DOMAIN3D-CIL: EVALUATING AND IMPROVING HETEROGENEOUS 3D CLASS-INCREMENTAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

3D perception plays a crucial role in real-world applications such as autonomous driving, robotics, and AR/VR. In practical scenarios, 3D perception models need to continually adapt to newly emerging 3D object categories, making class-incremental learning (CIL) particularly important. However, 3D point clouds are highly heterogeneous: objects from the same class may come not only from the clean CAD domain, but also from RGB-D camera scans of varying quality, video reconstructions, or even corrupted observations. This heterogeneity increases the complexity of intra-class feature distributions, posing challenges for effective representation and memory in incremental learning. To investigate this problem, we establish the **Domain3D-CIL** training and evaluation protocol, which includes point cloud data from heterogeneous domains, and adapt a wide range of mainstream CIL methods to the 3D modality for systematic evaluation. Building on this, we introduce **PolyMem**, an exemplar-free approach that implicitly models rich higher-order statistics of feature distributions to enhance cross-domain continual learning. Experiments demonstrate that our method improves average performance across domains in heterogeneous 3D class-incremental learning, with particularly notable accuracy gains on real-world domains.

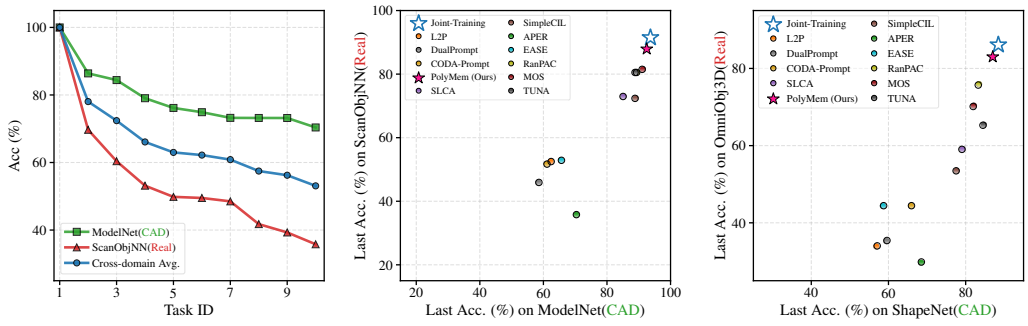
1 INTRODUCTION

3D perception holds broad application value in a wide range of real-world scenarios (Lang et al., 2019; Qi et al., 2019; Chen et al., 2019). For example, autonomous driving systems require real-time recognition and classification of surrounding obstacles, while robots rely on 3D perception to understand and interact with complex physical environments. As these systems operate in dynamic practical environments, the data they encounter can evolve over time, requiring 3D models to continually learn under changing data distributions. Class-incremental learning (CIL) addresses this need by enabling models to incrementally learn new classes while preserving knowledge of previously learned ones (Rebuffi et al., 2017; Li & Hoiem, 2017). In this work, we consider the more challenging practical setting in which historical samples cannot be retained for subsequent training. Under this constraint, mitigating catastrophic forgetting, a problem extensively studied in 2D CIL, becomes a central challenge for 3D CIL (Chowdhury et al., 2022).

However, the intra-class heterogeneity of 3D point clouds poses additional challenges for 3D CIL. When learning new categories from continually aggregated and multi-source data, training samples may include point clouds acquired through different techniques or 3D assets collected from crowdsourced platforms. 3D data may originate from Computer-Aided Design (CAD), RGB-D scanning, or video-based 3D reconstruction (Wu et al., 2015; Dai et al., 2017; Sun et al., 2020), and exhibit varying degrees of noise, missing geometry, and artifacts (Ren et al., 2022). Consequently, samples from the same class may share semantic labels while differing substantially in geometry and acquisition characteristics (Qin et al., 2019), resulting in complex intra-class feature distributions. Meanwhile, complete information about data acquisition may not be available when aggregating multiple sources (Deitke et al., 2023). Motivated by this practical constraint, we therefore focus on heterogeneous 3D CIL without relying on domain labels during training.

To systematically evaluate incremental learning in this setting, we establish a comprehensive training and evaluation protocol, **Domain3D-CIL**. Under this protocol, datasets consist of heterogeneous mixtures of clean CAD point clouds and real-world or corrupted point clouds. Samples from different

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.



(a) (b) (c)

Figure 1: **CIL performance on heterogeneous 3D data. (a)** Per-domain accuracy and their average for APER (Zhou et al., 2025a) on ModelNet(CAD) + ScanObjectNN(real) throughout incremental learning. **(b) & (c)** Final per-domain accuracy of representative PTM-based CIL methods on ModelNet + ScanObjectNN and ShapeNet(CAD) + OmniObject3D(real), respectively. PolyMem improves accuracy on both domains over the compared CIL baselines, with particularly notable gains on real-world data. Joint-training results are shown for reference.

domains are mixed within shared semantic classes and use the same semantic labels. To support controlled benchmark construction and per-domain evaluation, we combine datasets with known sources. Domain labels are used only for data organization and evaluation, not as inputs to model training. Detailed descriptions of Domain3D-CIL are provided in Sec. 3. For the baselines, we adopt the widely used Pre-Trained Model (PTM)-based CIL paradigm (Zhou et al., 2024a) and systematically adapt representative methods to the 3D modality for fair comparison. As illustrated by APER in Fig. 1a, accuracy on heterogeneous 3D data declines as incremental learning progresses, highlighting the challenge of maintaining performance while learning new classes.

Some PTM-based CIL methods achieve strong performance by accumulating feature statistics to retain information about old classes without storing historical samples. For example, SimpleCIL uses class prototypes (Zhou et al., 2025a), while RanPAC accumulates statistics in a nonlinearly transformed random feature space (McDonnell et al., 2023). These approaches highlight that the effectiveness of statistical memory depends not only on which statistics are retained, but also on the feature representation in which they are computed. To accommodate the complex intra-class distributions of heterogeneous 3D data, we explore enriching statistical memory through structured higher-order feature interactions. To this end, we introduce **PolyMem**, which constructs a feature mapping through a structured polynomial approximation of the RBF kernel and accumulates statistics in the resulting space for exemplar-free incremental learning. As shown in Fig. 1b and Fig. 1c, PolyMem achieves higher accuracy on both CAD and real-world domains.

In summary, the main contributions of this paper are as follows:

- We establish **Domain3D-CIL**, a protocol for class-incremental learning with heterogeneous 3D data without domain labels during training.
- We adapt representative CIL baselines to the 3D modality and systematically evaluate them under a unified protocol.
- We propose **PolyMem**, enriching statistical memory with higher-order interactions to improve average accuracy, especially on real-world data.

2 Related Works

3D Representation Learning and Data Heterogeneity. 3D point cloud representation learning has evolved from shared-MLP architectures such as PointNet and PointNet++ (Qi et al., 2017a;b) to Transformers with stronger global context modeling (Zhao et al., 2021; Wu et al., 2022; 2024). Subsequent advances include self-supervised pre-training through masked point modeling (Yu et al., 2022; Pang et al., 2023; Zhang et al., 2022) and multimodal alignment across text, images, and point clouds (Xue et al., 2023; Liu et al., 2023; Xue et al., 2024; Zhou et al., 2023b).

Recent studies further explore unified representations across heterogeneous 3D sources (Wu et al., 2025; Zhang et al., 2025). Different data construction processes, including CAD modeling (Wu et al., 2015; Chang et al., 2015), RGB-D scanning (Uy et al., 2019; Dai et al., 2017; Wu et al., 2023), and multi-view reconstruction (Reizenstein et al., 2021; Yu et al., 2023), introduce variations in geometry, noise, and intra-class feature distributions. This heterogeneity motivates effective representation and memory for class-incremental learning from multi-source 3D data.

Class-incremental Learning. Class-incremental learning aims to continuously learn new classes while retaining previously acquired knowledge (Zhou et al., 2024a). Models learn a sequence of tasks, each introducing new classes, without access to full historical data. Early approaches mitigate inter-task interference through exemplar replay (Rebuffi et al., 2017; Wu et al., 2019; Hou et al., 2019; Ma & Zhu, 2026), parameter regularization (Li & Hoiem, 2017; Kirkpatrick et al., 2017; Miao et al., 2021), or architectural expansion (Yan et al., 2021; Wang et al., 2022a; Zhou et al., 2023a). More recently, Pre-Trained Model (PTM)-based CIL has emerged as an important paradigm (Zhou et al., 2024a). These methods leverage strong pretrained representations, with some using frozen backbones and lightweight adaptation without retaining historical samples. Representative approaches include prompt-based strategies (Wang et al., 2022d;c; Smith et al., 2023), feature calibration (Zhang et al., 2023; Goswami et al., 2023), lightweight mappings (Gao et al., 2023; Sun et al., 2025), and subspace expansion (Zhou et al., 2024b; Wang et al., 2025; Zhou et al., 2025b). In the replay-free setting, some methods construct statistical memory by recording feature-space statistics. Specifically, SimpleCIL (Zhou et al., 2025a) uses class prototypes, while RanPAC (McDonnell et al., 2023) accumulates class-wise feature sums and a feature Gram matrix in a nonlinearly transformed random feature space. Our work builds on statistical memory and explores enhancing its representational capacity through structured higher-order feature interactions.

In the 3D domain, existing studies address incremental object recognition and few-shot class-incremental learning (Chowdhury et al., 2022; Cheraghian et al., 2024; Xiang et al., 2025). Beyond few-shot settings, some work investigates incremental learning of 3D objects or point clouds (Dong et al., 2023; 2021; Resani & Nasihatkon, 2025), while other work explores replay exemplar selection (Resani et al., 2025). These studies provide important foundations for 3D continual learning. Our work focuses on exemplar-free class-incremental learning from heterogeneous 3D data using pretrained representations.

Prior works have explored 2D domain-incremental learning (Wang et al., 2022b; 2024) or class-and-domain incremental learning (Xie et al., 2022; Park et al., 2024). We focus on heterogeneous 3D data mixed within shared semantic classes: samples from different sources coexist within the same incremental task, domain labels are not provided during training, and both average performance across domains and per-domain accuracy are evaluated as new classes are introduced.

3 DOMAIN3D-CIL PROTOCOL

We consider a sequence of T class-incremental tasks $\{\mathcal{T}_t\}_{t=1}^{T}$. The training set of task t is denoted by $\mathcal{D}_t = \{(x_{t,i}, y_{t,i})\}_{i=1}^{n_t}$, where $x_{t,i}$ and $y_{t,i}$ are the i-th sample and its label, n_t is the number of training samples, and $\mathcal{Y}_t$ is the task label set. Different tasks have mutually disjoint label sets, i.e., $\mathcal{Y}_t \cap \mathcal{Y}_{t'} = \emptyset$ for any $t \neq t'$. In our replay-free setting, the model is trained at each stage using only the current task data $\mathcal{D}_t$, without storing raw samples of classes from previous tasks, and performs classification over all observed classes $\mathcal{C}_t = \bigcup_{\tau=1}^{t} \mathcal{Y}_\tau$.

Building upon this formulation, we establish a heterogeneous 3D training and evaluation protocol, termed **Domain3D-CIL**. For each task t, the training set mixes samples from a CAD domain and one additional domain, i.e., $\mathcal{D}_t = \mathcal{D}_t^{\mathrm{CAD}} \cup \mathcal{D}_t^{\mathrm{O}}$, where $\mathcal{D}_t^{\mathrm{CAD}} = \{(x_{t,i}^{\mathrm{CAD}}, y_{t,i}^{\mathrm{CAD}})\}_{i=1}^{n_t^{\mathrm{CAD}}}$ and $\mathcal{D}_t^{\mathrm{O}} = \{(x_{t,j}^{\mathrm{O}}, y_{t,j}^{\mathrm{O}})\}_{j=1}^{n_t^{\mathrm{O}}}$. The additional domain O consists of real-world scanned point clouds, video-reconstructed point clouds, or corrupted CAD point clouds. For each task, the CAD subset covers the full task label space $\mathcal{Y}_t$, while the non-CAD subset contains only samples that can be semantically matched to CAD categories. Its label space therefore satisfies $\mathcal{Y}_t^{\mathrm{O}} \subseteq \mathcal{Y}_t$. Samples with the same semantic meaning across domains share the same class label rather than forming domain-specific categories. Unmatched non-CAD categories are excluded, while CAD categories without corresponding non-CAD samples are retained in $\mathcal{D}_t^{\mathrm{CAD}}$.

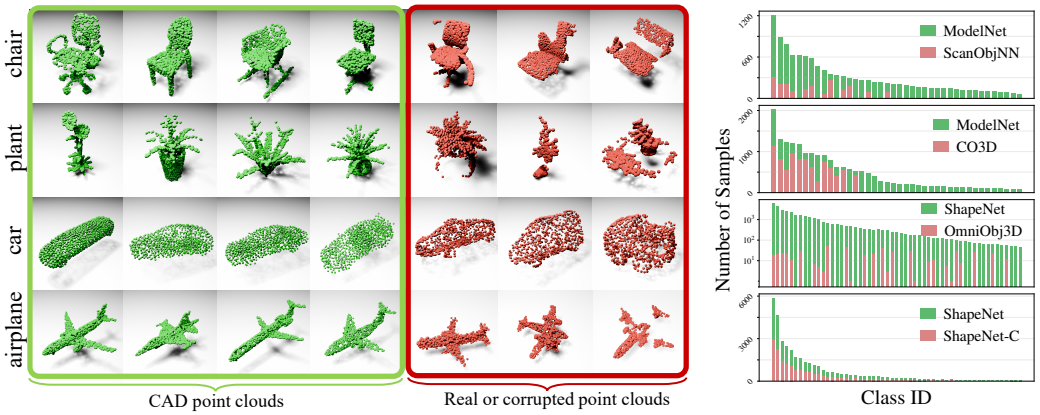


Figure 2: **Visualization of Domain3D-CIL. (a)** shows representative categories containing cross-domain samples from the mixed datasets. **(b)** shows the corresponding class distributions.

During training, the model observes only the mixed training set $\mathcal{D}_t$ and semantic labels; domain labels are **not provided**. During evaluation, test data are partitioned into domain-specific subsets, $\mathcal{D}_{t,\text{test}}^{\text{CAD}}$ and $\mathcal{D}_{t,\text{test}}^{\text{O}}$. The prediction space always includes all observed classes $\mathcal{C}_t$, while accuracy for each domain is computed on its available test samples from the observed classes. We report per-domain accuracy and use the equally weighted average of the two domain accuracies to measure average performance across domains. This setting accounts for the potential unavailability of complete generation and acquisition information when aggregating data from multiple sources. We construct the benchmark from datasets with known sources to support controlled construction and per-domain evaluation, but do not provide these source labels during training. Domain3D-CIL thus enables systematic evaluation of both average performance across domains and per-domain accuracy in heterogeneous 3D CIL.

Fig. 2 illustrates the mixed point cloud datasets used in Domain3D-CIL. Each row in Fig. 2a and Fig. 2b corresponds to the same dataset combination, as indicated by the legends in Fig. 2b. From top to bottom, the four combinations are ModelNet (Wu et al., 2015) + ScanObjectNN (Uy et al., 2019), ModelNet + CO3D (Reizenstein et al., 2021), ShapeNet (Chang et al., 2015) + OmniObject3D (Wu et al., 2023), and ShapeNet + ShapeNet-C (Ren et al., 2022). As shown in Fig. 2a, CAD point clouds from ModelNet and ShapeNet are generally clean and complete, while high-quality real-world scans from OmniObject3D also exhibit relatively clear geometry. In contrast, low-quality RGB-D scans from ScanObjectNN, video-reconstructed point clouds from CO3D, and corrupted point clouds from ShapeNet-C contain more artifacts, including missing regions, redundant points, and perturbations. Fig. 2b shows the corresponding class distributions, illustrating sample-count imbalances across classes and data sources. In particular, the third row uses a logarithmic vertical axis to highlight the substantially smaller sample count of OmniObject3D relative to ShapeNet in this dataset combination.

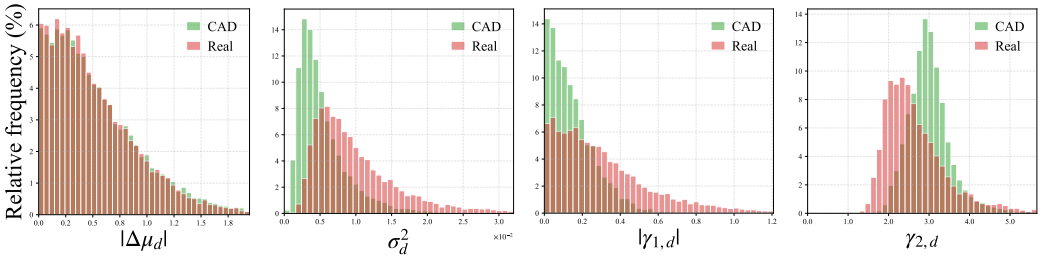
4 Methodology

4.1 Statistical Memory Representation

In exemplar-free class-incremental learning, statistical memory retains information about old classes by accumulating feature statistics. Given a frozen pre-trained backbone f, we encode each sample x_i as an L2-normalized feature $F_i = f(x_i)/\|f(x_i)\|_2 \in \mathbb{R}^D$ and apply a fixed mapping $\phi : \mathbb{R}^D \to \mathbb{R}^M$ to obtain $F_i' = \phi(F_i)$. The representational capacity of statistical memory depends not only on which statistics are retained, but also on the feature space in which they are computed. We adopt the accumulated-statistics ridge regression framework (Hoerl & Kennard, 1970) and focus on constructing a feature mapping for heterogeneous 3D data.

Samples from heterogeneous sources may exhibit distinct distributional properties even within the same class. To illustrate this heterogeneity, Fig. 3 compares relative-frequency histograms of per-dimension feature statistics between the CAD and real-world domains on ShapeNet + OmniObject3D.

For each class and domain, we compute four statistics along each of the 1024 backbone feature dimensions: (a) the absolute deviation of the domain-specific class mean from the mixed-domain class mean, (b) variance, (c) absolute skewness, and (d) kurtosis. The histograms are averaged over 10 semantically matched classes. The observed differences in these statistical profiles motivate us to enrich statistical memory through multi-order feature interactions.



(a) 1st: Mean (b) 2nd:Variance (c) 3rd:Skewness (d) 4th:Kurtosis

Figure 3: Histograms of 1st to 4th-order statistics on ShapeNet (CAD) + OmniObject3D (Real).

To capture such multi-order distributional characteristics, nonlinear mappings provide a natural mechanism to encode nonlinear interactions of the original features, as explored by RanPAC (McDonnell et al., 2023). We further investigate how these interactions are organized: under a fixed output dimensionality, we explicitly construct interactions of different orders and combine them using specified weights. To this end, **PolyMem** adopts an order-wise polynomial mapping to incorporate higher-order interactions into incremental statistical memory.

4.2 Structured Polynomial Mapping

Explicit polynomial mappings capture order-wise feature interactions but become prohibitively expensive at higher orders. For a D-dimensional feature vector, explicitly constructing a full k-th-order polynomial feature tensor requires $\mathcal{O}(D^k)$ elements, growing rapidly with the order. Therefore, PolyMem uses compact kernel approximations to represent these interactions and combine different orders within a fixed output dimensionality.

Specifically, we use the RBF kernel to specify the multi-order interactions and their weights (Schölkopf & Smola, 2002), following an existing polynomial kernel approximation framework (Cui et al., 2017). For two normalized features $F_i, F_j \in \mathbb{R}^D$,

$$k_{\mathrm{RBF}}(F_i, F_j) = \exp\big(-\gamma\|F_i - F_j\|_2^2\big) = e^{-2\gamma}\exp(2\gamma F_i^\top F_j), \tag{1}$$

where γ is the RBF scale parameter and is set using the median heuristic on normalized features. The final factor admits the Taylor expansion:

$$\exp(2\gamma F_i^\top F_j) = 1 + 2\gamma(F_i^\top F_j) + \frac{(2\gamma)^2}{2!}(F_i^\top F_j)^2 + \frac{(2\gamma)^3}{3!}(F_i^\top F_j)^3 + \cdots. \tag{2}$$

This expansion expresses RBF similarity as a weighted sum of polynomial interactions of different orders, with the coefficients specifying the contribution of each order. PolyMem truncates the expansion at order L, yielding

$$k^{(L)}_{\mathrm{Poly}}(F_i, F_j) = e^{-2\gamma}\sum_{n=0}^{L}\frac{(2\gamma)^n}{n!}(F_i^\top F_j)^n. \tag{3}$$

To avoid explicitly constructing polynomial feature tensors, we use TensorSketch (Pham & Pagh, 2013) to compactly approximate each term. Let $\mathrm{TS}_n(F_i) \in \mathbb{R}^{M_n}$ denote the sketch feature for the n-th-order interaction, satisfying $\mathrm{TS}_n(F_i)^\top \mathrm{TS}_n(F_j) \approx (F_i^\top F_j)^n$. Given a total output dimensionality M, we allocate one dimension to the constant term and evenly distribute the remaining dimensions across orders, such that $1 + \sum_{n=1}^{L} M_n = M$. The resulting mapping is

$$\phi_{\mathrm{Poly}}(F_i) = \left[e^{-\gamma},\ e^{-\gamma}\sqrt{\frac{2\gamma}{1!}}\mathrm{TS}_1(F_i)^\top,\ \ldots,\ e^{-\gamma}\sqrt{\frac{(2\gamma)^L}{L!}}\mathrm{TS}_L(F_i)^\top\right]^\top \in \mathbb{R}^M. \tag{4}$$

Consequently,

$$\phi_{\text{Poly}}(F_i)^\top \phi_{\text{Poly}}(F_j) \approx k^{(L)}_{\text{Poly}}(F_i, F_j) \tag{5}$$

This construction explicitly combines interactions of different orders within a fixed dimensionality: L controls the maximum interaction order, M_n determines the dimensionality allocated to each order, and the RBF expansion coefficients specify their weights. We apply this mapping to incremental statistical accumulation over frozen pre-trained features, keeping the coefficients fixed at their expansion values without learning them through backpropagation. Further background on kernel approximation is provided in Appendix F.

4.3 Incremental Updates and Prediction

For each sample in task t, PolyMem computes the mapped feature $F'_{t,i} = \phi_{\text{Poly}}(F_{t,i})$ and updates the statistics required for closed-form ridge regression (Hoerl & Kennard, 1970):

$$Q_t = Q_{t-1} + \sum_{i=1}^{n_t} F'_{t,i} e_{y_{t,i}}^\top, \qquad G_t = G_{t-1} + \sum_{i=1}^{n_t} F'_{t,i} {F'_{t,i}}^\top. \tag{6}$$

Here, $e_{y_{t,i}} \in \mathbb{R}^{|\mathcal{C}_t|}$ is the one-hot label vector. The statistics are initialized to zero. When new classes arrive, Q_{t-1} is first padded with zero columns to align with the full observed class set $\mathcal{C}_t$. Thus, $Q_t \in \mathbb{R}^{M \times |\mathcal{C}_t|}$ stores class-wise sums of mapped features, while $G_t \in \mathbb{R}^{M \times M}$ stores a global uncentered second-order statistic over all observed samples.

The classifier is obtained by closed-form ridge regression:

$$W_t = (G_t + \lambda_t I)^{-1} Q_t, \tag{7}$$

where λ_t is selected from a fixed grid using a validation split of the current task, and I is the identity matrix. For a test sample x, the model predicts over all observed classes:

$$\hat{y} = \arg\max_{c \in \mathcal{C}_t} \left[W_t^\top \phi_{\text{Poly}}(F(x)) \right]_c, \tag{8}$$

where $F(x)$ is the normalized backbone feature. The backbone and feature mapping remain fixed throughout incremental learning, ensuring statistics from different tasks in a common representation space where they can be directly accumulated. Given the same feature mapping and regularization coefficient, the resulting classifier is equivalent to batch ridge regression on all observed training samples, without revisiting historical samples.

Importantly, the polynomial mapping enriches the statistical memory by incorporating higher-order interactions. While Q_t accumulates polynomial features up to degree L, products between the degree-n and degree-m feature blocks in G_t encode degree-$(n+m)$ interactions, reaching degree $2L$. PolyMem therefore captures higher-order interactions of the original features through first- and second-order statistical accumulation in the mapped space.

The resulting statistical memory has a storage complexity of $\mathcal{O}(M^2 + M|\mathcal{C}_t|)$. The size of G_t remains fixed, while Q_t grows with the number of observed classes; neither grows directly with the number of historical samples. By combining compact polynomial features with accumulated statistics, PolyMem enables higher-order statistical memory without historical exemplar replay. Detailed storage and runtime analyses appear in Appendix C.

5 Experiments

5.1 Experimental Setup

Baselines. We adopt the following PTM-based CIL methods as baselines: L2P (Wang et al., 2022d), DualPrompt (Wang et al., 2022c), CODA-Prompt (Smith et al., 2023), SLCA (Zhang et al., 2023), SimpleCIL (Zhou et al., 2025a), APER (Zhou et al., 2025a), EASE (Zhou et al., 2024b), RanPAC (McDonnell et al., 2023), MOS (Sun et al., 2025), and TUNA (Wang et al., 2025). For a fair comparison, all baselines and PolyMem use the same pre-trained backbone, detailed in Appendix A. Classic CIL baselines appear in Appendix D.

Backbone. We use **Uni3D-Base** (Zhou et al., 2023b) pre-trained on Objaverse (Deitke et al., 2023) as the backbone. Uni3D shares a transformer-based architectural design with Vision Transformer (Dosovitskiy et al., 2021), which facilitates the adaptation of existing PTM-based CIL methods to 3D perception tasks.

Datasets and Task Splits. We use six public 3D datasets to construct four heterogeneous dataset combinations: ModelNet (Wu et al., 2015) + ScanObjectNN (Uy et al., 2019), ModelNet + CO3D (Reizenstein et al., 2021), ShapeNet (Chang et al., 2015) + OmniObject3D (Wu et al., 2023), and ShapeNet + ShapeNet-C (Ren et al., 2022). Training and evaluation follow the Domain3D-CIL protocol in Sec. 3. The two ModelNet combinations use the **B4Inc4** setting, with 4 initial classes and 4 new classes per subsequent task, resulting in 10 tasks. See Appendix B for additional task splits. The two ShapeNet combinations use the **B5Inc5** setting, with 5 initial classes and 5 new classes per subsequent task, resulting in 11 tasks. The dataset composition and semantic class matching are described in Sec. 3, with further implementation detailed in Appendix H. See Appendix B for additional task splits.

Evaluation Metrics. We report performance on two domains, denoted as D_1 and D_2, where D_1 corresponds to the CAD domain and D_2 to the real-world or corrupted domain. Let $a_{t,d}$ denote accuracy after task t on the test samples from observed classes in domain D_d, with predictions made over all observed classes. We adopt **Last Accuracy (LA)**, defined as $\mathrm{LA}_{D_d} = a_{T,d}$, and **Cumulative Accuracy (CA)**, defined as $\mathrm{CA}_{D_d} = \frac{1}{T}\sum_{t=1}^{T} a_{t,d}$. For each metric, we report the results on D_1 and D_2, together with the equally weighted average across domains, $\mathrm{Avg.} = \frac{1}{K}\sum_{d=1}^{K} \mathrm{Metric}_{D_d}$, where K is the number of domains and $\mathrm{Metric} \in \{\mathrm{LA}, \mathrm{CA}\}$. All reported results are averaged over 3 runs.

5.2 Main Results

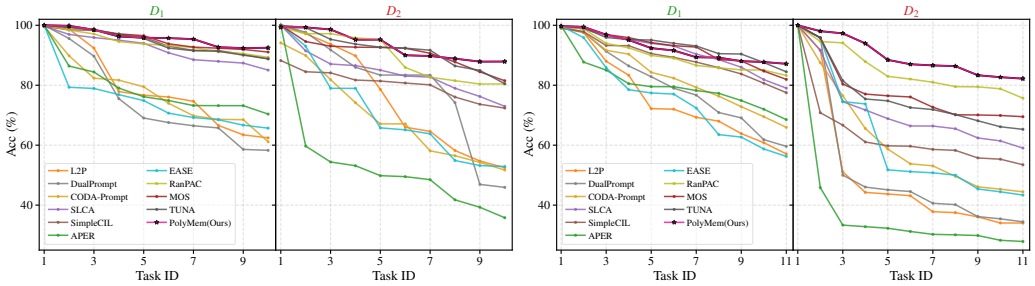
Table 1: Quantitative evaluation of PTM-based CIL baselines and PolyMem on Domain3D-CIL. Avg. denotes the equally weighted average of D_1 and D_2 accuracies. The best and second-best results are highlighted in **bold** and underlined, respectively. See Appendix A for standard deviations.

Method	ModelNet (D_1) + ScanObjectNN (D_2)						ModelNet (D_1) + CO3D (D_2)					
	Last Acc.			Cum. Acc.			Last Acc.			Cum. Acc.		
	$D_1\uparrow$	$D_2\uparrow$	Avg.$\uparrow$	$D_1\uparrow$	$D_2\uparrow$	Avg.$\uparrow$	$D_1\uparrow$	$D_2\uparrow$	Avg.$\uparrow$	$D_1\uparrow$	$D_2\uparrow$	Avg.$\uparrow$
L2P (Wang et al., 2022d)	62.44	52.49	57.47	78.49	72.54	75.52	52.03	24.63	38.33	68.21	53.16	60.69
DualPrompt (Wang et al., 2022c)	58.59	45.92	52.26	78.88	74.65	76.77	52.07	23.13	37.60	72.07	54.93	63.50
CODA-Prompt (Smith et al., 2023)	61.14	51.69	56.42	77.51	69.49	73.50	58.43	36.51	47.47	72.63	55.20	63.92
SLCA (Zhang et al., 2023)	85.05	72.96	79.01	92.13	84.41	88.27	80.88	58.41	69.65	89.81	77.23	83.52
SimpleCIL (Zhou et al., 2025a)	88.82	72.37	80.60	94.48	80.29	87.39	83.14	58.87	71.01	90.93	75.16	83.05
APER (Zhou et al., 2025a)	70.38	35.79	53.09	79.09	53.19	66.14	55.31	43.40	49.36	72.35	64.29	68.32
EASE (Zhou et al., 2024b)	65.68	52.88	59.28	75.07	70.65	72.86	49.39	31.97	40.68	65.64	55.05	60.35
RanPAC (McDonnell et al., 2023)	89.30	80.52	84.91	94.07	88.70	91.39	90.34	81.32	85.83	93.93	85.37	89.65
MOS (Sun et al., 2025)	91.07	81.50	86.29	95.38	91.06	93.22	88.01	76.51	82.26	**94.38**	87.00	90.69
TUNA (Wang et al., 2025)	88.82	80.52	84.67	94.46	91.69	93.08	87.24	66.10	76.67	93.41	79.74	86.58
PolyMem (Ours)	**92.50**	**87.90**	**90.20**	**95.90**	**93.41**	**94.66**	**90.67**	**84.35**	**87.51**	94.29	**88.56**	**91.43**

Method	ShapeNet (D_1) + OmniObject3D (D_2)						ShapeNet (D_1) + ShapeNet-C (D_2)					
	Last Acc.			Cum. Acc.			Last Acc.			Cum. Acc.		
	$D_1\uparrow$	$D_2\uparrow$	Avg.$\uparrow$	$D_1\uparrow$	$D_2\uparrow$	Avg.$\uparrow$	$D_1\uparrow$	$D_2\uparrow$	Avg.$\uparrow$	$D_1\uparrow$	$D_2\uparrow$	Avg.$\uparrow$
L2P (Wang et al., 2022d)	57.08	34.03	45.56	75.58	50.31	62.95	61.96	50.93	56.45	80.22	72.06	76.14
DualPrompt (Wang et al., 2022c)	59.61	35.42	47.52	79.54	51.67	65.61	62.69	52.37	57.53	77.15	68.33	72.74
CODA-Prompt (Smith et al., 2023)	65.96	44.44	55.20	82.71	61.87	72.29	69.28	56.98	63.13	84.06	74.72	79.39
SLCA (Zhang et al., 2023)	79.03	59.03	69.03	91.19	71.64	81.42	78.20	63.35	70.78	91.18	82.00	86.59
SimpleCIL (Zhou et al., 2025a)	77.52	53.47	65.50	88.98	63.59	76.29	76.60	69.34	72.97	88.50	82.76	85.63
APER (Zhou et al., 2025a)	68.52	29.86	49.19	80.28	38.34	59.31	66.75	59.29	63.02	79.11	72.31	75.71
EASE (Zhou et al., 2024b)	58.74	44.44	51.59	75.29	61.92	68.61	68.82	52.07	60.45	84.79	69.73	77.26
RanPAC (McDonnell et al., 2023)	83.26	74.69	78.98	89.98	85.10	87.54	76.33	69.70	73.02	90.06	84.34	87.20
MOS (Sun et al., 2025)	81.95	70.14	76.05	92.29	78.01	85.15	82.16	72.29	77.23	92.40	86.53	89.47
TUNA (Wang et al., 2025)	84.48	65.28	74.88	**93.20**	76.53	84.87	83.13	75.36	79.25	92.81	88.16	90.49
PolyMem (Ours)	**87.15**	**83.52**	**85.34**	92.42	**89.63**	**91.03**	**83.72**	**77.85**	**80.79**	**93.21**	**89.45**	**91.33**

Tab. 1 presents the results on Domain3D-CIL. PolyMem achieves the highest average LA and CA across domains on all four dataset combinations. Its average LA reaches 90.20%, 87.51%, 85.34%, and 80.79% on ModelNet + ScanObjectNN, ModelNet + CO3D, ShapeNet + OmniObject3D, and ShapeNet + ShapeNet-C, respectively. The consistent improvements in average CA further demonstrate strong performance across incremental stages, beyond the final task.

For per-domain performance, PolyMem achieves the highest LA and CA on D_2 across all four combinations. Compared with the strongest baseline on each dataset, PolyMem improves LA by



(a) ModelNet (D_1) + ScanObjectNN (D_2) (b) ShapeNet (D_1) + OmniObject3D (D_2)

Figure 4: Task-wise accuracy on the CAD (D_1) and real-world (D_2) domains.

6.40, 3.03, and 8.83 percentage points on ScanObjectNN, CO3D, and OmniObject3D, respectively, and by 2.49 percentage points on ShapeNet-C. Meanwhile, it achieves the best LA and the best or second-best CA on the CAD domain in all four combinations. The task-wise accuracy curves in Fig. 4a and Fig. 4b provide further insight into these results. In particular, on the real-world domain in Fig. 4a, several competitive baselines perform well during early tasks but decline as more classes are introduced, while PolyMem maintains higher accuracy in later stages. This pattern explains the more pronounced improvement in LA than in CA: CA averages over all tasks, moderating later-stage gains with smaller early-stage differences, whereas LA directly reflects final-stage performance.

5.3 Ablation Studies

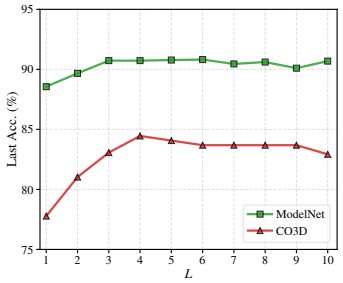
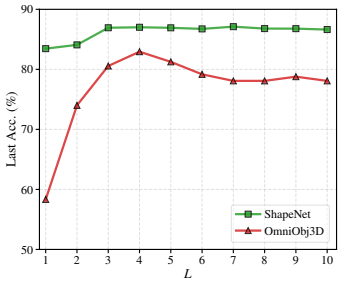


Figure 5: Ablation on L



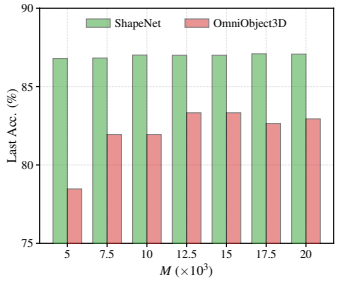


Figure 6: Ablation on M

Interaction Order L. In the main experiments, PolyMem uses $L = 4$ and $M = 10000$, with M matching RanPAC. Fig. 5 examines the effect of the interaction order L on two mixed datasets with $M = 10000$ fixed. Accuracy improves on both domains as L increases from 1 to 4. Specifically, on ShapeNet + OmniObject3D, the $L = 2$ variant achieves LA of 84.08% and 74.00% on the CAD and real-world domains, respectively, close to RanPAC's 83.26% and 74.69% in Tab. 1. Increasing L from 2 to 4 improves LA by 3.07% and 9.52% on the two domains. These results indicate that, under the same total dimensional budget, a structured low-order mapping alone does not deliver the full performance gains, while incorporating third- and fourth-order interactions further improves performance. When $L > 4$, performance on the CAD domain saturates, while accuracy on the real-world domain declines. Under a fixed total dimensionality, increasing the order reduces the sketch dimensions allocated to each order, which may limit the benefits of additional higher-order interactions. Based on the results on these two datasets, we adopt $L = 4$ to balance interaction order and per-order representation dimensionality.

Mapping Dimension M. Fig. 6 examines the effect of the mapping dimension M with $L = 4$ fixed. At $M = 5000$, accuracy on the real-world domain is relatively low. Increasing M to 10000 substantially improves real-world accuracy, while CAD accuracy changes only slightly. With a fixed interaction order, increasing M allocates more sketch dimensions to each order, providing greater representational capacity for polynomial interactions. Further increases in M yield broadly stable performance on both domains, with some fluctuations in real-world accuracy rather than consistent gains. These results suggest that a moderate mapping dimension is sufficient for strong performance. Balancing accuracy and statistical-memory storage costs, we adopt $M = 10000$ in the main experiments, matching the mapping dimension of RanPAC.

Different Backbones. To evaluate whether PolyMem's improvements rely on a strong pre-trained backbone, we replace Uni3D with Point-BERT (Yu et al., 2022) and Point-MAE (Pang et al., 2023) and evaluate on ModelNet + ScanObjectNN and ModelNet + CO3D. These two backbones are self-supervised pre-trained only on the CAD dataset ShapeNet. We compare PolyMem with SimpleCIL and RanPAC using the same backbone within each setting, where PolyMem and RanPAC differ only in the feature mapping used to construct statistical memory. As shown in Tab. 2, PolyMem outperforms RanPAC in LA and CA on each domain across backbones and dataset combinations, yielding higher average accuracy across domains. These results indicate that PolyMem's improvements stem from its enhanced statistical memory and do not rely on a strong pre-trained backbone.

Table 2: Performance with different pre-trained backbones on Domain3D-CIL.

Method	ModelNet (D_1) + ScanObjectNN (D_2)						ModelNet (D_1) + CO3D (D_2)					
	Last Acc.			Cum. Acc.			Last Acc.			Cum. Acc.		
	D_1↑	D_2↑	Avg.↑	D_1↑	D_2↑	Avg.↑	D_1↑	D_2↑	Avg.↑	D_1↑	D_2↑	Avg.↑
Point-BERT Backbone												
SimpleCIL (Zhou et al., 2025a)	90.64	37.97	64.31	94.82	44.33	69.58	89.26	25.94	57.60	93.88	33.31	63.60
RanPAC (McDonnell et al., 2023)	90.93	66.38	78.66	94.79	79.14	86.97	90.24	65.07	77.66	93.04	71.26	82.15
PolyMem (Ours)	**91.00**	**73.96**	**82.48**	**95.02**	**83.04**	**89.03**	**91.53**	**70.39**	**80.96**	**94.11**	**75.06**	**84.59**
Point-MAE Backbone												
SimpleCIL (Zhou et al., 2025a)	77.48	65.23	71.36	87.91	77.91	82.91	62.44	39.23	50.84	71.62	48.95	60.29
RanPAC (McDonnell et al., 2023)	88.19	77.93	83.06	92.66	82.29	87.48	88.82	73.67	81.25	92.71	78.35	85.53
PolyMem (Ours)	**90.60**	**83.30**	**86.95**	**94.54**	**87.72**	**91.13**	**89.06**	**76.87**	**82.97**	**93.83**	**81.07**	**87.45**

Number of Domains. We additionally evaluate single-domain ($K = 1$) and three-domain ($K = 3$) extensions of the main two-domain protocol. As shown in Tab. 3, PolyMem achieves the highest LA and CA among the compared methods on both ModelNet and ShapeNet under single-domain CIL, indicating that its benefits are not limited to cross-domain mixing. When mixing ShapeNet, OmniObject3D, and ShapeNet-C, PolyMem achieves the highest LA on all three domains and the highest domain-averaged LA and CA among the compared methods. These results show that its advantages extend to mixtures involving more domains.

Table 3: Performance under different numbers of domains. $K = 1$: single-source CIL on ModelNet or ShapeNet. $K = 3$: ShapeNet (D_1), OmniObject3D (D_2), and ShapeNet-C (D_3) are mixed within shared semantic classes. Avg. denotes the equally weighted average across the 3 domains.

Method	$K = 1$				$K = 3$							
	ModelNet		ShapeNet		Last Acc.				Cum. Acc.			
	LA↑	CA↑	LA↑	CA↑	D_1↑	D_2↑	D_3↑	Avg.↑	D_1↑	D_2↑	D_3↑	Avg.↑
SimpleCIL (Zhou et al., 2025a)	88.95	94.52	77.30	89.12	76.91	54.17	70.22	67.10	88.63	64.58	83.06	78.76
RanPAC (McDonnell et al., 2023)	89.62	94.21	83.04	90.21	82.57	75.69	71.56	76.61	89.55	85.39	84.76	86.57
MOS (Sun et al., 2025)	91.12	95.41	83.53	92.35	81.60	69.52	71.83	74.32	91.94	77.46	86.10	85.17
TUNA (Wang et al., 2025)	89.05	94.50	84.21	92.57	83.97	64.71	74.88	74.52	**92.86**	75.92	87.70	85.49
PolyMem (Ours)	**92.31**	**95.74**	**87.15**	**93.08**	**86.64**	**82.64**	**78.61**	**82.63**	92.01	**91.08**	**89.08**	**90.72**

6 Conclusion

3D point clouds can originate from CAD models, real-world scans, or video reconstructions, with differences in geometric completeness, sampling density, and noise characteristics. As data from multiple sources are continually collected to learn new classes, samples within the same class may be heterogeneous, while acquisition and source information may not be fully preserved. Our setting of mixed-domain learning without training-time domain labels aims to reflect this practical scenario.

To this end, we establish Domain3D-CIL and adapt mainstream CIL methods to the 3D modality for systematic evaluation. We further propose PolyMem, which incorporates higher-order feature interactions into exemplar-free statistical memory through a structured polynomial mapping. Experiments demonstrate improved average accuracy across domains, with notable gains on real-world data, and consistent effectiveness across different backbones, domain counts, and task splits.

Limitation and Future Work Our experiments align semantic classes and mix source datasets using their existing sample sizes, without systematically evaluating the effects of varying domain mixing ratios. Future work will investigate robustness under different mixing ratios and explore this approach for heterogeneous data in other modalities.

AI use statement Generative AI tools were used to provide assistance with literature search, language polishing, manuscript formatting, and coding. For literature search assistance, the authors independently reviewed and verified all relevant sources and did not rely on AI-generated summaries. For coding assistance, the authors reviewed, tested, and validated all AI-assisted code and take full responsibility for its correctness and use. Generative AI was not used for developing the motivation, generating research ideas, or formulating the proposed method. The authors carefully reviewed all AI-assisted outputs and take full responsibility for the accuracy, integrity, conclusions, and final content of this work.

Ethics statement This work focuses on improving the adaptability of 3D perception models under heterogeneous data distributions. All experiments are conducted using existing publicly available datasets. Our work does not involve the collection of new human-subject data and no human subjects are directly involved in this research. While the proposed method does not introduce direct societal risks, its deployment in safety-critical environments requires further validation of robustness and reliability. The use of real-world 3D data may also involve privacy and data provenance considerations. We encourage responsible data usage and appropriate evaluation before deploying the proposed method in real-world applications.

Reproducibility statement We provide detailed descriptions of the proposed method in the main text, including the polynomial feature mapping, statistical updates, and closed-form classifier in Sec. 4. We also document the implementation details, training configurations, and hyperparameter settings in Sec. 5.1 and Appendix I. The experimental results are reported under a consistent evaluation protocol to enable systematic comparison across heterogeneous 3D domains. The supplementary material includes demonstration code for the 3D CIL setup. As this work is currently under review, we will release our code for the proposed method and evaluation upon publication.

REFERENCES

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.

Qi Chen, Sihai Tang, Qing Yang, and Song Fu. Cooper: Cooperative perception for connected autonomous vehicles based on 3d point clouds. In *2019 IEEE 39th International Conference on distributed computing systems (ICDCS)*, pp. 514–524. IEEE, 2019.

Ali Cheraghian, Zeeshan Hayder, Sameera Ramasinghe, Shafin Rahman, Javad Jafaryahya, Lars Petersson, and Mehrtash Harandi. Canonical shape projection is all you need for 3d few-shot class incremental learning. In *European Conference on Computer Vision*, pp. 36–53. Springer, 2024.

Townim Chowdhury, Ali Cheraghian, Sameera Ramasinghe, Sahar Ahmadi, Morteza Saberi, and Shafin Rahman. Few-shot class-incremental learning for 3d point cloud objects. In *European Conference on Computer Vision*, pp. 204–220. Springer, 2022.

Yin Cui, Feng Zhou, Jiang Wang, Xiao Liu, Yuanqing Lin, and Serge Belongie. Kernel pooling for convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2921–2930, 2017.

Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5828–5839, 2017.

Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 36:35799–35813, 2023.

Jiahua Dong, Yang Cong, Gan Sun, Bingtao Ma, and Lichen Wang. I3dol: Incremental 3d object learning without catastrophic forgetting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 6066–6074, 2021.

Jiahua Dong, Yang Cong, Gan Sun, Lixu Wang, Lingjuan Lyu, Jun Li, and Ender Konukoglu. Inor-net: Incremental 3-d object recognition network for point cloud representation. *IEEE Transactions on Neural Networks and Learning Systems*, 34(10):6955–6967, 2023.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *European conference on computer vision*, pp. 86–102. Springer, 2020.

Tom Fischer, Yaoyao Liu, Artur Jesslen, Noor Ahmed, Prakhar Kaushik, Angtian Wang, Alan L Yuille, Adam Kortylewski, and Eddy Ilg. inemo: Incremental neural mesh models for robust class-incremental learning. In *European conference on computer vision*, pp. 357–374. Springer, 2024.

Qiankun Gao, Chen Zhao, Yifan Sun, Teng Xi, Gang Zhang, Bernard Ghanem, and Jian Zhang. A unified continual learning framework with general parameter-efficient tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11483–11493, 2023.

Dipam Goswami, Yuyang Liu, Bartłomiej Twardowski, and Joost Van De Weijer. Fecam: Exploiting the heterogeneity of class distributions in exemplar-free continual learning. *Advances in Neural Information Processing Systems*, 36:6582–6595, 2023.

Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 831–839, 2019.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12697–12705, 2019.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

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. *Advances in neural information processing systems*, 36:44860–44879, 2023.

Jinge Ma and Fengqing Zhu. Temporal imbalance of positive and negative supervision in class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

Mark D McDonnell, Dong Gong, Amin Parvaneh, Ehsan Abbasnejad, and Anton Van den Hengel. Ranpac: Random projections and pre-trained models for continual learning. *Advances in Neural Information Processing Systems*, 36:12022–12053, 2023.

Zichen Miao, Ze Wang, Wei Chen, and Qiang Qiu. Continual learning with filter atom swapping. In *International Conference on Learning Representations*, 2021.

Yatian Pang, Eng Hock Francis Tay, Li Yuan, and Zhenghua Chen. Masked autoencoders for 3d point cloud self-supervised learning. *World Scientific Annual Review of Artificial Intelligence*, 1:2440001, 2023.

Min-Yeong Park, Jae-Ho Lee, and Gyeong-Moon Park. Versatile incremental learning: Towards class and domain-agnostic incremental learning. In *European Conference on Computer Vision*, pp. 271–288. Springer, 2024.

Karl Pearson. Contributions to the mathematical theory of evolution. ii. skew variation in homogeneous material. *Philosophical Transactions of the Royal Society of London. A*, 186:343–414, 1895. doi: 10.1098/rsta.1895.0010.

Karl Pearson. "Das Fehlergesetz und Seine Verallgemeinerungen durch Fechner und Pearson." A Rejoinder. *Biometrika*, 4(1–2):169–212, 1905. doi: 10.1093/biomet/4.1-2.169.

Ninh Pham and Rasmus Pagh. Fast and scalable polynomial kernels via explicit feature maps. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 239–247, 2013.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017a.

Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9277–9286, 2019.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017b.

Can Qin, Haoxuan You, Lichen Wang, C-C Jay Kuo, and Yun Fu. Pointdan: A multi-scale 3d domain adaption network for point cloud representation. *Advances in Neural Information Processing Systems*, 32, 2019.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10901–10911, 2021.

Jiawei Ren, Liang Pan, and Ziwei Liu. Benchmarking and analyzing point cloud classification under corruptions. In *International Conference on Machine Learning*, pp. 18559–18575. PMLR, 2022.

Hossein Resani and Behrooz Nasihatkon. Miracle 3d: Memory-efficient integrated robust approach for continual learning on 3d point clouds via shape model construction. In *International Conference on Learning Representations*, volume 2025, pp. 59974–59987, 2025.

Hossein Resani, Behrooz Nasihatkon, and Mohammadreza Alimoradi Jazi. Continual learning in 3d point clouds: Employing spectral techniques for exemplar selection. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 2921–2931. IEEE, 2025.

Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11909–11919, 2023.

Hai-Long Sun, Da-Wei Zhou, Hanbin Zhao, Le Gan, De-Chuan Zhan, and Han-Jia Ye. Mos: Model surgery for pre-trained model-based class-incremental learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 20699–20707, 2025.

Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2446–2454, 2020.

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 *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1588–1597, 2019.

Fu-Yun Wang, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Foster: Feature boosting and compression for class-incremental learning. In *European conference on computer vision*, pp. 398–414. Springer, 2022a.

Qiang Wang, Yuhang He, Songlin Dong, Xinyuan Gao, Shaokun Wang, and Yihong Gong. Non-exemplar domain incremental learning via cross-domain concept integration. In *European Conference on Computer Vision*, pp. 144–162. Springer, 2024.

Yabin Wang, Zhiwu Huang, and Xiaopeng Hong. S-prompts learning with pre-trained transformers: An occam's razor for domain incremental learning. *Advances in Neural Information Processing Systems*, 35:5682–5695, 2022b.

Yan Wang, Da-Wei Zhou, and Han-Jia Ye. Integrating task-specific and universal adapters for pre-trained model-based class-incremental learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 806–816, 2025.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *European conference on computer vision*, pp. 631–648. Springer, 2022c.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 139–149, 2022d.

Tong Wu, Jiarui Zhang, Xiao Fu, Yuxin Wang, Jiawei Ren, Liang Pan, Wayne Wu, Lei Yang, Jiaqi Wang, Chen Qian, et al. Omniobject3d: Large-vocabulary 3d object dataset for realistic perception, reconstruction and generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 803–814, 2023.

Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. Point transformer v2: Grouped vector attention and partition-based pooling. *Advances in Neural Information Processing Systems*, 35:33330–33342, 2022.

Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler faster stronger. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4840–4851, 2024.

Xiaoyang Wu, Daniel DeTone, Duncan Frost, Tianwei Shen, Chris Xie, Nan Yang, Jakob Engel, Richard Newcombe, Hengshuang Zhao, and Julian Straub. Sonata: Self-supervised learning of reliable point representations. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 22193–22204, 2025.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 374–382, 2019.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920, 2015.

Tuo Xiang, Xuemiao Xu, Bangzhen Liu, Jinyi Li, Yong Li, and Shengfeng He. Seeing 3d through 2d lenses: 3d few-shot class-incremental learning via cross-modal geometric rectification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6761–6771, 2025.

Jiangwei Xie, Shipeng Yan, and Xuming He. General incremental learning with domain-aware categorical representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14351–14360, 2022.

Le Xue, Mingfei Gao, Chen Xing, Roberto Martín-Martín, Jiajun Wu, Caiming Xiong, Ran Xu, Juan Carlos Niebles, and Silvio Savarese. Ulip: Learning a unified representation of language, images, and point clouds for 3d understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1179–1189, 2023.

Le Xue, Ning Yu, Shu Zhang, Artemis Panagopoulou, Junnan Li, Roberto Martín-Martín, Jiajun Wu, Caiming Xiong, Ran Xu, Juan Carlos Niebles, et al. Ulip-2: Towards scalable multimodal pre-training for 3d understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 27091–27101, 2024.

Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3014–3023, 2021.

Xianggang Yu, Mutian Xu, Yidan Zhang, Haolin Liu, Chongjie Ye, Yushuang Wu, Zizheng Yan, Chenming Zhu, Zhangyang Xiong, Tianyou Liang, et al. Mvimgnet: A large-scale dataset of multi-view images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9150–9161, 2023.

Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 19313–19322, 2022.

Gengwei Zhang, Liyuan Wang, Guoliang Kang, Ling Chen, and Yunchao Wei. Slca: Slow learner with classifier alignment for continual learning on a pre-trained model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19148–19158, 2023.

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. *Advances in neural information processing systems*, 35:27061–27074, 2022.

Yujia Zhang, Xiaoyang Wu, Yixing Lao, Chengyao Wang, Zhuotao Tian, Naiyan Wang, and Hengshuang Zhao. Concerto: Joint 2d-3d self-supervised learning emerges spatial representations. In *Advances in Neural Information Processing Systems*, 2025.

Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. Maintaining discrimination and fairness in class incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13208–13217, 2020.

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 16259–16268, 2021.

Da-Wei Zhou, Qi-Wei Wang, Han-Jia Ye, and De-Chuan Zhan. A model or 603 exemplars: Towards memory-efficient class-incremental learning. In *International Conference on Learning Representations*, 2023a.

Da-Wei Zhou, Hai-Long Sun, Jingyi Ning, Han-Jia Ye, and De-Chuan Zhan. Continual learning with pre-trained models: A survey. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pp. 8363–8371, 2024a. doi: 10.24963/ijcai.2024/924.

Da-Wei Zhou, Hai-Long Sun, Han-Jia Ye, and De-Chuan Zhan. Expandable subspace ensemble for pre-trained model-based class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 23554–23564, 2024b.

Da-Wei Zhou, Zi-Wen Cai, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Revisiting class-incremental learning with pre-trained models: Generalizability and adaptivity are all you need. *International Journal of Computer Vision*, 133(3):1012–1032, 2025a.

Da-Wei Zhou, Zi-Wen Cai, Han-Jia Ye, Lijun Zhang, and De-Chuan Zhan. Dual consolidation for pre-trained model-based domain-incremental learning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 20547–20557, 2025b.

Junsheng Zhou, Jinsheng Wang, Baorui Ma, Yu-Shen Liu, Tiejun Huang, and Xinlong Wang. Uni3d: Exploring unified 3d representation at scale. In *The Twelfth International Conference on Learning Representations*, 2023b.

The appendix is organized as follows. Sec. A provides detailed experimental results, including run-to-run standard deviations and additional task-wise accuracy curves. Sec. B evaluates PolyMem under alternative class-incremental task splits. Sec. C analyzes learnable parameters, auxiliary storage, and training and inference time. Sec. E describes the computational environment. Sec. D reports the performance of classic CIL baselines trained from scratch on Domain3D-CIL. Secs. F and G provide the theoretical background and algorithmic description of PolyMem. Finally, Sec. H details dataset construction, preprocessing, and semantic class alignment, and Sec. I explains the adaptation of baseline implementations to 3D point clouds.

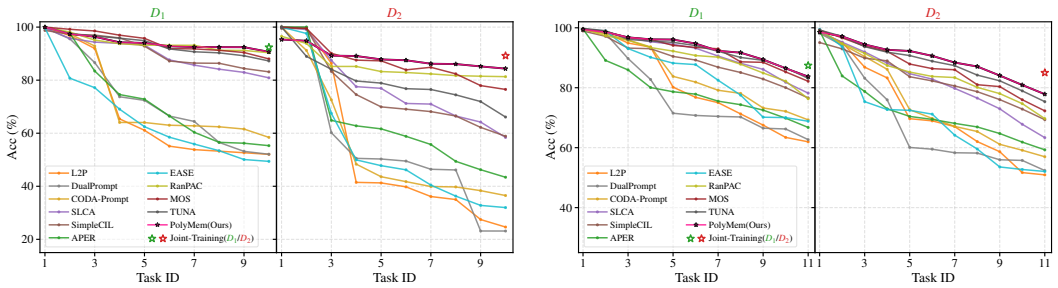
A PTM-BASED CIL EXPERIMENT DETAILS

Detailed Results. We report the mean and run-to-run standard deviation of the per-domain results over the same three runs used in Tab. 1. Results are organized by mixed benchmark: ModelNet + ScanObjectNN in Tab. 4, ModelNet + CO3D in Tab. 5, ShapeNet + OmniObject3D in Tab. 6, and ShapeNet + ShapeNet-C in Tab. 7. For both Last Accuracy (LA) and Cumulative Accuracy (CA), Avg. reports the equally weighted mean across the two domains. Joint-training results are included for completeness, with evaluation on the same test sets used for final-task accuracy. The best and second-best results among CIL methods are highlighted in **bold** and underlined, respectively.

Table 4: Detailed results on ModelNet + ScanObjectNN (%).

Method	Last Acc.			Cum. Acc.		
	$D_1\uparrow$	$D_2\uparrow$	Avg.↑	$D_1\uparrow$	$D_2\uparrow$	Avg.↑
Joint-Training	93.68	91.48	92.58	–	–	–
L2P (Wang et al., 2022d)	$62.44_{\pm 0.69}$	$52.49_{\pm 1.06}$	$57.47_{\pm 0.63}$	$78.49_{\pm 0.33}$	$72.54_{\pm 0.82}$	$75.52_{\pm 0.44}$
DualPrompt (Wang et al., 2022c)	$58.59_{\pm 0.42}$	$45.92_{\pm 1.24}$	$52.26_{\pm 0.65}$	$78.88_{\pm 0.61}$	$74.65_{\pm 0.70}$	$76.77_{\pm 0.46}$
CODA-Prompt (Smith et al., 2023)	$61.14_{\pm 0.83}$	$51.69_{\pm 0.57}$	$56.42_{\pm 0.50}$	$77.51_{\pm 0.29}$	$69.49_{\pm 0.96}$	$73.50_{\pm 0.50}$
SLCA (Zhang et al., 2023)	$85.05_{\pm 0.44}$	$72.96_{\pm 0.72}$	$79.01_{\pm 0.42}$	$92.13_{\pm 0.18}$	$84.41_{\pm 0.69}$	$88.27_{\pm 0.36}$
SimpleCIL (Zhou et al., 2025a)	$88.82_{\pm 0.31}$	$72.37_{\pm 1.10}$	$80.60_{\pm 0.57}$	$94.48_{\pm 0.26}$	$80.29_{\pm 0.73}$	$87.39_{\pm 0.39}$
APER (Zhou et al., 2025a)	$70.38_{\pm 0.92}$	$35.79_{\pm 1.38}$	$53.09_{\pm 0.83}$	$79.09_{\pm 0.58}$	$53.19_{\pm 1.21}$	$66.14_{\pm 0.67}$
EASE (Zhou et al., 2024b)	$65.68_{\pm 0.47}$	$52.88_{\pm 0.91}$	$59.28_{\pm 0.51}$	$75.07_{\pm 0.73}$	$70.65_{\pm 0.44}$	$72.86_{\pm 0.43}$
RanPAC (McDonnell et al., 2023)	$89.30_{\pm 0.29}$	$80.52_{\pm 0.67}$	$84.91_{\pm 0.37}$	$94.07_{\pm 0.12}$	$88.70_{\pm 0.51}$	$91.39_{\pm 0.26}$
MOS (Sun et al., 2025)	$\underline{91.07}_{\pm 0.26}$	$\underline{81.50}_{\pm 0.41}$	$\underline{86.29}_{\pm 0.24}$	$\underline{95.38}_{\pm 0.21}$	$91.06_{\pm 0.37}$	$\underline{93.22}_{\pm 0.21}$
TUNA (Wang et al., 2025)	$88.82_{\pm 0.18}$	$80.52_{\pm 0.74}$	$84.67_{\pm 0.38}$	$94.46_{\pm 0.27}$	$\underline{91.69}_{\pm 0.46}$	$93.08_{\pm 0.27}$
PolyMem (Ours)	$\mathbf{92.50}_{\pm 0.27}$	$\mathbf{87.90}_{\pm 0.55}$	$\mathbf{90.20}_{\pm 0.31}$	$\mathbf{95.90}_{\pm 0.17}$	$\mathbf{93.41}_{\pm 0.42}$	$\mathbf{94.66}_{\pm 0.23}$

Additional Task-wise Accuracy Curves. The main paper reports task-wise accuracy curves for ModelNet + ScanObjectNN and ShapeNet + OmniObject3D in Fig. 4a and Fig. 4b. Here, Fig. 7a and Fig. 7b provide the corresponding curves for ModelNet + CO3D and ShapeNet + ShapeNet-C. Each curve reports accuracy on the test samples from all observed classes in the corresponding domain, with predictions made over all observed classes.



(a) ModelNet (D_1) + CO3D (D_2) (b) ShapeNet (D_1) + ShapeNet-C (D_2)

Figure 7: Task-wise accuracy on two additional Domain3D-CIL benchmarks. Accuracy is evaluated separately on each domain after every incremental task.

Overall Performance. Fig. 8 summarizes the final-task performance of PTM-based CIL methods across the four mixed benchmarks. For each method, we first average LA equally across domains

Table 5: Detailed results on ModelNet + CO3D (%).

Method	Last Acc.			Cum. Acc.		
	D_1↑	D_2↑	Avg.↑	D_1↑	D_2↑	Avg.↑
Joint-Training	92.88	89.21	91.05	–	–	–
L2P (Wang et al., 2022d)	52.03$_{\pm 1.18}$	24.63$_{\pm 2.04}$	38.33$_{\pm 1.18}$	68.21$_{\pm 0.77}$	53.16$_{\pm 1.41}$	60.69$_{\pm 0.80}$
DualPrompt (Wang et al., 2022c)	52.07$_{\pm 0.96}$	23.13$_{\pm 1.83}$	37.60$_{\pm 1.03}$	72.07$_{\pm 0.49}$	54.93$_{\pm 1.28}$	63.50$_{\pm 0.69}$
CODA-Prompt (Smith et al., 2023)	58.43$_{\pm 0.74}$	36.51$_{\pm 1.62}$	47.47$_{\pm 0.89}$	72.63$_{\pm 0.67}$	55.20$_{\pm 1.05}$	63.92$_{\pm 0.62}$
SLCA (Zhang et al., 2023)	80.88$_{\pm 0.56}$	58.41$_{\pm 1.14}$	69.65$_{\pm 0.64}$	89.81$_{\pm 0.36}$	77.23$_{\pm 0.93}$	83.52$_{\pm 0.50}$
SimpleCIL (Zhou et al., 2025a)	83.14$_{\pm 0.47}$	58.87$_{\pm 1.36}$	71.01$_{\pm 0.72}$	90.93$_{\pm 0.41}$	75.16$_{\pm 0.82}$	83.05$_{\pm 0.46}$
APER (Zhou et al., 2025a)	55.31$_{\pm 1.09}$	43.40$_{\pm 1.21}$	49.36$_{\pm 0.81}$	72.35$_{\pm 0.70}$	64.29$_{\pm 1.02}$	68.32$_{\pm 0.62}$
EASE (Zhou et al., 2024b)	49.39$_{\pm 0.84}$	31.97$_{\pm 1.69}$	40.68$_{\pm 0.94}$	65.64$_{\pm 0.91}$	55.05$_{\pm 1.13}$	60.35$_{\pm 0.73}$
RanPAC (McDonnell et al., 2023)	90.34$_{\pm 0.35}$	81.32$_{\pm 0.79}$	85.83$_{\pm 0.43}$	93.93$_{\pm 0.19}$	85.37$_{\pm 0.64}$	89.65$_{\pm 0.33}$
MOS (Sun et al., 2025)	88.01$_{\pm 0.28}$	76.51$_{\pm 0.66}$	82.26$_{\pm 0.36}$	**94.38**$_{\pm 0.24}$	87.00$_{\pm 0.49}$	90.69$_{\pm 0.27}$
TUNA (Wang et al., 2025)	87.24$_{\pm 0.52}$	66.10$_{\pm 1.07}$	76.67$_{\pm 0.59}$	93.41$_{\pm 0.31}$	79.74$_{\pm 0.88}$	86.58$_{\pm 0.47}$
PolyMem (Ours)	**90.67**$_{\pm 0.31}$	**84.35**$_{\pm 0.68}$	**87.51**$_{\pm 0.37}$	94.29$_{\pm 0.22}$	**88.56**$_{\pm 0.46}$	**91.43**$_{\pm 0.25}$

Table 6: Detailed results on ShapeNet + OmniObject3D (%).

Method	Last Acc.			Cum. Acc.		
	D_1↑	D_2↑	Avg.↑	D_1↑	D_2↑	Avg.↑
Joint-Training	88.47	86.11	87.29	–	–	–
L2P (Wang et al., 2022d)	57.08$_{\pm 1.31}$	34.03$_{\pm 2.24}$	45.56$_{\pm 1.30}$	75.58$_{\pm 0.86}$	50.31$_{\pm 1.87}$	62.95$_{\pm 1.03}$
DualPrompt (Wang et al., 2022c)	59.61$_{\pm 1.05}$	35.42$_{\pm 2.03}$	47.52$_{\pm 1.14}$	79.54$_{\pm 0.79}$	51.67$_{\pm 1.61}$	65.61$_{\pm 0.90}$
CODA-Prompt (Smith et al., 2023)	65.96$_{\pm 0.91}$	44.44$_{\pm 1.74}$	55.20$_{\pm 0.98}$	82.71$_{\pm 0.69}$	61.87$_{\pm 1.30}$	72.29$_{\pm 0.74}$
SLCA (Zhang et al., 2023)	79.03$_{\pm 0.64}$	59.03$_{\pm 1.47}$	69.03$_{\pm 0.80}$	91.19$_{\pm 0.44}$	71.64$_{\pm 1.12}$	81.42$_{\pm 0.60}$
SimpleCIL (Zhou et al., 2025a)	77.52$_{\pm 0.78}$	53.47$_{\pm 1.62}$	65.50$_{\pm 0.90}$	88.98$_{\pm 0.52}$	63.59$_{\pm 1.09}$	76.29$_{\pm 0.60}$
APER (Zhou et al., 2025a)	68.52$_{\pm 1.26}$	29.86$_{\pm 2.58}$	49.19$_{\pm 1.44}$	80.28$_{\pm 0.94}$	38.34$_{\pm 2.14}$	59.31$_{\pm 1.17}$
EASE (Zhou et al., 2024b)	58.74$_{\pm 0.87}$	44.44$_{\pm 1.52}$	51.59$_{\pm 0.88}$	75.29$_{\pm 0.74}$	61.92$_{\pm 1.24}$	68.61$_{\pm 0.72}$
RanPAC (McDonnell et al., 2023)	83.26$_{\pm 0.48}$	74.69$_{\pm 0.92}$	78.98$_{\pm 0.52}$	89.98$_{\pm 0.33}$	85.10$_{\pm 0.69}$	87.54$_{\pm 0.38}$
MOS (Sun et al., 2025)	81.95$_{\pm 0.37}$	70.14$_{\pm 0.81}$	76.05$_{\pm 0.45}$	92.29$_{\pm 0.29}$	78.01$_{\pm 0.76}$	85.15$_{\pm 0.41}$
TUNA (Wang et al., 2025)	84.48$_{\pm 0.53}$	65.28$_{\pm 1.11}$	74.88$_{\pm 0.62}$	**93.20**$_{\pm 0.34}$	76.53$_{\pm 0.91}$	84.87$_{\pm 0.49}$
PolyMem (Ours)	**87.15**$_{\pm 0.42}$	**83.52**$_{\pm 0.86}$	**85.34**$_{\pm 0.48}$	92.42$_{\pm 0.24}$	**89.63**$_{\pm 0.63}$	**91.03**$_{\pm 0.34}$

Table 7: Detailed results on ShapeNet + ShapeNet-C (%).

Method	Last Acc.			Cum. Acc.		
	D_1↑	D_2↑	Avg.↑	D_1↑	D_2↑	Avg.↑
Joint-Training	87.42	85.01	86.22	–	–	–
L2P (Wang et al., 2022d)	61.96$_{\pm 0.88}$	50.93$_{\pm 1.23}$	56.45$_{\pm 0.76}$	80.22$_{\pm 0.61}$	72.06$_{\pm 0.92}$	76.14$_{\pm 0.55}$
DualPrompt (Wang et al., 2022c)	62.69$_{\pm 0.73}$	52.37$_{\pm 1.06}$	57.53$_{\pm 0.64}$	77.15$_{\pm 0.66}$	68.33$_{\pm 0.83}$	72.74$_{\pm 0.53}$
CODA-Prompt (Smith et al., 2023)	69.28$_{\pm 0.64}$	56.98$_{\pm 0.94}$	63.13$_{\pm 0.57}$	84.06$_{\pm 0.47}$	74.72$_{\pm 0.71}$	79.39$_{\pm 0.43}$
SLCA (Zhang et al., 2023)	78.20$_{\pm 0.52}$	63.35$_{\pm 0.81}$	70.78$_{\pm 0.48}$	91.18$_{\pm 0.36}$	82.00$_{\pm 0.58}$	86.59$_{\pm 0.34}$
SimpleCIL (Zhou et al., 2025a)	76.60$_{\pm 0.39}$	69.34$_{\pm 0.71}$	72.97$_{\pm 0.41}$	88.50$_{\pm 0.31}$	82.76$_{\pm 0.63}$	85.63$_{\pm 0.35}$
APER (Zhou et al., 2025a)	66.75$_{\pm 0.81}$	59.29$_{\pm 1.12}$	63.02$_{\pm 0.69}$	79.11$_{\pm 0.58}$	72.31$_{\pm 0.86}$	75.71$_{\pm 0.52}$
EASE (Zhou et al., 2024b)	68.82$_{\pm 0.79}$	52.07$_{\pm 1.35}$	60.45$_{\pm 0.78}$	84.79$_{\pm 0.69}$	69.73$_{\pm 0.97}$	77.26$_{\pm 0.60}$
RanPAC (McDonnell et al., 2023)	76.33$_{\pm 0.34}$	69.70$_{\pm 0.66}$	73.02$_{\pm 0.37}$	90.06$_{\pm 0.21}$	84.34$_{\pm 0.52}$	87.20$_{\pm 0.28}$
MOS (Sun et al., 2025)	82.16$_{\pm 0.29}$	72.29$_{\pm 0.62}$	77.23$_{\pm 0.34}$	92.40$_{\pm 0.25}$	86.53$_{\pm 0.46}$	89.47$_{\pm 0.26}$
TUNA (Wang et al., 2025)	83.13$_{\pm 0.36}$	75.36$_{\pm 0.78}$	79.25$_{\pm 0.43}$	92.81$_{\pm 0.28}$	88.16$_{\pm 0.57}$	90.49$_{\pm 0.32}$
PolyMem (Ours)	**83.72**$_{\pm 0.24}$	**77.85**$_{\pm 0.49}$	**80.79**$_{\pm 0.27}$	**93.21**$_{\pm 0.19}$	**89.45**$_{\pm 0.44}$	**91.33**$_{\pm 0.24}$

within each benchmark and then average across benchmarks:

$$\overline{\mathrm{LA}} = \frac{1}{|\mathcal{B}|} \sum_{b \in \mathcal{B}} \frac{1}{K_b} \sum_{d=1}^{K_b} \mathrm{LA}^b_{D_d}, \tag{9}$$

where $\mathcal{B}$ is the set of four mixed benchmarks and $K_b = 2$ for each benchmark. PolyMem achieves the highest average LA among the evaluated CIL methods, consistent with the results in Tab. 1.

B Additional Task-Split Experiments

We evaluate PolyMem under two additional class-incremental task partitions: ModelNet + CO3D with B20Inc4, comprising 20 initial classes followed by 4 new classes per task for 6 tasks in total;

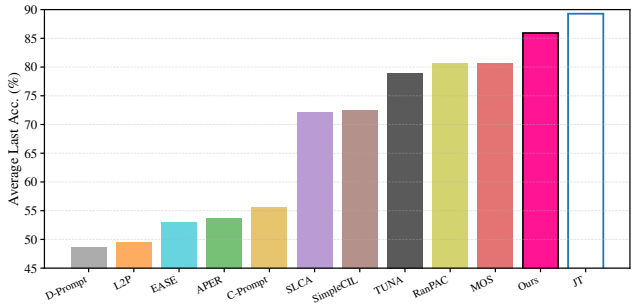


Figure 8: Overall comparison of PTM-based CIL methods on Domain3D-CIL. Last Accuracy is averaged equally across domains within each benchmark and then across the four benchmarks.

and ShapeNet + ShapeNet-C with B7Inc2, comprising 7 initial classes followed by 2 new classes per task for 25 tasks in total. Compared with the main experiments, these settings use a larger initial task with fewer incremental stages and a smaller class increment with more stages, respectively.

Tabs. 8 and 9 report LA and CA on each domain and their equally weighted average. PolyMem achieves the highest average LA and CA in both settings. On ModelNet + CO3D, its average LA and CA reach 90.28% and 91.42%, respectively; on ShapeNet + ShapeNet-C, they reach 82.68% and 88.86%. PolyMem also achieves the highest LA on each domain in both settings, and the highest CA in all cases except the CAD domain of ModelNet + CO3D, where it ranks second. These results support its effectiveness across different class-incremental task partitions. The best and second-best results among CIL methods are highlighted in **bold** and underlined, respectively.

Table 8: Results on ModelNet + CO3D under B20Inc4 (%). Avg. denotes the equally weighted average across domains.

Method	Last Acc.			Cum. Acc.		
	$D_1\uparrow$	$D_2\uparrow$	Avg.$\uparrow$	$D_1\uparrow$	$D_2\uparrow$	Avg.$\uparrow$
Joint-Training	$\overline{92.88}$	$\overline{89.21}$	$\overline{91.05}$	–	–	–
L2P (Wang et al., 2022d)	$64.29_{\pm 0.72}$	$38.15_{\pm 1.16}$	$51.22_{\pm 0.68}$	$74.44_{\pm 0.48}$	$54.53_{\pm 0.84}$	$64.49_{\pm 0.48}$
DualPrompt (Wang et al., 2022c)	$70.06_{\pm 0.61}$	$43.06_{\pm 1.02}$	$56.56_{\pm 0.59}$	$79.49_{\pm 0.44}$	$58.61_{\pm 0.76}$	$69.05_{\pm 0.44}$
CODA-Prompt (Smith et al., 2023)	$62.15_{\pm 0.58}$	$54.82_{\pm 0.95}$	$58.49_{\pm 0.56}$	$70.77_{\pm 0.39}$	$65.23_{\pm 0.68}$	$68.00_{\pm 0.39}$
SLCA (Zhang et al., 2023)	$82.54_{\pm 0.41}$	$69.89_{\pm 0.86}$	$76.22_{\pm 0.48}$	$87.47_{\pm 0.25}$	$74.25_{\pm 0.58}$	$80.86_{\pm 0.32}$
SimpleCIL (Zhou et al., 2025a)	$82.98_{\pm 0.33}$	$59.03_{\pm 0.73}$	$71.01_{\pm 0.40}$	$86.24_{\pm 0.29}$	$61.13_{\pm 0.52}$	$73.69_{\pm 0.30}$
APER (Zhou et al., 2025a)	$75.08_{\pm 0.63}$	$70.29_{\pm 0.82}$	$72.69_{\pm 0.52}$	$83.93_{\pm 0.47}$	$83.80_{\pm 0.69}$	$83.87_{\pm 0.42}$
EASE (Zhou et al., 2024b)	$57.13_{\pm 0.79}$	$28.98_{\pm 1.08}$	$43.06_{\pm 0.67}$	$74.24_{\pm 0.52}$	$59.88_{\pm 0.74}$	$67.06_{\pm 0.45}$
RanPAC (McDonnell et al., 2023)	$\underline{91.57}_{\pm 0.22}$	$85.27_{\pm 0.49}$	$88.42_{\pm 0.27}$	$\mathbf{94.14}_{\pm 0.16}$	$83.52_{\pm 0.36}$	$88.83_{\pm 0.20}$
MOS (Sun et al., 2025)	$91.30_{\pm 0.28}$	$\underline{86.37}_{\pm 0.44}$	$\underline{88.84}_{\pm 0.26}$	$92.69_{\pm 0.20}$	$\underline{88.08}_{\pm 0.31}$	$\underline{90.39}_{\pm 0.18}$
TUNA (Wang et al., 2025)	$89.02_{\pm 0.37}$	$78.46_{\pm 0.62}$	$83.74_{\pm 0.36}$	$92.36_{\pm 0.24}$	$83.27_{\pm 0.48}$	$87.82_{\pm 0.27}$
PolyMem (Ours)	$\mathbf{92.27}_{\pm 0.21}$	$\mathbf{88.28}_{\pm 0.46}$	$\mathbf{90.28}_{\pm 0.25}$	$\underline{93.69}_{\pm 0.15}$	$\mathbf{89.14}_{\pm 0.38}$	$\mathbf{91.42}_{\pm 0.20}$

Fig. 9 shows the task-wise cross-domain average accuracy under these two task splits. Following the notation in the main paper, let $a_{t,d}$ denote accuracy on domain D_d after task t. Each curve reports the equally weighted average $\mathrm{AvgAcc}_t = \frac{1}{K}\sum_{d=1}^{K} a_{t,d}$, where $K = 2$. Predictions are made over all classes observed so far.

Fig. 9a presents ModelNet + CO3D under B20Inc4, while Fig. 9b presents ShapeNet + ShapeNet-C under B7Inc2. The curves complement the aggregate LA and CA results by showing how accuracy evolves over 6 and 25 tasks, respectively.

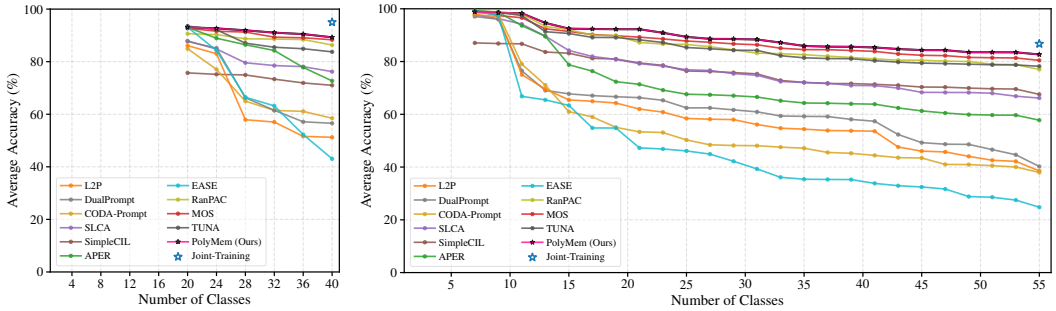
C Parameter, Storage, and Runtime Analysis

Parameter and Storage Costs. We compare the method-specific parameter and storage costs of PTM-based CIL methods, excluding the shared pre-trained backbone.

Tab. 10 reports two categories. **Learnable Params** counts additional model parameters, including prompts, adapters, and fitted classifier parameters. **Auxiliary Storage** counts additional stored entries

Table 9: Results on ShapeNet + ShapeNet-C under B7Inc2 (%). Avg. denotes the equally weighted average across domains.

Method	Last Acc.			Cum. Acc.		
	D_1↑	D_2↑	Avg.↑	D_1↑	D_2↑	Avg.↑
Joint-Training	87.42	85.01	86.22	–	–	–
L2P (Wang et al., 2022d)	$42.35_{\pm 1.42}$	$34.72_{\pm 1.85}$	$38.54_{\pm 1.17}$	$62.33_{\pm 0.94}$	$54.84_{\pm 1.31}$	$58.59_{\pm 0.81}$
DualPrompt (Wang et al., 2022c)	$45.33_{\pm 1.26}$	$35.11_{\pm 1.72}$	$40.22_{\pm 1.07}$	$65.83_{\pm 0.82}$	$57.59_{\pm 1.18}$	$61.71_{\pm 0.72}$
CODA-Prompt (Smith et al., 2023)	$42.34_{\pm 1.18}$	$33.56_{\pm 1.58}$	$37.95_{\pm 0.99}$	$58.21_{\pm 0.91}$	$49.03_{\pm 1.09}$	$53.62_{\pm 0.71}$
SLCA (Zhang et al., 2023)	$75.29_{\pm 0.74}$	$57.00_{\pm 1.28}$	$66.15_{\pm 0.74}$	$83.40_{\pm 0.55}$	$70.18_{\pm 0.96}$	$76.79_{\pm 0.55}$
SimpleCIL (Zhou et al., 2025a)	$76.27_{\pm 0.62}$	$58.82_{\pm 1.17}$	$67.55_{\pm 0.66}$	$83.91_{\pm 0.46}$	$68.03_{\pm 0.81}$	$75.97_{\pm 0.47}$
APER (Zhou et al., 2025a)	$64.24_{\pm 1.07}$	$51.28_{\pm 1.43}$	$57.76_{\pm 0.89}$	$75.26_{\pm 0.71}$	$65.57_{\pm 1.05}$	$70.42_{\pm 0.63}$
EASE (Zhou et al., 2024b)	$28.71_{\pm 1.58}$	$20.83_{\pm 2.06}$	$24.77_{\pm 1.30}$	$48.20_{\pm 1.12}$	$43.89_{\pm 1.33}$	$46.05_{\pm 0.87}$
RanPAC (McDonnell et al., 2023)	$81.78_{\pm 0.53}$	$72.24_{\pm 0.97}$	$77.01_{\pm 0.55}$	$88.63_{\pm 0.34}$	$82.64_{\pm 0.68}$	$85.64_{\pm 0.38}$
MOS (Sun et al., 2025)	$84.99_{\pm 0.49}$	$75.90_{\pm 0.88}$	$80.45_{\pm 0.50}$	$90.41_{\pm 0.31}$	$83.79_{\pm 0.59}$	$87.10_{\pm 0.33}$
TUNA (Wang et al., 2025)	$81.76_{\pm 0.66}$	$74.64_{\pm 1.04}$	$78.20_{\pm 0.62}$	$87.72_{\pm 0.42}$	$82.59_{\pm 0.74}$	$85.16_{\pm 0.43}$
PolyMem (Ours)	$\mathbf{86.51}_{\pm 0.38}$	$\mathbf{78.85}_{\pm 0.76}$	$\mathbf{82.68}_{\pm 0.42}$	$\mathbf{91.40}_{\pm 0.27}$	$\mathbf{86.32}_{\pm 0.58}$	$\mathbf{88.86}_{\pm 0.32}$



(a) ModelNet + CO3D, B20Inc4

(b) ShapeNet + ShapeNet-C, B7Inc2

Figure 9: Cross-domain average accuracy under additional task splits. Each curve reports the equally weighted average of the two domain-specific accuracies after each incremental task. Left: ModelNet + CO3D with B20Inc4 and 6 tasks. Right: ShapeNet + ShapeNet-C with B7Inc2 and 25 tasks.

used during computation, including feature statistics, fixed mappings, and method-specific reference copies. For methods retaining an additional copy of the Uni3D-Base feature extractor, this copy contributes 88.961 million entries under our implementation. All counts are reported in millions of scalar entries; actual memory usage depends on their numerical precision.

For PolyMem, the main auxiliary storage consists of the accumulated matrices $G \in \mathbb{R}^{M\times M}$ and $Q \in \mathbb{R}^{M\times C_{\text{tot}}}$. With $M = 10000$ and $C_{\text{tot}} = 55$, these matrices contain

$$M^2 + MC_{\text{tot}} = 100{,}550{,}000 \tag{10}$$

entries in total. The fixed TensorSketch descriptors additionally store hash indices and random signs. For $L = 4$ and $D = 1024$, their total size is

$$2\sum_{n=1}^{L} nD = L(L+1)D = 20{,}480. \tag{11}$$

PolyMem therefore requires approximately 100.571 million auxiliary entries, compared with 111.04 million for RanPAC.

The storage complexity of PolyMem's statistical memory is $\mathcal{O}(M^2 + MC_{\text{tot}})$. The dominant matrix G remains fixed in size as tasks and classes accumulate, while Q grows with the number of observed classes. Neither matrix grows directly with the number of historical samples. TensorSketch descriptors contribute only approximately 0.021 million entries.

These auxiliary statistics do not require gradient storage and can reside in CPU memory for incremental updates and classifier computation. During inference, neither G nor Q is needed: the model uses only the backbone, the fixed polynomial mapping, and the fitted classifier W. For $M = 10000$

Table 10: Additional model parameters and auxiliary storage of PTM-based CIL methods, excluding the shared pre-trained backbone. Values are reported in millions of scalar entries. Fitted classifier parameters are included in Learnable Params.

Method	Learnable Params (M)	Auxiliary Storage (M)
L2P (Wang et al., 2022d)	1.88	88.96
DualPrompt (Wang et al., 2022c)	0.97	88.96
CODA-Prompt (Smith et al., 2023)	3.10	–
SLCA (Zhang et al., 2023)	0.05	32.44
SimpleCIL (Zhou et al., 2025a)	0.05	–
MOS (Sun et al., 2025)	7.34	32.44
TUNA (Wang et al., 2025)	7.65	32.44
RanPAC (McDonnell et al., 2023)	0.53	111.04
APER (Zhou et al., 2025a)	89.07	88.96
EASE (Zhou et al., 2024b)	0.98	–
PolyMem (Ours)	0.53	100.57

and 55 classes, W and the TensorSketch descriptors together contain approximately 0.571 million entries, excluding the backbone. Thus, the statistical memory maintained during incremental learning does not need to remain resident during inference.

Runtime. Fig. 10 compares the total training and testing time of the evaluated methods under the computational environment described in Sec. E. PolyMem requires substantially less time than most evaluated baselines and has a runtime comparable to RanPAC. Its fixed polynomial mapping and accumulated-statistics classifier support incremental learning without backpropagation through the backbone or feature mapping.

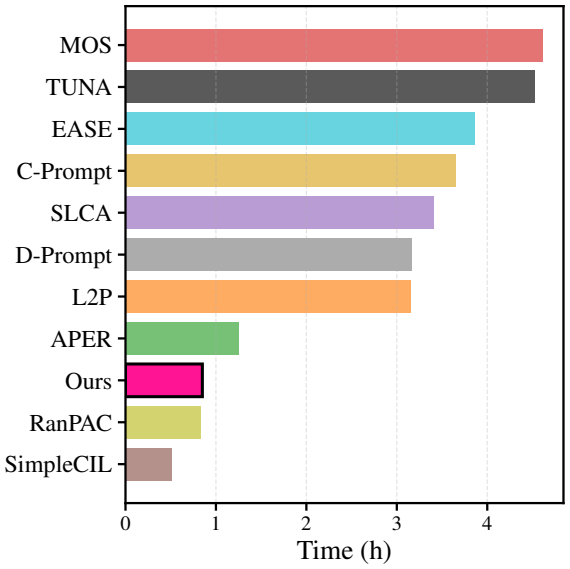


Figure 10: Total training and testing time of PTM-based CIL methods and PolyMem under the same experimental setting. Computational resources are described in Sec. E.

D Classic CIL on Domain3D-CIL

We additionally evaluate classic CIL baselines on Domain3D-CIL. These methods are adapted to point clouds using PointNet++ (Qi et al., 2017b) trained from scratch, with an exemplar memory budget of 10 samples per class. This setting complements the PTM-based experiments by evaluating incremental learning with exemplar replay and without a pre-trained backbone. We report these results separately because the backbone, initialization, and replay protocol differ from those of the main experiments.

Tab. 11 reports LA and CA on ModelNet + ScanObjectNN and ModelNet + CO3D. For each metric, we report accuracy on the CAD domain D_1, the real-scanned or reconstructed domain D_2, and their equally weighted average. Joint-training results use the same PointNet++ architecture.

Among the evaluated CIL methods, DER achieves the highest cross-domain average LA and CA on both benchmarks. Its average LA reaches 59.46% on ModelNet + ScanObjectNN and 61.42% on ModelNet + CO3D, while its average CA reaches 70.30% and 70.11%, respectively. DER also achieves the highest LA and CA on the real-world domain in both benchmarks. On ModelNet + CO3D, WA achieves the highest CAD-domain LA, while PODNet achieves the highest CAD-domain CA, showing that method rankings vary across domains and metrics. Overall, these experiments extend the evaluation of heterogeneous 3D CIL to the classic replay-based setting.

Table 11: Quantitative evaluation of classic CIL baselines on Domain3D-CIL (%). All methods use PointNet++ without pre-training. CIL methods use a replay budget of 10 exemplars per class. Avg. denotes the equally weighted average across domains. The best and second-best results among CIL methods are highlighted in **bold** and underlined, respectively.

Method	ModelNet (D_1) + ScanObjectNN (D_2)						ModelNet (D_1) + CO3D (D_2)					
	Last Acc.			Cum. Acc.			Last Acc.			Cum. Acc.		
	$D_1\uparrow$	$D_2\uparrow$	Avg.↑	$D_1\uparrow$	$D_2\uparrow$	Avg.↑	$D_1\uparrow$	$D_2\uparrow$	Avg.↑	$D_1\uparrow$	$D_2\uparrow$	Avg.↑
Joint-Training	90.72	86.51	88.62	–	–	–	91.21	86.62	88.92	–	–	–
ER	69.00	40.95	54.98	74.17	46.38	60.28	66.17	35.28	50.73	67.53	38.53	53.03
iCaRL (Rebuffi et al., 2017)	72.20	39.88	56.04	79.29	53.63	66.46	65.15	39.41	52.28	72.66	48.99	60.83
BiC (Wu et al., 2019)	60.94	38.57	49.76	63.60	40.67	52.14	61.39	38.46	49.93	61.39	42.92	52.16
WA (Zhao et al., 2020)	71.03	38.67	54.85	79.18	51.73	65.46	**68.68**	46.17	57.43	75.77	52.87	64.32
PODNet (Douillard et al., 2020)	68.68	38.17	53.43	80.13	52.04	66.09	62.93	36.33	49.63	**79.10**	51.08	65.09
DER (Yan et al., 2021)	**74.59**	**44.33**	**59.46**	**83.99**	**56.60**	**70.30**	67.10	**55.74**	**61.42**	78.41	**61.81**	**70.11**
FOSTER (Wang et al., 2022a)	65.19	43.94	54.57	72.18	50.93	61.56	64.14	44.08	54.11	75.37	50.23	62.80
iNeMo (Fischer et al., 2024)	69.98	35.39	52.69	76.88	46.08	61.48	57.74	42.49	50.12	63.82	46.30	55.06

E Compute Resources

The running-time comparison in Fig. 10 and the parameter analysis in Tab. 10 are measured under the same computational environment described below. All experiments were conducted on compute nodes equipped with 8× NVIDIA A40 GPUs. Each GPU provides 46,068 MiB of memory, i.e., about 48 GB. The NVIDIA driver version is 570.195.03, and the CUDA compatibility version reported by the driver is 12.8. This reported CUDA version indicates the highest CUDA toolkit level supported by the driver, and does not necessarily correspond to the CUDA version used to compile a specific deep learning framework.

For each random seed, a single training run uses only one GPU. We do not use multi-GPU data parallelism within a single run. Multiple GPUs on the same node are used only to execute different experiments in parallel, such as different methods, random seeds, or hyperparameter configurations.

F Theoretical Background

This subsection provides background knowledge on distributional statistics, polynomial feature interactions, kernel approximation, and closed-form ridge regression. Readers familiar with these concepts may skip this section.

F.1 Distributional Statistics

A statistic is a quantity computed from data to summarize a property of an underlying distribution. Moments are a fundamental class of statistics that describe the shape of a distribution through expectations of powers of a random variable. For a scalar random variable X, the r-th raw moment and the r-th central moment are defined as

$$m_r = \mathbb{E}[X^r], \qquad \mu_r = \mathbb{E}\big[(X-\mu)^r\big], \tag{12}$$

where $\mu = \mathbb{E}[X]$ is the mean. The mean and variance are the most common low-order summaries:

$$\mu = \mathbb{E}[X], \qquad \sigma^2 = \mathbb{E}\big[(X-\mu)^2\big]. \tag{13}$$

The mean describes the central location of a distribution, while the variance describes its dispersion.

Beyond the first two orders, higher-order moments characterize more detailed properties of the distributional shape. The third central moment measures the asymmetry of a distribution around its mean. Its standardized form is known as skewness (Pearson, 1895):

$$\gamma_1 = \frac{\mathbb{E}\big[(X-\mu)^3\big]}{\sigma^3}. \tag{14}$$

A positive skewness indicates that the distribution has a longer or heavier right tail, while a negative skewness indicates a longer or heavier left tail. When the distribution is symmetric around its mean, the third central moment is zero. Thus, skewness captures information that cannot be reflected by the mean or variance alone.

The fourth central moment characterizes the concentration and tail behavior of a distribution. Its standardized form is called kurtosis (Pearson, 1905):

$$\gamma_2 = \frac{\mathbb{E}\big[(X-\mu)^4\big]}{\sigma^4}. \tag{15}$$

A distribution with larger kurtosis typically places more probability mass in the tails and around the center, compared with a distribution with lighter tails. Kurtosis is therefore commonly used to describe heavy-tailed behavior and the frequency of extreme deviations from the mean.

F.2 Polynomial Feature Interactions

The previous subsection introduces moments as statistics obtained from expectations of powers of random variables. Polynomial feature interactions provide the corresponding feature-level building blocks for such statistics. The key distinction is that a polynomial interaction from a single sample is not yet a statistic; it becomes a statistic only after taking expectation over a distribution or empirical aggregation over samples.

To see this in the simplest case, consider a scalar random variable X. The polynomial terms X^2, X^3, and X^4 become raw moments after taking expectation:

$$\mathbb{E}[X^2], \qquad \mathbb{E}[X^3], \qquad \mathbb{E}[X^4]. \tag{16}$$

Central moments can be computed from these raw moments. For example, the second central moment is

$$\mathbb{E}[(X-\mu)^2] = \mathbb{E}[X^2] - \mu^2, \tag{17}$$

and the third central moment is

$$\mathbb{E}[(X-\mu)^3] = \mathbb{E}[X^3] - 3\mu\mathbb{E}[X^2] + 2\mu^3. \tag{18}$$

Thus, higher-order statistics such as variance and skewness are formed from aggregated polynomial terms, rather than from isolated single-sample products.

For a vector $F \in \mathbb{R}^D$, the same idea extends to cross-dimensional statistics. An n-th order polynomial interaction is a degree-n product of feature dimensions:

$$F_{d_1} F_{d_2} \cdots F_{d_n}. \tag{19}$$

For a single sample, this is only a feature-level product. After aggregation, it forms a high-order raw moment or cross-moment:

$$\mathbb{E}[F_{d_1} F_{d_2} \cdots F_{d_n}], \qquad \frac{1}{N}\sum_{i=1}^{N} F_{i,d_1} F_{i,d_2} \cdots F_{i,d_n}. \tag{20}$$

For example, second-order products $F_p F_q$ give

$$\frac{1}{N}\sum_{i=1}^{N} F_{i,p} F_{i,q}, \tag{21}$$

which is a second-order raw cross-moment. When $p = q$, it reduces to a raw second moment of one feature dimension; after centering, the diagonal case corresponds to variance. Similarly, third-order products $F_p F_q F_r$ give

$$\frac{1}{N}\sum_{i=1}^{N} F_{i,p}F_{i,q}F_{i,r}, \tag{22}$$

which is a third-order raw moment or cross-moment. After centering, the one-dimensional case $p = q = r$ corresponds to the numerator of skewness. Therefore, polynomial feature interactions are elementary components from which high-order distributional statistics can be constructed through sample-level aggregation.

Polynomial interactions also arise naturally from polynomial kernels. Given two vectors $F_i, F_j \in \mathbb{R}^D$, the degree-$n$ polynomial inner product

$$(F_i^\top F_j)^n \tag{23}$$

can be viewed as an inner product in a degree-n ordered monomial feature space, or equivalently a tensor-product feature space. For instance, the quadratic case expands as

$$(F_i^\top F_j)^2 = \left(\sum_{d=1}^{D} F_{i,d}F_{j,d}\right)^2 = \sum_{p=1}^{D}\sum_{q=1}^{D} F_{i,p}F_{i,q}F_{j,p}F_{j,q}. \tag{24}$$

This expansion shows that the quadratic kernel compares all ordered second-degree feature products between F_i and F_j. Higher-degree polynomial kernels analogously compare higher-degree ordered feature products.

However, explicitly storing all polynomial interaction terms is impractical. A full ordered n-th order interaction tensor over a D-dimensional vector contains D^n entries. For $D = 1024$, this gives

$$D^2 = 1024^2 = 1{,}048{,}576, \tag{25}$$

$$D^3 = 1024^3 = 1{,}073{,}741{,}824, \tag{26}$$

and

$$D^4 = 1024^4 = 1{,}099{,}511{,}627{,}776 \tag{27}$$

entries for second-, third-, and fourth-order interactions, respectively. Using 32-bit floating point storage, these correspond to approximately 4 MiB, 4 GiB, and 4 TiB for a single full tensor. The cost would further increase if such tensors were maintained separately for multiple classes.

This distinction is important for PolyMem. PolyMem does not explicitly recover or store each high-order moment tensor. Instead, it constructs a compact polynomially lifted feature space and accumulates ridge-regression statistics in that space. In this sense, polynomial feature interactions make high-order feature products available for statistical accumulation, while the compact approximation introduced in the next subsection keeps the representation memory-efficient.

F.3 RBF KERNEL

A kernel function provides a way to measure the similarity between two inputs. From the perspective of kernel methods, a positive semi-definite kernel $k(\cdot,\cdot)$ corresponds to an implicit feature mapping $\phi(\cdot)$ such that

$$k(F_i, F_j) = \phi(F_i)^\top \phi(F_j). \tag{28}$$

Therefore, a nonlinear similarity function in the original space can be interpreted as a linear inner product in a mapped feature space. This viewpoint is useful when the mapped feature space is high-dimensional or even infinite-dimensional, because the kernel value can be evaluated without explicitly constructing all mapped features.

The radial basis function (RBF) kernel is one of the most widely used kernels. For two vectors $F_i, F_j \in \mathbb{R}^D$, it is defined as

$$k_{\mathrm{RBF}}(F_i, F_j) = \exp\!\left(-\gamma \|F_i - F_j\|_2^2\right), \tag{29}$$

where $\gamma > 0$ controls the bandwidth of the kernel. A larger γ makes the similarity decay faster as the distance between F_i and F_j increases, while a smaller γ gives a smoother similarity measure.

In PolyMem, the backbone features are L2-normalized before constructing the polynomial mapping. For normalized features satisfying $\|F_i\|_2 = \|F_j\|_2 = 1$, the squared Euclidean distance can be written as

$$\|F_i - F_j\|_2^2 = \|F_i\|_2^2 + \|F_j\|_2^2 - 2F_i^\top F_j = 2 - 2F_i^\top F_j. \tag{30}$$

Substituting this into the RBF kernel gives

$$\begin{aligned} k_{\mathrm{RBF}}(F_i, F_j) &= \exp\big(-\gamma(2 - 2F_i^\top F_j)\big) \\ &= \exp(-2\gamma)\exp(2\gamma F_i^\top F_j). \end{aligned} \tag{31}$$

The second exponential term can be expanded using the Taylor series:

$$\exp(2\gamma F_i^\top F_j) = \sum_{n=0}^{\infty} \frac{(2\gamma)^n}{n!}(F_i^\top F_j)^n. \tag{32}$$

Thus, the RBF kernel can be decomposed as

$$k_{\mathrm{RBF}}(F_i, F_j) = \exp(-2\gamma)\sum_{n=0}^{\infty} \frac{(2\gamma)^n}{n!}(F_i^\top F_j)^n. \tag{33}$$

This decomposition connects the RBF kernel to the polynomial feature interactions introduced in Appendix F.2. The zeroth-order term is a constant term, the first-order term corresponds to linear interactions, the second-order term corresponds to quadratic interactions, and higher-order terms correspond to higher-degree polynomial interactions. Therefore, the RBF kernel provides an order-wise decomposition that contains polynomial interactions of all degrees.

PolyMem uses a finite-order approximation by retaining the terms up to order L:

$$k_{\mathrm{Poly}}^{(L)}(F_i, F_j) = \exp(-2\gamma)\sum_{n=0}^{L} \frac{(2\gamma)^n}{n!}(F_i^\top F_j)^n. \tag{34}$$

The order L controls the maximum degree of polynomial interaction included in the mapped feature space. In this way, the RBF Taylor expansion provides a structured and order-wise basis for the PolyMem mapping, while the next subsection explains how these polynomial terms can be compactly approximated by TensorSketch.

F.4 Approximation by TensorSketch

The RBF Taylor expansion in Appendix F.3 decomposes the RBF kernel into polynomial interactions of different orders. However, directly constructing the explicit monomial feature map for each order is impractical. For the n-th order term, the explicit feature map contains degree-n products of the input dimensions, whose size grows on the order of $\mathcal{O}(D^n)$. As discussed in Appendix F.2, this cost becomes prohibitive even for moderate feature dimensions.

TensorSketch (Pham & Pagh, 2013) provides a compact randomized approximation to such polynomial feature maps. Let $\psi_n(F)$ denote the explicit degree-n monomial feature map. It satisfies

$$\psi_n(F_i)^\top \psi_n(F_j) = (F_i^\top F_j)^n. \tag{35}$$

Instead of materializing $\psi_n(F)$, TensorSketch constructs a compact feature

$$\mathrm{TS}_n(F) \in \mathbb{R}^{M_n}, \qquad M_n \ll D^n, \tag{36}$$

such that

$$\mathrm{TS}_n(F_i)^\top \mathrm{TS}_n(F_j) \approx (F_i^\top F_j)^n. \tag{37}$$

Therefore, TensorSketch keeps the polynomial-kernel inner product approximately available while avoiding the explicit $\mathcal{O}(D^n)$ expansion.

PolyMem applies TensorSketch separately to each polynomial order in the finite-order RBF Taylor expansion. For the n-th order term, PolyMem uses a sketch feature $\mathrm{TS}_n(F) \in \mathbb{R}^{M_n}$ and scales it according to the corresponding Taylor coefficient. Specifically, the order-n block is written as

$$e^{-\gamma}\sqrt{\frac{(2\gamma)^n}{n!}}\mathrm{TS}_n(F). \tag{38}$$

The square-root scaling is used because the kernel approximation is obtained by taking an inner product between two mapped features. When two order-n blocks are multiplied, the product of the two scaling factors recovers the coefficient $e^{-2\gamma}(2\gamma)^n/n!$ in the RBF Taylor expansion.

By concatenating the constant term and all order-wise sketch features, PolyMem defines

$$\phi_{\mathrm{Poly}}(F) = \left[e^{-\gamma},\ e^{-\gamma}\sqrt{\frac{2\gamma}{1!}}\mathrm{TS}_1(F),\ \ldots,\ e^{-\gamma}\sqrt{\frac{(2\gamma)^L}{L!}}\mathrm{TS}_L(F) \right] \in \mathbb{R}^M, \tag{39}$$

where the sketch dimensions satisfy

$$1 + \sum_{n=1}^{L} M_n = M. \tag{40}$$

With this construction, the inner product between two PolyMem features approximates the finite-order RBF Taylor kernel:

$$\begin{aligned} \phi_{\mathrm{Poly}}(F_i)^\top \phi_{\mathrm{Poly}}(F_j) &\approx e^{-2\gamma} \sum_{n=0}^{L} \frac{(2\gamma)^n}{n!} (F_i^\top F_j)^n \\ &= k^{(L)}_{\mathrm{Poly}}(F_i, F_j). \end{aligned} \tag{41}$$

Thus, TensorSketch enables PolyMem to retain structured order-wise polynomial interactions in a fixed-dimensional mapped feature space, without explicitly storing the full high-order interaction tensors.

F.5 Ridge Regression

The previous subsections explain how PolyMem constructs a compact mapped feature $F'_i = \phi_{\mathrm{Poly}}(F_i) \in \mathbb{R}^M$ that approximates the finite-order RBF Taylor feature space. After this mapping is fixed, the remaining question is how to learn a classifier from these mapped features without storing previous-task samples. PolyMem follows the closed-form ridge regression formulation (Hoerl & Kennard, 1970), which naturally supports additive statistic accumulation.

For each sample x_i, let $F'_i = \phi_{\mathrm{Poly}}(F_i)$ denote its mapped feature, and let $e_{y_i} \in \mathbb{R}^C$ denote the one-hot label vector. A linear classifier in the mapped feature space predicts

$$s(x_i) = W^\top F'_i, \tag{42}$$

where $W \in \mathbb{R}^{M\times C}$ is the classifier weight matrix. Ridge regression learns W by minimizing the regularized squared loss:

$$\min_W \sum_{i=1}^{N} \left\| W^\top F'_i - e_{y_i} \right\|_2^2 + \lambda \|W\|_F^2, \tag{43}$$

where λ is the ridge coefficient. The regularization term improves numerical stability and controls the magnitude of the classifier weights.

This objective has a closed-form solution. Taking the derivative with respect to W and setting it to zero gives the normal equation

$$(G + \lambda I)W = Q, \tag{44}$$

where

$$G = \sum_{i=1}^{N} F'_i F'^\top_i \in \mathbb{R}^{M\times M}, \qquad Q = \sum_{i=1}^{N} F'_i e_{y_i}^\top \in \mathbb{R}^{M\times C}. \tag{45}$$

Thus, the classifier can be obtained as

$$W = (G + \lambda I)^{-1} Q. \tag{46}$$

Here, G stores the uncentered second-order statistic of the mapped features, while Q stores the correlation between mapped features and class labels.

The key property for class-incremental learning is that both G and Q are additive over samples. Suppose task t provides the current training set $\mathcal{D}_t$. Instead of recomputing statistics from all previous data, we only need to add the current-task contributions:

$$G \leftarrow G + \sum_{(x_i,y_i)\in\mathcal{D}_t} F_i' F_i'^{\top}, \qquad Q \leftarrow Q + \sum_{(x_i,y_i)\in\mathcal{D}_t} F_i' e_{y_i}^{\top}. \tag{47}$$

After the update, solving $(G + \lambda I)W = Q$ gives the classifier over all classes represented in the accumulated statistics. Therefore, previous-task samples do not need to be stored.

This additive formulation is valid only when all tasks are represented in a consistent feature space. This is why PolyMem initializes $\phi_{\mathrm{Poly}}(\cdot)$ once at the first task and keeps it fixed afterwards. With a fixed mapping, the statistics accumulated from different tasks are compatible, and the memory cost is determined by the stored matrices $G \in \mathbb{R}^{M\times M}$ and $Q \in \mathbb{R}^{M\times C}$, rather than by the number of training samples. This directly leads to the task-wise PolyMem procedure summarized in Algorithm 1.

G Algorithmic Description of PolyMem

We summarize the task-wise training procedure of PolyMem in Algorithm 1. The construction of the PolyMem mapping $\phi_{\mathrm{Poly}}(\cdot)$ follows the finite-order RBF Taylor expansion and the TensorSketch-based polynomial approximation introduced in Appendix F.3 and Appendix F.4. Here, we focus on how this fixed mapping is used for statistic accumulation and closed-form classifier estimation in the PTM-based CIL setting.

Given an input sample x_i, the pre-trained backbone extracts a feature $F_i = f(x_i) \in \mathbb{R}^D$, which is mapped to $F_i' = \phi_{\mathrm{Poly}}(F_i) \in \mathbb{R}^M$. The mapping $\phi_{\mathrm{Poly}}(\cdot)$ is initialized once at the first task and then kept fixed throughout the incremental learning process. This fixed-mapping design ensures that the accumulated statistics from all tasks are represented in the same polynomially lifted feature space. At each task, PolyMem only uses the current-task training samples to update the statistics Q and G defined in Eq. 6. The classifier is then obtained by the closed-form ridge regression solution described in Appendix F.5. For a fair comparison with RanPAC (McDonnell et al., 2023), the ridge coefficient is selected from the same fixed candidate set $\Lambda = \{10^{-8}, 10^{-7}, \ldots, 10^{7}, 10^{8}\}$. Therefore, PolyMem does not store raw samples or raw backbone features from previous tasks.

Algorithm 1 PolyMem for PTM-based Class-Incremental Learning

Require: Incremental training sets $\{\mathcal{D}_t\}_{t=0}^{T-1}$; backbone $f(\cdot)$; total number of classes C; PolyMem dimension M; Taylor order L; ridge candidate set Λ

Ensure: Classifier weights for all seen classes after each task

1: Initialize $Q \leftarrow \mathbf{0} \in \mathbb{R}^{M\times C}$, $G \leftarrow \mathbf{0} \in \mathbb{R}^{M\times M}$, and PolyMem mapping parameters as empty
2: **for** $t = 0, 1, \ldots, T-1$ **do**
3: Let $\mathcal{Y}_t$ be the classes introduced at task t, and extract $F_i = f(x_i)$ for all $(x_i, y_i) \in \mathcal{D}_t$
4: **if** $t = 0$ **then**
5: Initialize the fixed mapping $\phi_{\mathrm{Poly}}(\cdot)$ with Taylor order L and dimension M; set $\gamma = \left(2\,\mathrm{median}_{(i,j)\in\mathcal{P}_0}\|\bar{F}_i - \bar{F}_j\|_2^2\right)^{-1}$ on randomly sampled task-0 normalized feature pairs unless γ is manually specified
6: **end if**
7: Map current-task features as $F_i' = \phi_{\mathrm{Poly}}(F_i) \in \mathbb{R}^M$ and construct one-hot labels $e_{y_i} \in \mathbb{R}^C$
8: Update accumulated statistics:

$$Q \leftarrow Q + \sum_{(x_i,y_i)\in\mathcal{D}_t} F_i' e_{y_i}^{\top}, \qquad G \leftarrow G + \sum_{(x_i,y_i)\in\mathcal{D}_t} F_i' F_i'^{\top}.$$

9: Select $\lambda_t \in \Lambda$ using a validation split of the current task and solve

$$W_t = (G + \lambda_t I)^{-1} Q.$$

10: Use the columns of W_t corresponding to seen classes $\bigcup_{j=0}^{t} \mathcal{Y}_j$ for classification
11: **end for**

H Dataset Details

This appendix summarizes the source datasets, mixed benchmarks, preprocessing, and semantic alignment used in Domain3D-CIL. Unless otherwise stated, point clouds are XYZ-only, resampled to $N{=}1024$ points per shape, centered by subtracting the point-set centroid, and isotropically scaled by the maximum absolute coordinate so that all points fit inside $[-1, 1]^3$. We refer to this preprocessing as *unit-cube normalization*.

Domain tag `type` vs. training. Each local HDF5 record stores a string field `type` indicating the data source or corruption type, e.g., `clean`, `real`, `noisy`, or a specific corruption identifier. **The `type` field is used only for bookkeeping and domain-stratified evaluation; it is not exposed to the model during incremental training**, consistent with the main protocol where domain labels are unavailable during training.

H.1 Source Datasets

ModelNet40 (Wu et al., 2015). ModelNet40 contains synthetic CAD objects from common indoor and household categories. We use the standard **40**-class split with **train/test = 9,840/2,468** shapes, totaling **12,308** shapes.

ShapeNetCore(v2) (Chang et al., 2015). We adopt a **fixed 55-class** point-cloud taxonomy, used consistently across ShapeNet, overlapping OmniObject3D categories, and ShapeNet-C. This taxonomy is not intended to enumerate the full ShapeNet synset inventory. The commonly used split contains **train/validation/test = 35,708/5,158/10,261** shapes. Following our experimental protocol, we use only the training and test partitions, resulting in **train/test = 35,708/10,261** shapes, totaling **45,969** shapes. The validation partition is not used in our benchmark construction.

ScanObjectNN (Uy et al., 2019). ScanObjectNN contains real indoor object scans captured with RGB-D sensors. We use the **15** categories in the official `shape_names_ext` order distributed with the HKUST reference materials. The full split contains **train/test = 2,309/581** scans, totaling **2,890** scans.

CO3D (Reizenstein et al., 2021). CO3D provides real object reconstructions from crowdsourced videos. We use the official **sequence-level reconstructed point clouds** in PLY format bundled with CO3D v2, typically corresponding to one fused reconstruction per video sequence rather than per-frame point clouds. For the ModelNet40 + CO3D benchmark, we retain **13** categories that can be aligned with ModelNet40. After an approximately **80/20** stratified split per class with a fixed random seed, the retained subset contains **train/test = 8,823/2,205** point clouds, totaling **11,028** point clouds.

OmniObject3D (Wu et al., 2023). OmniObject3D contains high-quality real-scanned 3D objects over **216** categories. Before semantic filtering and merging with ShapeNet, the split used here contains **train/test = 4,641/1,270** scans, totaling **5,911** scans.

ShapeNet-C. ShapeNet-C denotes corrupted ShapeNet counterparts constructed following PointCloud-C (Ren et al., 2022). We use the complete ShapeNet-C bank constructed from ShapeNet, covering the following corruption families: `scale`, `jitter`, `rotate`, `dropout_global`, `dropout_local`, `add_global`, and `add_local`. In the ShapeNet + ShapeNet-C benchmark, corrupted samples are randomly selected from this ShapeNet-C bank following the mixing protocol described in §H.3.

H.2 Semantic Mappings

Tables 12 and 13 summarize the semantic bridges used to merge non-CAD datasets into the corresponding CAD taxonomies.

Table 12: Semantic alignments to ModelNet40 for ModelNet40 + ScanObjectNN and ModelNet40 + CO3D.

(a) ScanObjectNN → ModelNet40

ScanObjectNN	ModelNet40
bed	bed
box	glass_box
cabinet	wardrobe
chair	chair
desk	desk
display	monitor
door	door
shelf	bookshelf
sink	sink
sofa	sofa
table	table
toilet	toilet

(b) CO3D → ModelNet40

CO3D	ModelNet40
bench	bench
bottle	bottle
bowl	bowl
car	car
chair	chair
couch	sofa
cup	cup
keyboard	keyboard
laptop	laptop
plant	plant
toilet	toilet
plane	airplane
vase	vase

Table 13: Full OmniObject3D → ShapeNet55 name mapping used in ShapeNet + OmniObject3D.

OmniObject3D	ShapeNet55	OmniObject3D	ShapeNet55
bed	bed	bottle	bottle
bowl	bowl	cabinet	cabinet
chair	chair	clock	clock
cup	mug	guitar	guitar
hat	cap	helmet	helmet
keyboard	keyboard	knife	knife
laptop	laptop	light	lamp
microwaveoven	microwave	monitor	display
picnic_basket	basket	pillow	pillow
remote_control	remote_control	skateboard	skateboard
sofa	sofa	speaker	loudspeaker
table	table	bus	bus
car	car	motorcycle	motorcycle
plane	airplane	train	train

H.3 SHAPENET + SHAPENET-C GENERATION

Training split. Let $\mathcal{I}_{\text{train}}$ be the ShapeNet training object index set. We randomly partition $\mathcal{I}_{\text{train}}$into two disjoint equal-size subsets, $\mathcal{I}_{\text{clean}}$ and $\mathcal{I}_{\text{corr}}$, such that

$$\mathcal{I}_{\text{clean}} \cap \mathcal{I}_{\text{corr}} = \emptyset, \qquad |\mathcal{I}_{\text{clean}}| = |\mathcal{I}_{\text{corr}}| = 17{,}854.$$

For indices in $\mathcal{I}_{\text{clean}}$, we keep the original clean ShapeNet point clouds. For indices in $\mathcal{I}_{\text{corr}}$, we randomly sample corresponding ShapeNet-C point clouds of the same underlying objects from the complete ShapeNet-C bank. When multiple corrupted candidates are available for an object, one candidate is randomly selected. Therefore, each physical CAD object appears in exactly one training branch, either clean or corrupted, but never both.

Test split. For evaluation, we concatenate the full clean ShapeNet test split with a corresponding ShapeNet-C test subset. Specifically, each of the **10,261** ShapeNet test objects appears once in the clean branch and once in the corrupted branch, giving **20,522** test samples in total. When multiple corrupted candidates are available for a test object, one candidate is selected using the same fixed-seed selection protocol. This design allows domain-wise evaluation between clean CAD point clouds and corrupted point clouds while preserving object-index correspondence.

H.4 Data Availability and Licensing

The source datasets used in this paper are available from their official providers, subject to their respective licenses or terms of use. We do **not** redistribute any official dataset files, raw point clouds, reconstructed point clouds, meshes, or processed HDF5 shards derived from these datasets. We will only provides the information needed to reproduce the benchmark construction after users obtain the official datasets themselves. **Users are responsible for obtaining the source datasets from the official providers and complying with the corresponding licenses and terms of use.** Thus, reproducing Domain3D-CIL requires following the mixed-benchmark protocol described above using properly obtained local copies of the source datasets.

I Baseline Implementation Details

The PTM-based CIL baselines in the main paper are implemented by adapting the official LAMDA-PILOT codebase (Zhou et al., 2024a) to the 3D setting. Since Uni3D (Zhou et al., 2023b) follows the same architecture with ViT , its interface is largely compatible with the vision-transformer backbones used in LAMDA-PILOT. Therefore, most baselines require only minimal modifications, mainly replacing the original 2D image encoder with the frozen Uni3D point-cloud encoder and adapting the data loader, feature extraction, and evaluation pipeline to point-cloud inputs. We keep the core algorithmic components of each baseline unchanged whenever possible, including prompt selection, prototype/statistical memory construction, classifier update rules, and evaluation metrics. The implementation of the PTM-based CIL baselines used for review is included in the **supplementary materials**.