: performance on familiar task families cannot distinguish transformation inference from the recognition and execution of a familiar task. To address this limitation, we introduce PIC-Bench, which evaluates five structured point cloud restoration tasks spanning deterministic and ambiguous settings, and propose TETRIS, a self-supervised task representation learning framework based on online compositions of atomic point-set operators. TETRIS achieves the best performance across all five PIC-Bench tasks and reduces the overall CD from 31.6 to 8.7 compared with the same backbone trained under the original supervised protocol. TETRIS pre-training also improves downstream data efficiency, reducing Chamfer distance from 9.8 to 4.9 with only 1% of the labeled training data.

### AI USE STATEMENT

In this work, we used generative AI tools solely to improve the readability and linguistic quality of the manuscript and to assist with software implementation. We did not use generative AI tools to formulate research ideas or hypotheses, design the methodology or experiments, develop theoretical or conceptual frameworks, derive mathematical claims or proofs, analyze data, or interpret experimental results. All AI-assisted text and code were manually reviewed by the authors, and the resulting code was tested and verified for correctness. The authors take full responsibility for the final content of the paper and all accompanying artifacts.

## REFERENCES

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *CVPR*, pp. 15619–15629, 2023.

Randall Balestriero and Yann LeCun. Lejepa: Provable and scalable self-supervised learning without the heuristics. *arXiv preprint arXiv:2511.08544*, 2025.

Amir Bar, Yossi Gandelsman, Trevor Darrell, Amir Globerson, and Alexei Efros. Visual prompting via image inpainting. *NeurIPS*, 35:25005–25017, 2022.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *NeurIPS*, 2020.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *2021 IEEE/CVF international conference on computer vision (ICCV)*, pp. 9630–9640. IEEE, 2021.

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

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, pp. 1597–1607. PmLR, 2020.

Zhongbin Fang, Xiangtai Li, Xia Li, Joachim M Buhmann, Chen Change Loy, and Mengyuan Liu. Explore in-context learning for 3d point cloud understanding. *NeurIPS*, 36:42382–42395, 2023.

Generalist-Team. Gen-1.5: Embodied foundation models are one-shot learners. *Generalist AI Blog*, 2026. https://generalistai.com/blog/gen-1.5.

Page C. Goddard, Lindsay M. Smith, Vudtiwat Ngampruetikorn, and David J. Schwab. When can in-context learning generalize out of task distribution? In *ICML*, volume 267, 2025.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *NeurIPS*, 33:21271–21284, 2020.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, pp. 9729–9738, 2020.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, pp. 16000–16009, 2022.

Roee Hendel, Mor Geva, and Amir Globerson. In-context learning creates task vectors. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 9318–9333, 2023.

Jincen Jiang, Qianyu Zhou, Yuhang Li, Xuequan Lu, Meili Wang, Lizhuang Ma, Jian Chang, and Jian Jun Zhang. Dg-pic: Domain generalized point-in-context learning for point cloud understanding. In *ECCV*, pp. 455–474. Springer, 2024a.

Jincen Jiang, Qianyu Zhou, Yuhang Li, Xinkui Zhao, Meili Wang, Lizhuang Ma, Jian Chang, Jian J Zhang, and Xuequan Lu. Pcotta: Continual test-time adaptation for multi-task point cloud understanding. *NeurIPS*, 37:96229–96253, 2024b.

Jincen Jiang, Qianyu Zhou, Yuhang Li, Kui Su, Meili Wang, Jian Chang, Jian Jun Zhang, and Xuequan Lu. Mamba learns in context: Structure-aware domain generalization for multi-task point cloud understanding. In *CVPR*, pp. 39098–39110, 2026.

Chengxing Lin, Wenju Xu, Jian Zhu, Yongwei Nie, Ruichu Cai, and Xuemiao Xu. Patchmixing masked autoencoders for 3d point cloud self-supervised learning. *IEEE TCSVT*, 34(10):9882–9897, 2024.

Chengxing Lin, Jinhong Deng, Yinjie Lei, and Wen Li. Deformation-based in-context learning for point cloud understanding. In *CVPR*, pp. 39001–39010, 2026.

Ziqian Lin and Kangwook Lee. Dual operating modes of in-context learning. In *ICML*, 2024.

Mengyuan Liu, Zhongbin Fang, Xia Li, Joachim M Buhmann, Deheng Ye, Xiangtai Li, and Chen Change Loy. Point-in-context: Understanding point cloud via in-context learning: M. liu et al. *IJCV*, 134(5):236, 2026.

Minghua Liu, Ruoxi Shi, Kaiming Kuang, Yinhao Zhu, Xuanlin Li, Shizhong Han, Hong Cai, Fatih Porikli, and Hao Su. Openshape: Scaling up 3d shape representation towards open-world understanding. *NeurIPS*, 36:44860–44879, 2023.

Jane Pan, Tianyu Gao, Howard Chen, and Danqi Chen. What in-context learning "learns" in-context: Disentangling task recognition and task learning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 8298–8319, 2023.

Yatian Pang, Wenxiao Wang, Francis EH Tay, Wei Liu, Yonghong Tian, and Li Yuan. Masked autoencoders for point cloud self-supervised learning. In *ECCV*, pp. 604–621. Springer, 2022.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, pp. 4195–4205, 2023.

Allan Raventós, Mansheej Paul, Feng Chen, and Surya Ganguli. Pretraining task diversity and the emergence of non-bayesian in-context learning for regression. *NeurIPS*, 36:14228–14246, 2023.

Feifei Shao, Ping Liu, Zhao Wang, Yawei Luo, Hongwei Wang, and Jun Xiao. Micas: Multi-grained in-context adaptive sampling for 3d point cloud processing. In *CVPR*, pp. 6616–6626, 2025.

Skild-AI. Introducing s1: In-context learning for robotics. *Skild AI Blog*, August 2026. URL `https://skild.ai/blogs/s1`.

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 *CVPR*, pp. 1588–1597, 2019.

Xinlong Wang, Wen Wang, Yue Cao, Chunhua Shen, and Tiejun Huang. Images speak in images: A generalist painter for in-context visual learning. In *CVPR*, pp. 6830–6839, 2023.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, pp. 1912–1920, 2015.

Jaemyung Yu, Jaehyun Choi, Dong-Jae Lee, HyeongGwon Hong, and Junmo Kim. Self-supervised transformation learning for equivariant representations. *NeurIPS*, 37:83068–83090, 2024.

Hangrui Zhang, Feifei Shao, Yawei Luo, Ping Liu, Jiaxiang Liu, Zuoqi Tang, Zhao Wang, Hongwei Wang, and Jun Xiao. Promptpath: Prompt-adaptive computational pathways for in-context learning. *arXiv preprint arXiv:2608.02129*, 2026.

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. *NeurIPS*, 35:27061–27074, 2022.

Xiangdong Zhang, Shaofeng Zhang, and Junchi Yan. Pcp-mae: Learning to predict centers for point masked autoencoders. *NeurIPS*, 37:80303–80327, 2024.

# A PIC-BENCH TASK SUBTYPES

Each PIC-Bench task contains 4 process subtypes and 5 difficulty levels. Subtypes vary the degradation mechanism, while levels control its severity. Continuous parameters provide diverse process realizations within each subtype–level combination, and episodes are balanced uniformly across all combinations within each task. We describe the distinctions among the subtypes below.

## A.1 STRUCTURED SENSOR DENOISING

*Range bias field* introduces a smooth, geometry-dependent bias in the measured range. The direction and magnitude of the displacement vary across the surface according to a shared sensor field. *Quantized depth banding* discretizes measured depth into bands, producing structured terraces along the sensor direction together with mild range perturbations. *Scanline wobble* applies a periodic displacement across scanlines, causing the observed surface to oscillate in the radial and tangential directions. *Multipath ghost denoising* adds displaced ghost returns that mimic secondary propagation paths. The input therefore contains both perturbed true returns and coherent false surfaces.

## A.2 SCENE CLUTTER REMOVAL

*Support plane* places a noisy planar surface beneath or near the target, modeling the support surfaces commonly retained during object scanning. *Foreign fragment* inserts a transformed fragment from another object category near the target, producing clutter with coherent object-like geometry. *Boundary leakage* adds points extending outward from a local target boundary, modeling imperfect foreground segmentation around the object contour. *Mixed clutter* combines a support plane, boundary leakage, and a foreign object fragment within the same observation.

## A.3 MULTI-VIEW MISALIGNMENT

*Pairwise pose error* fuses two partial scans with inconsistent estimated poses, so their overlapping regions no longer align. *Multi-scan drift* accumulates pose errors across a sequence of partial scans, producing progressively increasing drift in the fused point cloud. *One bad view* fuses several scans but applies a substantially larger pose error to one view than to the others. *Low overlap* combines pose errors with partial scans that share little visible geometry, making their relative alignment less constrained.

## A.4 RENDERED PARTIAL COMPLETION

*Single view* retains the surface visible from one viewpoint, leaving the opposite side of the object unobserved. *Sparse multiview* combines a small number of viewpoints, increasing coverage while leaving substantial regions unseen. *Edge thinned* removes additional points from a rendered partial observation, producing sparse and irregular visibility near observed boundaries. *Density falloff* gradually reduces point density with depth, producing a partial observation whose sampling becomes sparser away from the dominant visible surface.

## A.5 PHYSICAL OCCLUSION RECOVERY

*Object occlusion* places another object between the sensor and target, hiding part of the target while retaining points from the foreground object. *Planar occlusion* places a screen between the sensor and target, producing a compact missing region together with a coherent foreground surface. *Multiple-object occlusion* places several foreground objects at different positions, creating multiple disconnected hidden regions. *Near-field occlusion* places a foreground object close to the target, producing stronger and more localized interactions between the two geometries.

# B Implementation Details

## B.1 Atomic Operator Design

Each training episode samples $L \in \{1, 2, 3\}$ operators uniformly and without replacement, then applies them in the sampled order. The support and query share the operator identities, order, and parameters, while stochastic point indices and noise are sampled independently. For each operator, we first draw one of five magnitude levels uniformly and then sample its continuous parameter within the corresponding interval in Table 5.

Table 5: Magnitude ranges for the atomic operators across five levels.

| Parameter | Level 1 | Level 2 | Level 3 | Level 4 | Level 5 |
|---|---|---|---|---|---|
| Removal ratio | $[.02, .10]$ | $[.10, .25]$ | $[.25, .45]$ | $[.45, .65]$ | $[.65, .80]$ |
| Addition ratio | $[.02, .10]$ | $[.10, .22]$ | $[.22, .38]$ | $[.38, .55]$ | $[.55, .70]$ |
| Rotation angle ($^\circ$) | $[0, 15]$ | $[15, 35]$ | $[35, 60]$ | $[60, 100]$ | $[100, 180]$ |
| Jitter standard deviation | $[.002, .008]$ | $[.008, .018]$ | $[.018, .035]$ | $[.035, .060]$ | $[.060, .100]$ |
| Local-cluster offset | $[.08, .16]$ | $[.16, .26]$ | $[.26, .40]$ | $[.40, .58]$ | $[.58, .80]$ |
| Local-cluster standard deviation | $[.01, .03]$ | $[.03, .06]$ | $[.06, .10]$ | $[.10, .16]$ | $[.16, .24]$ |

**Global point addition.** This operator introduces spatially unstructured outliers. It samples points uniformly from $[-1.5, 1.5]^3$ and either replaces uniformly selected input points or merges the outliers with the input before resampling to the original point count. The two modes are selected with equal probability, and the outlier proportion follows the addition ratio in Table 5.

**Local point addition.** This operator creates a spatially concentrated outlier cluster. It selects a surface region by projecting the input points onto a random direction, places the cluster center beyond that region along the same direction, and samples the added points from an isotropic Gaussian around the center. The cluster offset, standard deviation, and outlier proportion are determined by the sampled magnitude level. As in global addition, the outliers either replace selected input points or are merged with the input before resampling.

**Global point removal.** This operator removes points distributed across the entire shape. It selects a uniformly random subset according to the sampled removal ratio. The selected points are either set to zero or removed and replaced by resampling the surviving points, with equal probability.

**Local point removal.** This operator removes a contiguous directional region. It samples a random direction and selects the points with the largest projections along that direction, with the number of selected points controlled by the removal ratio. The selected region is then either set to zero or replaced by resampling the remaining points.

**Rotation.** This operator samples a random axis and an angle from the Haar measure on $\mathrm{SO}(3)$ conditioned on the interval in Table 5. We also apply a random target reference frame, while keeping the relative rotation shared by the support and query.

**Scaling.** This operator applies an isotropic scale factor shared by the support and query. At magnitude level $m$, we sample $d \sim \mathcal{U}((m-1)\log 2/5, m \log 2/5)$ and select the scale factor uniformly from $\{\exp(d), \exp(-d)\}$, allowing expansion and contraction with equal probability.

**Jittering.** This operator adds independent zero-mean Gaussian noise to every point coordinate. The standard deviation follows the sampled magnitude level, and each noise value is clipped to three standard deviations.

### B.2 Architecture Configuration

All point clouds contain 1,024 points. We use $G = 64$ local patches with $S = 32$ points per patch and $R = 8$ task tokens, all with a feature dimension of 384. The task encoder contains four Transformer blocks, while the predictor contains eight blocks; both use six attention heads. The resulting model has 38.6M parameters. The predictor generates $G \times S = 2{,}048$ points, which we reduce to 1,024 points by farthest-point sampling before computing the loss or evaluation metrics.

### B.3 Optimization and Evaluation Protocol

Table 6 summarizes the configurations for self-supervised pre-training and supervised downstream fine-tuning. Training uses four GPUs with distributed data parallelism, automatic mixed precision, and FlashAttention where available.

Table 6: Training configurations for TETRIS pre-training and downstream fine-tuning. DeformPIC* uses the fine-tuning configuration but starts from random initialization.

| | TETRIS pre-training | Downstream fine-tuning |
|---|---|---|
| ***Training setup*** | | |
| Training data | OpenShape | ShapeNet-In-Context |
| Initialization | Random | TETRIS epoch-100 checkpoint |
| Input points | 1,024 | 1,024 |
| Global batch size | 512 | 128 |
| Training epochs | 100 | 300 |
| ***Optimizer*** | | |
| Optimizer | AdamW | AdamW |
| Adam $\beta_1, \beta_2$ | $0.9, 0.999$ | $0.9, 0.999$ |
| Learning rate | $5 \times 10^{-4}$ | $1 \times 10^{-4}$ |
| Weight decay | 0.05 | 0.05 |
| Learning-rate schedule | Cosine | Cosine |
| Warmup epochs | 10 | 10 |
| Warmup and minimum learning rate | $1 \times 10^{-6}$ | $1 \times 10^{-6}$ |
| Random seed | 0 | 0 |

Each pre-training epoch visits every remaining OpenShape point cloud once as the query, while the support point cloud and atomic composition are resampled online. Fine-tuning constructs a fresh optimizer and scheduler after loading the pre-trained weights. For the data-efficiency study, we use 1%, 5%, 33%, and 100% of the supervised training set, with subset seed 0.

We report the final checkpoints for TETRIS and all controlled runs: epoch 100 for self-supervised training and epoch 300 for supervised training; validation-best checkpoints are retained only for monitoring. PIC-Bench evaluation uses one support pair, seed 0, deterministic inference, and a single forward pass without parameter updates. We use the released checkpoints for the published baselines.