Informative Point cloud Dataset Extraction for Classification via Gradient-based Points Moving

Wenxiao Zhang
University of Science and Technology of China
Hefei, China
wenxxiao.zhang@gmail.com

Ziqi Wang
Wuhan University
Wuhan, China
ziqi.wangwang@gmail.com

Li Xu
Singapore University of Technology and Design
Singapore, Singapore
li_xu@mymail.sutd.edu.sg

Xun Yang
MoE Key Laboratory of Brain-inspired Intelligent Perception and Cognition, University of Science and Technology of China
Hefei, China
xyang21@ustc.edu.cn

Jun Liu*
Lancaster University
Lancaster, United Kingdom
j.liu81@lancaster.ac.uk

ABSTRACT

Point cloud plays a significant role in recent learning-based vision tasks, which contain additional information about the physical space compared to 2D images. However, such a 3D data format also results in more expensive training costs to train a sophisticated network with large 3D datasets. Previous methods for point cloud compression focus on compacting the representation of each point cloud for better storage and transmission but ignore the improvements in training efficiency. In this paper, we introduce a new open problem in the point cloud field, named *point cloud condensation*: Can we condense a large point cloud dataset into a much smaller synthetic dataset while preserving the important information of the original large dataset? In other words, we explore the possibility of training a network on a smaller dataset of informative point clouds extracted from the original large dataset but maintaining similar network classification performance. Training on this small synthetic dataset could largely improve the training efficiency. To achieve this goal, we propose a two-stage approach to generate the synthetic dataset. We first introduce a nearest-feature-mean based strategy to initialize the synthetic dataset, and then formulate our goal as a parameter-matching problem, which we solve by introducing a gradient-matching strategy to iteratively refine the synthetic dataset. We conduct extensive experiments on various synthetic and real-scanned 3D object classification benchmarks, showing that our synthetic dataset could achieve almost the same performance with only 5% point clouds of ScanObjectNN dataset compared to training with the full dataset. Codes are available at https://github.com/XLechter/PointCondensation.

CCS CONCEPTS

• **Computing methodologies** → **Point-based models**.

KEYWORDS

Point cloud Condensation, Point cloud, Shape Representation

ACM Reference Format:
Wenxiao Zhang, Ziqi Wang, Li Xu, Xun Yang, and Jun Liu. 2024. Informative Point cloud Dataset Extraction for Classification via Gradient-based Points Moving. In *Proceedings of the 32nd ACM International Conference on Multimedia (MM '24), October 28-November 1, 2024, Melbourne, VIC, AustraliaProceedings of the 32nd ACM International Conference on Multimedia (MM'24), October 28-November 1, 2024, Melbourne, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3664647.3680767

1 INTRODUCTION

Point clouds have recently gained great popularity for representing objects in 3D vision [13, 16, 32, 36, 73, 74, 76]. Several 3D object datasets have been created, such as ModelNet [61], ShapeNet [9], and ScanObjectNN [53]. Also, networks specifically designed for processing point cloud have been proposed [20, 28, 30, 38, 41–43, 52, 56, 73]. Training networks that work on point clouds is often more computationally expensive than the 2D image counterpart, as point clouds have irregular structures where the traditional convolution operation cannot be directly applied[5, 6, 34, 35, 57, 62, 71, 72, 75]. Some attempts have been made to design a convolution operation that operates directly on points [28, 42, 52, 56]. However, they commonly need to search the nearest neighbor points to combine the local property, involving a greatly high computational cost. Though downsampling the point cloud to a smaller number of points is a solution, it will significantly reduce the testing accuracy.

When a new network architecture is developed, as the new network involves new combinations of different layers, it must be trained from scratch again on the training sets. This process can take a long time, especially if the training set is large or if the

*Corresponding author.

MM '24, October 28-November 1, 2024, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-0686-8/24/10
https://doi.org/10.1145/3664647.3680767

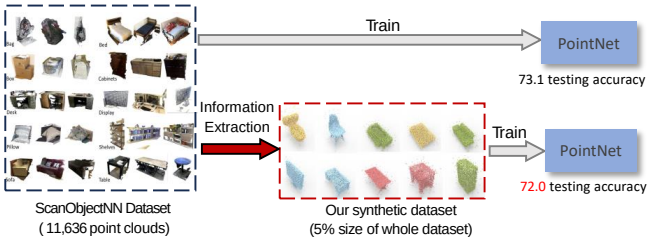

Figure 1: We investigate the possibility of synthesizing a much smaller and informative training dataset from a large point cloud dataset. With our generated synthetic training set, the trained model could achieve similar testing accuracy on the ScanObjectNN classification benchmark with only 5% size of the whole dataset.

network architecture is complex. Additionally, even though large volumes of point cloud data are requested by researchers to train neural networks, some data collectors may not want to make the data available to the general public for commercial reasons. For example, Shibuya et al. [51] have tried to transform point clouds into 3D line clouds to preserve point cloud data privacy.

Considering both the computation cost and data privacy, we investigate the feasibility of condensing a point cloud dataset into a compact set of representative synthetic point clouds, while preserving the majority of its information and minimizing performance degradation for classification task. Our objective is to create a synthetic dataset from a large dataset which should be significantly smaller in size compared to the original dataset, yet preserve the inherent characteristics of the dataset. Meanwhile, the generated synthetic dataset should be generalizable enough to train point cloud networks of different architectures. In this paper, we have explored that utilizing only a few synthetic point clouds generated with our method could result in plausible results when training the same model (Figure 1).

Similar tasks have been recently explored in 2D image areas, which are defined as image dataset condensation or distillation [55, 79, 81]. Most of these methods try to generate the synthetic image dataset to approximate the original training set via an iterative synthetic image optimization process. However, we find directly applying these methods to point cloud data is non-trivial where the generated synthetic point clouds are hard to converge to the optimal shapes by simply employing these methods. This challenge arises because 2D images are regular grids, allowing the optimization process to proceed effectively by simply altering pixel colors. In contrast, point clouds lack such regularity which allows points to move more freely. This complicates the optimization process because it becomes more difficult to determine the most effective optimization direction in 3D space.

We observe that the initialization of the synthetic point cloud dataset is a delicate factor which heavily impacts the synthetic set quality. In the 2D image field, most methods start with a synthetic dataset initialized with Gaussian noise or random samples, and often achieve satisfactory results after optimization. However, we find that due to the irregularity characteristic of point clouds, the initialization of the synthetic point cloud dataset is a delicate factor that significantly influences the synthetic set convergence. In other words, a strategic and well-considered initialization is essential for obtaining the more informative synthetic point cloud dataset.

To address this challenge, we structure our objective by introducing a two-stage approach. The first stage is centered on devising an effective method for initializing the synthetic point cloud dataset. Specifically, we design a strategy based on the mean of the nearest point cloud feature, which entails initializing our synthetic dataset with a carefully selected subset of point clouds. In the second stage, we propose a parameter-matching strategy to optimize the synthetic point cloud through moving the position of points. Specifically, given the same network initialization, the parameters of the deep network trained on the original and synthetic datasets should ideally be very similar. This parameter matching issue could be solved through an iterative gradient matching strategy where the gradients for the backpropagation of two networks should be similar in each training iteration. We illustrate our method pipeline in Figure 2. Instead of leveraging point cloud generative models[1, 27, 33, 65] to generate synthetic point clouds, we directly regard the input synthetic point clouds as learnable parameters, and treat the network weights as a differentiable function to optimize the synthetic input. In each training iteration, we compute the difference between the gradients from the original and synthetic point clouds and finally the synthetic point clouds could be optimized via moving the points.

Comprehensive experimental evaluation of our approach on the various synthetic and real-scanned point cloud classification datasets shows the effectiveness of our method, demonstrating that we could train point cloud networks with small synthetic dataset to significantly fasten the training process with acceptable performance drop. To further demonstrate the effectiveness of our method, we further test our approach on practical applications such as continual learning in our experiments.

In summary, our contributions are as follows: **1)** We define a new open problem in the point cloud field, exploring the possibility of synthesizing a smaller training dataset from a large point cloud dataset for efficient training. **2)** We introduce a two-stage solution, in which we propose a nearest-feature-mean based initialization strategy, followed by a parameter-matching strategy through gradient-based points moving. **3)** Extensive experiments on various classification benchmarks show the effectiveness of our synthetic dataset, demonstrating we could train point cloud networks with a small synthetic dataset to significantly fasten the training process with acceptable performance drop. **4)** We further evaluate our approach through applications in point cloud continual learning, demonstrating the significance of the new task and the efficacy of our proposed method.

2 RELATED WORK

Deep Learning on Point Cloud Analysis. The field of point cloud processing has undergone a significant transformation with the advent of deep learning technology[29, 66–69]. Traditional methods that relied on hand-crafted local descriptors [10, 17, 31] have been gradually replaced by learning-based methods. Qi et al. proposed PointNet [41] as a solution to the challenge posed by the disordered nature of point clouds, incorporating shared MLP and max pooling layers. PointNet++ [42] was subsequently introduced to enhance feature discrimination through hierarchical local feature

extraction. Wang et al. introduced DGCNN [56], which utilized the distance between each point in the feature space instead of the Euclidean distance to determine the area surrounding each point for graph feature extraction. Other works have employed convolution neural networks to learn point cloud local characteristics [3, 14, 18–20, 26, 28, 30, 38, 43, 48, 49, 52, 63]. Despite the success achieved by these methods in analyzing point clouds, many of them require the calculation of point-wise distance information to aggregate local characteristics, resulting in a higher computational cost than the convolution of conventional 2D images.

Generative models on point clouds. Our work is also closely related to generative models for point clouds, including point cloud generative adversarial networks [1, 27, 33], optimal transport [70], and probabilistic-based models [24, 37, 65]. PointFlow[65], DPF-Net[22], and SoftFlow[25] utilize Normalizing Flows, while SetVAE [23] employs VAEs to generate sets of point clouds. ShapeGF[4] learns the distributions of gradient fields that constitute shape surfaces, and SP-GAN[27] utilizes a prior based on a spherical point cloud. However, the aim of these point cloud generation methods is to generate realistic-looking point clouds capable of fooling humans, while the objective of our work is to create informative training examples that can be effectively utilized for the training of deep neural networks.

Dataset Compression & Condensation. Recent studies [21, 58, 60, 64, 77, 80, 82, 83] have employed learning-based methods to compress point cloud data. Given the disorderly nature of point clouds, current methods [39, 46] usually use a sophisticated data structure, like an octree, to arrange the unprocessed point cloud data. However, these methods focus on improving storage and transmission efficiency rather than training efficiency. In the 2D image community, some researchers aim to choose a subset of the entire training dataset, referred to as a "coreset", which can be used to achieve good performance during training. Most of these methods [11, 47] progressively select important data points based on heuristic criteria. Recent works [8, 54, 55, 78, 79, 81] have been proposed to generate datasets that are not limited to the original dataset, but applying these works to point clouds is non-trivial due to the irregularity of point clouds as discussed in the introduction. To this end, we explore the possibility of synthesizing a smaller, informative training dataset from a larger point cloud set.

3 PROBLEM DEFINITION

Given a training set $\mathcal{T}$ consisting of N point cloud and label pairs, $(x_i, y_i), i = 1, 2, \ldots N$, our objective is to generate a smaller set of synthetic point cloud and label pairs, $\mathcal{S} = (s_i, y_i)|i = 1, 2, \ldots M$, where $M(M \ll N)$ is much smaller than N. The ideal scenario would be a network trained on $\mathcal{S}$ converging faster and performing similarly to a network trained on $\mathcal{T}$.

The networks trained on $\mathcal{T}$ and $\mathcal{S}$ are denoted as $\phi_{\boldsymbol{\theta}^{\mathcal{T}}}$ and $\phi_{\boldsymbol{\theta}^{\mathcal{S}}}$, respectively, where $\boldsymbol{\theta}^{\mathcal{T}}$ and $\boldsymbol{\theta}^{\mathcal{S}}$ represent their respective parameters. The performance of each network can be defined as $\mathbb{E}_{\boldsymbol{x}\sim P_{\mathcal{D}}}\left[\ell\left(\phi_{\boldsymbol{\theta}^{\mathcal{T}}}(\boldsymbol{x}), y\right)\right]$ and $\mathbb{E}_{\boldsymbol{x}\sim P_{\mathcal{D}}}\left[\ell\left(\phi_{\boldsymbol{\theta}^{\mathcal{S}}}(\boldsymbol{x}), y\right)\right]$, where $P_{\mathcal{D}}$ represents the expected real data distribution and $\ell(.,.)$ is the classification loss function. Our objective can be formulated as:

$$\mathbb{E}_{\boldsymbol{x}\sim P_{\mathcal{D}}}\left[\ell\left(\phi_{\boldsymbol{\theta}}^{\mathcal{T}}(\boldsymbol{x}), y\right)\right] \simeq \mathbb{E}_{\boldsymbol{x}\sim P_{\mathcal{D}}}\left[\ell\left(\phi_{\boldsymbol{\theta}}^{\mathcal{S}}(\boldsymbol{x}), y\right)\right]. \quad (1)$$

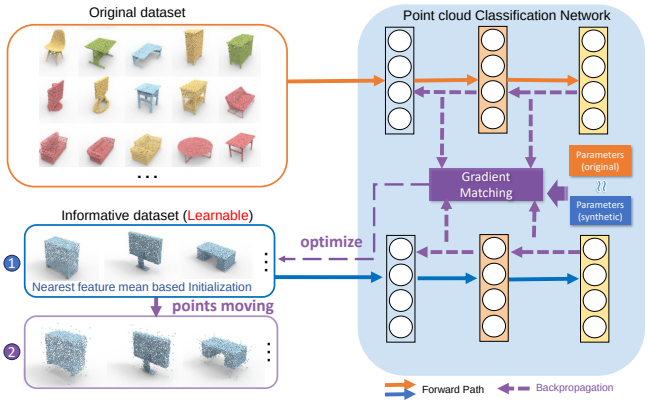

Figure 2: Method Overview. We learn an informative synthetic point cloud set which can get similar network parameters when a network is trained on it and the original dataset. We introduce a nearest feature mean based initialization strategy, and formulate our goal as an efficient gradient matching problem via points moving. The synthetic sets are regarded as learnable parameters and optimized iteratively.

In the experiments, the network performance is evaluated on the splited unseen testing set.

4 METHOD

To get the ideal $\mathcal{S}$ for a large point cloud dataset, we propose a two-stage solution. In the first stage, we introduce a nearest-feature-mean based initialization strategy, in which we initialize the synthetic point clouds with representive samples facilitating the following optimization process. In the second stage, we formulate our goal as a parameter-matching problem and further solve it by a gradient-matching strategy to iteratively optimize $\mathcal{S}$. We introduce our method details step-by-step in the following sections.

4.1 Nearest-feature-mean Based Initialization

As previously noted, the irregular nature of point clouds makes the initialization of $\mathcal{S}$ a sensitive factor in the subsequent optimization process. A good initialization can significantly enhance the convergence of $\mathcal{S}$ to get optimal performance. To define a good initial $\mathcal{S}$, inspired the heuristic sample selection method Herding [59], we consider $\mathcal{S}$ is well initialized if the initial $\mathcal{S}$ could approximate the overall distribution of the training set $\mathcal{T}$. To this end, we propose to initialize $\mathcal{S}$ by selecting the most representative samples in $\mathcal{T}$ based on the nearest feature mean.

Algorithm 1 describes our initialization strategy for $\mathcal{S}$. Specifically, $\mathcal{S}$ are initialized class by class. For the training set point clouds $\mathcal{T}_c$ belonging to class c, we extract the latent representation of each point cloud x in $\mathcal{T}_c$ through a feature function $\phi_f : x \rightarrow \mathbb{R}^d$. For the feature function ϕ_f, we train a point cloud classification network on $\mathcal{T}$, and use the last layer feature $f \in 1 \times D$ as the latent representation of each point cloud. Taking PointNet as an example, we use the feature after the max-pooling layer as the feature representation.Point clouds $s_1, \ldots, s_m$ are iteratively selected and accumulated until the desired total of m is reached. At each

iteration step, a point cloud from the current dataset $\mathcal{T}_c$ is incorporated into $\mathcal{S}_c$. Specifically, the chosen sample is the one that most closely aligns the average feature vector of all samples across the entire training set. To this end, $\mathcal{S}_c$ could finally approximate the distribution of the original set $\mathcal{T}_c$.

Algorithm 1 Synthetic Dataset Initialization Process

Input: point cloud dataset $\mathcal{T}_c = \{x_1, \ldots, x_n\}$ of class c
Input: m target number of the synthetic dataset $\mathcal{S}_c$ of class c
Require: feature function $\phi_f : x \to \mathbb{R}^d$
$\mu \leftarrow \frac{1}{n} \sum_{x \in \mathcal{T}_c} \phi_f(x)$ // current class feature mean
for $k = 1, \ldots, m$ **do**
$s_k \leftarrow \underset{x \in \mathcal{T}_c}{\operatorname{argmin}} \left\| \mu - \frac{1}{k} [\phi_f(x) + \sum_{j=1}^{k-1} \phi_f(s_j)] \right\|$
end for
$\mathcal{S}_c \leftarrow (s_1, \ldots, s_m)$
Output: Initialized synthetic dataset $\mathcal{S}_c$ of class c

4.2 Parameter Matching

To fulfill the objective outlined in Equation (1), it is crucial to not only achieve similar network output with minimal loss after training on $\mathcal{S}$ and $\mathcal{T}$, but also to obtain similar network parameters $\boldsymbol{\theta}^{\mathcal{S}} \approx \boldsymbol{\theta}^{\mathcal{T}}$, thus ensuring similar parameter optimization progress for improved generalization ability. Our goal can be expressed as:

$$\min_{\mathcal{S}} D\left(\boldsymbol{\theta}^{\mathcal{S}}, \boldsymbol{\theta}^{\mathcal{T}}\right), \tag{2}$$

$$\boldsymbol{\theta}^{\mathcal{S}}(\mathcal{S}) = \arg\min_{\boldsymbol{\theta}} \mathcal{L}^{\mathcal{S}}(\boldsymbol{\theta}) \tag{3}$$

$$\boldsymbol{\theta}^{\mathcal{T}}(\mathcal{T}) = \arg\min_{\boldsymbol{\theta}} \mathcal{L}^{\mathcal{T}}(\boldsymbol{\theta}) \tag{4}$$

where D denotes the distance function between the parameters of the two networks, and $\mathcal{L}$ denotes the classification function.

In the process of parameter learning, the final trained $\boldsymbol{\theta}^{\mathcal{T}}$ and the $\mathcal{S}$ is dependent on the initialization parameter $\boldsymbol{\theta}_0$ which is sampled from an initialization parameter distribution $P_{\boldsymbol{\theta}_0}$. Thus the above Equation 2, 3, 4 can be modified as:

$$\min_{\mathcal{S}} \mathrm{E}_{\boldsymbol{\theta}_0 \sim P_{\boldsymbol{\theta}_0}} \left[D\left(\boldsymbol{\theta}^{\mathcal{S}}\left(\boldsymbol{\theta}_0\right), \boldsymbol{\theta}^{\mathcal{T}}\left(\boldsymbol{\theta}_0\right)\right)\right] \tag{5}$$

$$\boldsymbol{\theta}^{\mathcal{S}}(\mathcal{S}) = \arg\min_{\boldsymbol{\theta}} \mathcal{L}^{\mathcal{S}}(\boldsymbol{\theta}(\boldsymbol{\theta}_0)) \tag{6}$$

$$\boldsymbol{\theta}^{\mathcal{T}}(\mathcal{T}) = \arg\min_{\boldsymbol{\theta}} \mathcal{L}^{\mathcal{T}}(\boldsymbol{\theta}(\boldsymbol{\theta}_0)) \tag{7}$$

Original point cloud set $\mathcal{T}$ is time-invariant during training, therefore we can first get trained $\boldsymbol{\theta}^{\mathcal{T}}$ in an offline manner, then $\boldsymbol{\theta}^{\mathcal{S}}$ can be learned with $\boldsymbol{\theta}^{\mathcal{T}}$ as the target.

The traditional approach to solve Equation 6, 7 involves implicit differentiation [44]. However, this can be complicated or expensive due to the formation of the derivative matrix. It becomes infeasible when the models are large. An alternative approach, back-optimization [15], is proposed where the network parameters ($\boldsymbol{\theta}$) are partially optimized according to the loss function:

$$\boldsymbol{\theta}^{\mathcal{S}}(\mathcal{S}) = Opt_{\boldsymbol{\theta}}\left(\mathcal{L}^{\mathcal{S}}\left(\boldsymbol{\theta}^{\mathcal{S}}\right), \mathcal{N}^{\mathcal{S}}\right) \tag{8}$$

$$\boldsymbol{\theta}^{\mathcal{T}}(\mathcal{T}) = Opt_{\boldsymbol{\theta}}\left(\mathcal{L}^{\mathcal{T}}\left(\boldsymbol{\theta}^{\mathcal{T}}\right), \mathcal{N}^{\mathcal{T}}\right) \tag{9}$$

where Opt means a specific optimization procedure with a fixed number of steps (N). However, using such a specific optimization procedure may not yield favorable results in our formulation. One reason is that the distance between the trained network parameters $\boldsymbol{\theta}^{\mathcal{T}}$ and the initial network parameters $\boldsymbol{\theta}^{S}$ may be too large, and there may be multiple local minima in the loss reduction process, making it difficult to pass through.

4.3 Gradient Matching via Critial Points Moving

Instead of matching the parameters of $\boldsymbol{\theta}^{\mathcal{S}}$ and trained $\boldsymbol{\theta}^{\mathcal{T}}$, we match the updated $\boldsymbol{\theta}_t^{\mathcal{S}}$ and $\boldsymbol{\theta}_t^{\mathcal{T}}$ at each training iteration t from the same initialization to overcome the problem of local minima traps. This converts the goal of parameter matching into smaller goals for each iteration which breaks down Equations (5, 6, 7) to:

$$\min_{\mathcal{S}} \mathrm{E}_{\boldsymbol{\theta}_0 \sim P_{\boldsymbol{\theta}_0}} \left[\sum_{t=0}^{T-1} D\left(\boldsymbol{\theta}_t^{\mathcal{S}}, \boldsymbol{\theta}_t^{\mathcal{T}}\right)\right] \tag{10}$$

$$\boldsymbol{\theta}_{t+1}^{\mathcal{S}}(\mathcal{S}) = Opt_{\boldsymbol{\theta}}\left(\mathcal{L}^{\mathcal{S}}\left(\boldsymbol{\theta}_t^{\mathcal{S}}\right), \mathcal{N}^{\mathcal{S}}\right) \tag{11}$$

$$\boldsymbol{\theta}_{t+1}^{\mathcal{T}}(\mathcal{T}) = Opt_{\boldsymbol{\theta}}\left(\mathcal{L}^{\mathcal{T}}\left(\boldsymbol{\theta}_t^{\mathcal{T}}\right), \mathcal{N}^{\mathcal{T}}\right) \tag{12}$$

where T denotes the total number of iterations. We hope that through this method of goal decomposition, we can train the network with the generated point cloud set $\mathcal{S}$ and original training set $\mathcal{T}$ to get the similar parameters after each iteration ($\boldsymbol{\theta}_t^{\mathcal{S}} \approx \boldsymbol{\theta}_t^{\mathcal{T}}$).

During the gradient descent optimization of iteration $t+1$, the update rule of Opt is:

$$\boldsymbol{\theta}_{t+1}^{\mathcal{S}} \leftarrow \boldsymbol{\theta}_t^{\mathcal{S}} - \eta_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}} \mathcal{L}^{\mathcal{S}}\left(\boldsymbol{\theta}_t^{\mathcal{S}}\right) \tag{13}$$

$$\boldsymbol{\theta}_{t+1}^{\mathcal{T}} \leftarrow \boldsymbol{\theta}_t^{\mathcal{T}} - \eta_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}} \mathcal{L}^{\mathcal{T}}\left(\boldsymbol{\theta}_t^{\mathcal{T}}\right) \tag{14}$$

where $\eta_{\boldsymbol{\theta}}$ refers to the learning rate. For a network designed for point clouds, e.g., PointNet, the assumption is that at each iteration t, $\boldsymbol{\theta}_t^{\mathcal{S}}$ and $\boldsymbol{\theta}_t^{\mathcal{T}}$ will become equal. This allows the synthetic input $\mathcal{S}$ to be optimized by minimizing the difference between the mean gradients, based on gradient matching theory [45, 50, 81]:

$$\min_{\mathcal{S}} \mathrm{E}_{\boldsymbol{\theta}_0 \sim P_{\boldsymbol{\theta}_0}} \left[\sum_{t=0}^{T-1} D\left(\nabla_{\boldsymbol{\theta}} \mathcal{L}^{\mathcal{S}}\left(\boldsymbol{\theta}_t\right), \nabla_{\boldsymbol{\theta}} \mathcal{L}^{\mathcal{T}}\left(\boldsymbol{\theta}_t\right)\right)\right] \tag{15}$$

$$\mathcal{L}^{\mathcal{S}}\left(\boldsymbol{\theta}_t^{\mathcal{S}}\right) = \frac{1}{M} \sum_{(s,y) \in \mathcal{S}} \ell(\phi_{\boldsymbol{\theta}_t}(s), y), \tag{16}$$

$$\mathcal{L}^{\mathcal{T}}\left(\boldsymbol{\theta}_t^{\mathcal{T}}\right) = \frac{1}{N} \sum_{(x,y) \in \mathcal{T}} \ell(\phi_{\boldsymbol{\theta}_t}(x), y), \tag{17}$$

where D denotes the Mean Squared Error (MSE) between the two gradients at each layer.

Instead of directly minimizing the difference between the parameters $\boldsymbol{\theta}_t^{\mathcal{S}}$ and $\boldsymbol{\theta}_t^{\mathcal{T}}$, the gradient distance is used as a simplified alternative to calculate $\mathcal{S}$. This is because the gradient directly reflects the change in parameters, avoiding the need for recalculating the previous parameters $\boldsymbol{\theta}_n$ ($n = 0, ..., n-1$) when the parameters are not significantly different, which leads to faster convergence and lower memory consumption.

The gradient matching loss $D(\cdot;\cdot)$ in Equation 15 measures the distance between two gradients. According to the network layer type and width, the size of gradients is different in each layer. Here, we flatten the gradient vectors to $1 \times D_i$ where D_i denotes the total gradient size in the i-th network layer. We compute the total layerwise gradient losses as:

$$d(\mathbf{G}^{\mathcal{S}}, \mathbf{G}^{\mathcal{T}}) = \sum_{i=1}^{\text{out}} \left(1 - \frac{\mathbf{G}_{\mathbf{i}}^{\mathcal{S}} \cdot \mathbf{G}_{\mathbf{i}}^{\mathcal{T}}}{\left\|\mathbf{G}_{\mathbf{i}}^{\mathcal{S}}\right\| \left\|\mathbf{G}_{\mathbf{i}}^{\mathcal{T}}\right\|} \right) \tag{18}$$

where $\mathbf{G}_{\mathbf{i}}^{\mathcal{S}}$ and $\mathbf{G}_{\mathbf{i}}^{\mathcal{T}}$ denote the gradients of $\mathcal{S}$ and $\mathcal{T}$ of the i-th layer, respectively. With gradient loss backpropagation, the points in $\mathcal{S}$ could move correspondingly to achieve similar gradients in each iteration. In our experiments, we find that those "critical points" move more significantly in the optimization process, which is discussed and illustrated in Sec. 6.2.

Additionally, the point cloud set $\mathcal{S}$ generated by our algorithm contains both point cloud data and the corresponding labels. Due to the complexity and learning difficulty, it is hard to optimize data and labels at the same time to get good results. Therefore, we fix a certain class label in a minibatch for generating the corresponding point cloud in a certain class.

5 TRAINING ALGORITHM

We illustrate our algorithm in Algorithm 2. We first choose a point cloud classification network backbone ϕ, e.g., PointNet, and initialize the synthetic training set $\mathcal{S}$ with the introduced nearest-feature-mean strategy. The entire algorithm involves an outer loop for network parameter θ optimization and an inner loop for synthetic data $\mathcal{S}$ optimization. For the outer loop, we re-randomly initialize θ_0 after T iterations for K times to increase the data generalization ability. In the inner loop, given the iteration t, we sample two training batches $B_c^{\mathcal{S}}$ and $B_c^{\mathcal{T}}$ from both the synthetic and original training set from the same class c. Then we compute the gradient matching loss between gradients $\nabla_{\theta}\mathcal{L}_c^{\mathcal{S}}(\theta_t)$ and $\nabla_{\theta}\mathcal{L}_c^{\mathcal{T}}(\theta_t)$ by applying gradient descent with learning rate $\eta_{\mathcal{S}}$. The gradient matching is performed class by class. After each iteration t, we optimize θ by computing the classification loss on the updated synthetic images $\mathcal{S}$. Though we could optimize $\mathcal{S}$ that are from multiple classes, we find that imitating the mean gradients of the data from a single class is easier compared to those of multiple classes.

6 EXPERIMENTS

Evaluation Datasets. We evaluate our proposed method on both synthetic and real-scanned classification datasets. **1) ModelNet.** ModelNet [61] is a widely used dataset for point cloud classification. Experiments are conducted on both ModelNet10 and ModelNet40. ModelNet10 contains 4,899 CAD models from 10 categories, with 3,991 shapes for training and 908 for testing. ModelNet40 contains 12,311 shapes from 40 categories, split into 9,843 for training and 2,468 for testing. **2) ShapeNet.** We also evaluate our methods on ShapeNetPart for classification, a subset of ShapeNet [9] with 16 categories containing 12,137 and 2,874 objects for training and testing, respectively. **3) ScanObjectNN.** ScanObjectNN [53] is a challenging real-scanned dataset containing 11,636 training objects and 2,814 testing objects that are categorized into 15 classes in the real world. We evaluated our method on ScanObjectNN without backgrounds.

Evaluated Networks. We use PointNet as the default classification backbone in all experiments if not specified. In the cross-architecture experiment, we test following 3D classifiers: PointNet [41], PointNet++ [42], DGCNN [56], PointNeXt [43], PointMLP [38], covering the pointwise-MLP-based, convolution-based, and graph-based methods.

Experimental Settings. We evaluate our method in three settings, generating a synthetic dataset containing one point cloud per category, 1% point clouds of the whole dataset, and 5% point clouds of the whole dataset. Each experiment involves two phases. First, we learn to synthesize a small synthetic set from a large real training set. Then we use the learned synthetic set to train randomly initialized neural networks and evaluate their performance on the real testing set. We use point clouds with 2048 points as inputs. We run each experiment 5 times and report the best performance of the learned synthetic set. The batch size $B_c^{\mathcal{T}}$ and $B_c^{\mathcal{S}}$ for sampling the original dataset and synthetic dataset are all set to 128 for PointNet and 32 for all other networks on a single A5000. We first use a learning rate of 0.0001 to optimize the synthetic point clouds for 1,000 iterations. After we get the synthetic dataset, we train new networks with a learning rate of 0.001 for 300 epochs with batch size 128.

Baselines We evaluate the performance of our method against baselines as follows:

- Random point clouds: It involves randomly selecting M point clouds from each class in the training set.
- Herding: It uses the Herding algorithm [2, 7, 11, 59], which is the same as our initialization strategy to select representative samples without further optimization.
- K-Center: It uses the K-Center algorithm [47], another coreset approach, which selects multiple center samples and minimizes the largest distance between a data sample and its nearest center.
- DC [81] is a 2D dataset condensation approach. The core idea is to generate the synthetic dataset in a way that produces the same path of model updates as if training on the original dataset, which is similar to our gradient matching progress.

6.1 Performance comparison

In our experiments, we evaluated the performance of our proposed method by comparing it with various baseline methods, as depicted in Table 1. We employed PointNet as the default network backbone. Our results show that though some coreset methods, such as Herding and K-Center, outperformed random selection, our approach demonstrated a clear advantage over these selection methods as it was not restricted to a subset of the original dataset. Our approach outperforms those directly applying 2D dataset condensation methods, such as DC and DM, indicating the better convergence of our synthetic dataset. Our synthetic data achieved 73.5% accuracy using only 10 point clouds on the ModelNet10 dataset and 83.5% accuracy with only 5% of the point clouds. Our method also performed well on the ModelNet40 and ShapeNet datasets. On the real-world ScanObjectNN dataset, our method achieved 72.0 % accuracy using only 5% of the data, which is almost the same as the performance achieved using the entire dataset (73.1% accuracy).

Algorithm 2 Synthesizing point cloud training dataset.

Input: Original Training dataset $\mathcal{T}$
Initialization: A point cloud classification network ϕ, synthetic set $\mathcal{S}$ initialized with nearest-feature-mean strategy for C classes, outer loop training iterations K, inner loop training iterations T, learning rate η.

for outer iteration k in K **do**
  Re-randomly initializing network parameters θ_0
  for inner iteration t in T **do**
    for class c in C **do**
      Sample a minibatch pair $B_c^{\mathcal{T}} \sim \mathcal{T}$ and $B_c^{\mathcal{S}} \sim \mathcal{S}$ ▷ $B_c^{\mathcal{T}}$ and $B_c^{\mathcal{S}}$ are of the same class c.
      Compute the classification loss $\mathcal{L}_c^{\mathcal{T}} = \frac{1}{|B_c^{\mathcal{T}}|} \sum_{(\boldsymbol{x},y)\in B_c^{\mathcal{T}}} \ell\left(\phi_{\boldsymbol{\theta}_t}(\boldsymbol{x}), y\right)$ and $\mathcal{L}_c^{\mathcal{S}} = \frac{1}{|B_c^{\mathcal{S}}|} \sum_{(\boldsymbol{s},y)\in B_c^{\mathcal{S}}} \ell\left(\phi_{\boldsymbol{\theta}_t}(\boldsymbol{s}), y\right)$
      Update $\mathcal{S}_c \leftarrow \text{Opt}\, g_{\mathcal{S}}\left(D\left(\nabla_{\boldsymbol{\theta}}\mathcal{L}_c^{\mathcal{S}}(\boldsymbol{\theta}_t), \nabla_{\boldsymbol{\theta}}\mathcal{L}_c^{\mathcal{T}}(\boldsymbol{\theta}_t)\right), \eta_{\mathcal{S}}\right)$
    end for
    Update $\boldsymbol{\theta}_{t+1} \leftarrow \text{Opt}\log_{\boldsymbol{\theta}}\left(\mathcal{L}^{\mathcal{S}}(\boldsymbol{\theta}_t), \eta_{\boldsymbol{\theta}}\right)$ ▷ Use the whole $\mathcal{S}$
  end for
end for

Output: Synthetic training set $\mathcal{S}$

	Subset size	Ratio %	Training time (s)	Random	Herding	K-Center	DC	Ours	Whole Dataset
ModelNet10	10	0.25	1.8	39.3	69.6	69.6	71.5	**73.5**	91.9
	40	1	8.6	73.4	75.6	68.7	78.6	**79.0**	
	200	5	40.2	76.1	79.8	70.5	82.9	**83.5**	
ModelNet40	40	0.4	7.0	35.3	45.3	45.3	47.5	**49.5**	87.7
	100	1	16.9	47.7	47.3	29.3	57.1	**57.9**	
	500	5	82.1	59.6	58.0	38.3	70.8	**71.1**	
ShapeNet	16	0.13	3.0	51.4	58.7	58.7	63.5	**65.5**	97.1
	122	1	22.9	76.9	77.5	72.5	80.9	**81.1**	
	607	5	109.2	89.3	84.5	80.4	86.7	**88.4**	
ScanObjectNN	15	0.12	3.0	23.9	31.0	31.0	32.6	**34.7**	73.1
	116	1	20.9	42.7	45.7	33.8	52.9	**53.1**	
	582	5	103.6	59.8	59.4	61.2	69.0	**72.0**	

Table 1: The testing accuracy (%) of different methods on different datasets with PointNet for training and testing.

	ModelNet10				ModelNet40				ShapeNet				ScanObjectNN			
Samples per class	1	1%	5%	whole set	1	1%	5%	whole set	1	1%	5%	whole set	1	1%	5%	whole set
Learning synthetic set	1510s	1816s	2898s	-	1789s	2203s	4782s	-	1651s	2325s	5338s	-	1630s	2286s	5270s	-
Training network	5s	21s	105s	2313s	20s	56s	271s	5320s	8s	67s	312s	5461s	8s	63s	311s	5295s

Table 2: Time consumption of learning the synthetic data and training PointNet with the synthetic data. The synthetic point cloud dataset only needs to be generated once.

6.2 Visualization of synthetic point clouds

We visualize the learned synthetic point clouds of ModelNet10 and ModelNet40 in Figure 3. We find that the synthetic point clouds are visually recognizable, and we further conduct further analysis to explore why our generated synthetic point clouds yield better training performance.

In the visualization from Figure 3, we observe that the network changes some of the point positions, which leads us to consider whether our method is modifying some of the "critical points" of the original point clouds, which is discussed in PointNet and PointNet++ [41, 42]. Using PointNet as the backbone, we illustrate the critical points that activate the final max-pooling layer. We visualize the synthetic point clouds and their corresponding critical points at different epochs in Figure 4. It can be seen that after optimization, the overall point clouds do not change much, but the critical points have been significantly altered, which verifies our hypothesis.

6.3 Synthetic set initialization strategies

In our default settings, we initialize our synthetic set with our proposed nearest-feature-mean based strategy. Here, we make a comparison via initializing the point clouds with random samples. In Table 3, we show the averaged results with 20 different random seeds. We could observe that we could get better performance with our initialized strategy on all datasets, indicating that the synthetic dataset $\mathcal{S}$ could converge better with our initialization.

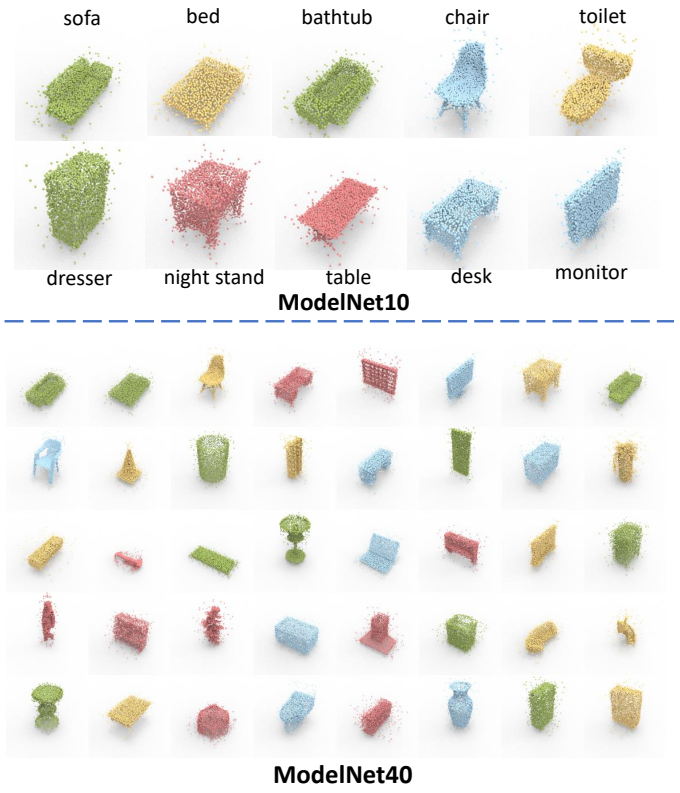

Figure 3: Visualization of the learned synthetic data on ModelNet10 and ModelNet40 with one point cloud per class.

	ModelNet10			ModelNet40			ShapeNet			ScanObjectNN		
Samples	1	%1	5%	1	%1	5%	1	%1	5%	1	%1	5%
Random	71.5	80.0	83.1	47.5	57.9	71.1	63.5	80.9	88.4	31.7	52.9	69.2
Ours	**73.5**	**79.0**	**83.5**	**50.5**	**60.9**	**72.1**	**65.5**	**81.1**	**90.4**	**34.2**	**53.1**	**72.0**

Table 3: Synthetic set initialized with random samples and our initialization strategy.

6.4 Time consumption analysis

We give a time consumption analysis in Table 2 conducted on a single A5000. We report the time for learning the synthetic set with different numbers of point clouds per class, and the time for training PointNet with the synthetic set. Table 2 shows that the time for learning the synthetic set with 1 sample, 1% samples, and 5% samples per class for 1,000 iterations is less than training on the whole set for 300 epochs. With a smaller training set size, the training efficiency has been improved significantly. It is worth noting that **the synthetic point cloud dataset only needs to be generated once and then distributed to different users for efficient training without re-generation**.

6.5 Cross-architecture generalization

It is expected that the synthetic dataset will also be effective in training networks with different architectures. Thus we could generate a synthetic set using one backbone to gain generalization performance to different networks. We verify this through conducting two experiments that aim to evaluate the cross-architecture generalization.

To begin with, we analyze the synthetic datasets generated through three different PointNet architectures with different layer numbers and feature dimensions. Specifically, we employ three distinct PointNet configurations, namely the standard PointNet (which involves 5 shared Multi-Layer Perceptron (MLP) layers for feature extraction), the PointNet-small (which involves 3 shared MLP layers), and the PointNet-large (which involves 7 shared MLP layers). Specifically, we use shared MLP layers with (64, 128, 512) channels for PointNet-small, (64, 64, 64 128, 1024) channels for standard PointNet, and (64, 64, 64 128, 256, 512, 1024) channels for PointNet-large. Cross-testing is performed across these architectures, with results displayed in Table 4. It indicates that the generated synthetic datasets show commendable generalization performance on all of the PointNet architectures.

Train/Test	PointNet	PointNet-small	PointNet-large
PointNet	73.5	73.5	74.3
PointNet-small	71.2	70.6	71.9
PointNet-large	71.6	70.8	73.9

Table 4: Cross-architecture testing accuracy (%) for one point cloud per class on ModelNet10.

Nest, we generate synthetic datasets using a range of networks including PointNet, PointNet++, DGCNN, PointNeXt, PointMLP, and VoxelNet, all configured with default settings for classification. Cross-testing was performed across these networks and the results were evaluated on ModelNet10 with one point cloud per category, which are presented in Table 5. Results indicate that the synthetic dataset generated by PointNet produces the best performance on all the different architectures. **We hypothesize this is because all other baselines involve sorting and ranking operations (e.g., TopK operators in Pytorch) for performing local convolution, such as ball query (PointNet++) and group convolution (DGCNN) operators.** These kinds of operations are demonstrated as not gradient backpropagating friendly in [12]. Therefore, leveraging PointNet as the classification backbone, which is without sorting and ranking operations, could better optimize the synthetic dataset. Furthermore, the generated synthetic set using PointNet also performs well on other baselines, demonstrating that it could well represent the original set and is generalized with different architectures. Note that we set a small learning rate to 0.00001 for all baselines except PointNet, as we found that a larger learning rate will make the results collapes.

Train/Test	PointNet	PointNet++	DGCNN	PointNeXt	PointMLP	VoxelNet
PointNet	**73.5**	**68.3**	**71.2**	**72.3**	**73.9**	**70.2**
PointNet++	58.3	62.9	58.6	62.9	61.6	54.9
DGCNN	64.2	61.4	62.8	61.0	59.9	56.8
PointNeXt	61.3	63.4	60.8	63.8	60.6	58.7
PointMLP	62.3	60.4	61.4	59.6	61.0	59.0

Table 5: Cross-architecture testing accuracy (%) with different networks for one point cloud per class on ModelNet10.

6.6 Application

As outlined in our introduction, the synthesized compact point cloud dataset can be utilized in two key applications: 1) It can facilitate continual learning in point cloud field, when processing new,

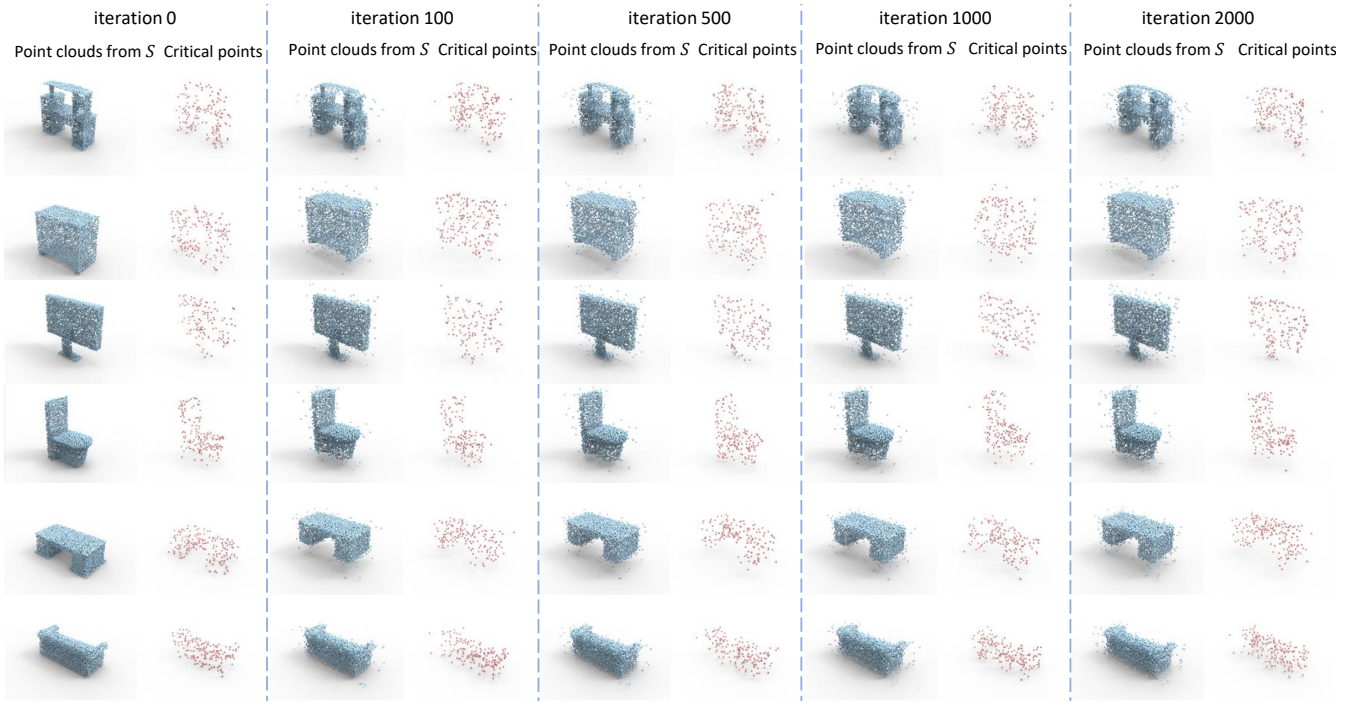

Figure 4: Visualization moving process of the critical points which activate the max-pooling layer of PointNet.

previously unseen classes of point cloud data. 2) Privacy concerns associated with the original data can be mitigated by sharing the generated datasets rather than the actual ones.

As outlined in our introduction, the synthesized compact point cloud dataset can be utilized in continual learning application. It can facilitate continual learning in point cloud field, when processing new, previously unseen classes of point cloud data.

Point Cloud Continual Learning. We adapted our method for continual learning to incrementally learn point clouds from new classes while preserving the performance on existing classes. We use the setting in [40] as our baseline, in which it progressively stores training samples in memory to maintain class balance. We replaced its Herding sample selection process by using our generated synthetic point cloud dataset. This includes generating synthetic point clouds from current classes and storing them in memory. We then assess the model performance in a task-incremental learning scenario using the ModelNet40 dataset of 40 classes.

Our method is benchmarked against Random selection, Herding, and DC, using a memory budget of 5 synthetic point clouds per class on the ModelNet40 dataset. We structure the learning into 5 and 10 steps, dividing the 40 classes into equal parts for each step. Utilizing the PointNet backbone, our approach consistently outperformed the others in both 5-step and 10-step learning scenarios which is shown in Figure 5. This indicates that our condensed dataset is more representative of the original dataset.

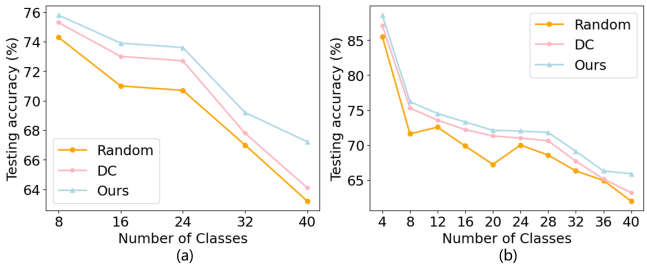

Figure 5: (a) 5-step and (b) 10-step class-continual learning on ModelNet40.

7 CONCLUSION AND FUTURE WORKS

The purpose of our paper is to explore the feasibility of training a point cloud classification network to obtain high accuracy using only a reduced number of synthetic point clouds derived from a large dataset. This objective is achieved by formulating it as a parameter-matching problem where a network can attain comparable network parameters after being trained on both the original and the generated synthetic point clouds. To resolve this challenge, a nearest-feature-mean based initialization strategy and effective iterative gradient matching approach are introduced. Experimental results demonstrate the effectiveness of the proposed method across various point cloud classification datasets and its application in point cloud continual learning and data privacy protection. In future work, besides the classification task, we aim to explore the application to more complex point cloud tasks.

8 ACKNOWLEDGEMENT

This research was supported by the National Natural Science Foundation of China (NSFC) under Grant U22A2094. This research was also supported by the advanced computing resources provided by the Supercomputing Center of the USTC.

REFERENCES

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. 2018. Learning representations and generative models for 3d point clouds. In *International conference on machine learning*. PMLR, 40–49.

[2] Eden Belouadah and Adrian Popescu. 2020. Scail: Classifier weights scaling for class incremental learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 1266–1275.

[3] Yizhak Ben-Shabat, Michael Lindenbaum, and Anath Fischer. 2018. 3dmfv: Three-dimensional point cloud classification in real-time using convolutional neural networks. *IEEE Robotics and Automation Letters* 3, 4 (2018), 3145–3152.

[4] Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan. 2020. Learning gradient fields for shape generation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*. Springer, 364–381.

[5] Tuo Cao, Fei Luo, Yanping Fu, Wenxiao Zhang, Shengjie Zheng, and Chunxia Xiao. 2022. DGECN: A depth-guided edge convolutional network for end-to-end 6D pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3783–3792.

[6] Tuo Cao, Wenxiao Zhang, Yanping Fu, Shengjie Zheng, Fei Luo, and Chunxia Xiao. 2023. Dgecn++: A depth-guided edge convolutional network for end-to-end 6d pose estimation via attention mechanism. *IEEE Transactions on Circuits and Systems for Video Technology* (2023).

[7] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. 2018. End-to-end incremental learning. In *Proceedings of the European conference on computer vision (ECCV)*. 233–248.

[8] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A. Efros, and Jun-Yan Zhu. 2022. Dataset Distillation by Matching Training Trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 4750–4759.

[9] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. 2015. *ShapeNet: An Information-Rich 3D Model Repository*. Technical Report 1. Stanford University — Princeton University — Toyota Technological Institute at Chicago.

[10] Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. 2003. On visual similarity based 3D model retrieval. In *Computer graphics forum*, Vol. 22. Wiley Online Library, 223–232.

[11] Yutian Chen, Max Welling, and Alex Smola. 2010. Super-samples from kernel herding. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*. 109–116.

[12] Marco Cuturi, Olivier Teboul, and Jean-Philippe Vert. 2019. Differentiable ranking and sorting using optimal transport. *Advances in neural information processing systems* 32 (2019).

[13] Donglin Di, Jiahui Yang, Chaofan Luo, Zhou Xue, Wei Chen, Xun Yang, and Yue Gao. 2024. Hyper-3DG: Text-to-3D Gaussian Generation via Hypergraph. *arXiv preprint arXiv:2403.09236* (2024).

[14] Miguel Dominguez, Rohan Dhamdhere, Atir Petkar, Saloni Jain, Shagan Sah, and Raymond Ptucha. 2018. General-purpose deep point cloud feature extractor. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 1972–1981.

[15] Justin Domke. 2012. Generic methods for optimization-based modeling. In *Artificial Intelligence and Statistics*. PMLR, 318–326.

[16] Dan Guo, Kun Li, Bin Hu, Yan Zhang, and Meng Wang. 2024. Benchmarking Micro-action Recognition: Dataset, Methods, and Applications. *IEEE Transactions on Circuits and Systems for Video Technology* 34, 7 (2024), 6238–6252. https://doi.org/10.1109/TCSVT.2024.3358415

[17] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, and Jianwei Wan. 2014. 3D object recognition in cluttered scenes with local surface features: A survey. *IEEE transactions on pattern analysis and machine intelligence* 36, 11 (2014), 2270–2287.

[18] Pedro Hermosilla, Tobias Ritschel, Pere-Pau Vázquez, Àlvar Vinacua, and Timo Ropinski. 2018. Monte carlo convolution for learning on non-uniformly sampled point clouds. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–12.

[19] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. 2018. Pointwise convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 984–993.

[20] Siyuan Huang, Bo Zhang, Botian Shi, Hongsheng Li, Yikang Li, and Peng Gao. 2023. Sug: Single-dataset unified generalization for 3d point cloud classification. In *Proceedings of the 31st ACM International Conference on Multimedia*. 8644–8652.

[21] Tianxin Huang and Yong Liu. 2019. 3d point cloud geometry compression on deep learning. In *Proceedings of the 27th ACM international conference on multimedia*. 890–898.

[22] Hyeongju Kim, Hyeonseung Lee, Woo Hyun Kang, Joun Yeop Lee, and Nam Soo Kim. 2020. Softflow: Probabilistic framework for normalizing flow on manifolds. *Advances in Neural Information Processing Systems* 33 (2020), 16388–16397.

[23] Jinwoo Kim, Jaehoon Yoo, Juho Lee, and Seunghoon Hong. 2021. Setvae: Learning hierarchical composition for generative modeling of set-structured data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 15059–15068.

[24] Takumi Kimura, Takashi Matsubara, and Kuniaki Uehara. 2022. Topology-Aware Flow-Based Point Cloud Generation. *IEEE Transactions on Circuits and Systems for Video Technology* 32, 11 (2022), 7967–7982. https://doi.org/10.1109/TCSVT.2022.3181212

[25] Roman Klokov, Edmond Boyer, and Jakob Verbeek. 2020. Discrete point flow networks for efficient point cloud generation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII 16*. Springer, 694–710.

[26] Jiaxin Li, Ben M Chen, and Gim Hee Lee. 2018. So-net: Self-organizing network for point cloud analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 9397–9406.

[27] Ruihui Li, Xianzhi Li, Ka-Hei Hui, and Chi-Wing Fu. 2021. SP-GAN: Sphere-guided 3D shape generation and manipulation. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–12.

[28] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. 2018. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems* 31 (2018).

[29] Yicong Li, Xun Yang, An Zhang, Chun Feng, Xiang Wang, and Tat-Seng Chua. 2023. Redundancy-aware transformer for video question answering. In *Proceedings of the 31st ACM International Conference on Multimedia*. 3172–3180.

[30] Xuanxiang Lin, Ke Chen, and Kui Jia. 2021. Object point cloud classification via poly-convolutional architecture search. In *Proceedings of the 29th ACM International Conference on Multimedia*. 807–815.

[31] Haibin Ling and David W Jacobs. 2007. Shape classification using the inner-distance. *IEEE transactions on pattern analysis and machine intelligence* 29, 2 (2007), 286–299.

[32] Liu Liu, Jianming Du, Hao Wu, Xun Yang, Zhenguang Liu, Richang Hong, and Meng Wang. 2023. Category-level articulated object 9d pose estimation via reinforcement learning. In *Proceedings of the 31st ACM International Conference on Multimedia*. 728–736.

[33] Xinyue Liu, Xiangnan Kong, Lei Liu, and Kuorong Chiang. 2018. TreeGAN: syntax-aware sequence generation with generative adversarial networks. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1140–1145.

[34] Chen Long, Wenxiao Zhang, Zhe Chen, Haiping Wang, Yuan Liu, Peiling Tong, Zhen Cao, Zhen Dong, and Bisheng Yang. 2024. SparseDC: Depth Completion from sparse and non-uniform inputs. *Information Fusion* 110 (2024), 102470.

[35] Chen Long, WenXiao Zhang, Ruihui Li, Hao Wang, Zhen Dong, and Bisheng Yang. 2022. Pc2-pu: Patch correlation and point correlation for effective point cloud upsampling. In *Proceedings of the 30th ACM International Conference on Multimedia*. 2191–2201.

[36] Chaofan Luo, Donglin Di, Yongjia Ma, Zhou Xue, Chen Wei, Xun Yang, and Yebin Liu. 2024. TrAME: Trajectory-Anchored Multi-View Editing for Text-Guided 3D Gaussian Splatting Manipulation. *arXiv preprint arXiv:2407.02034* (2024).

[37] Shitong Luo and Wei Hu. 2021. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2837–2845.

[38] Xu Ma, Can Qin, Haoxuan You, Haoxi Ran, and Yun Fu. 2022. Rethinking network design and local geometry in point cloud: A simple residual MLP framework. *arXiv preprint arXiv:2202.07123* (2022).

[39] Donald Meagher. 1982. Geometric modeling using octree encoding. *Computer graphics and image processing* 19, 2 (1982), 129–147.

[40] Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. 2020. Gdumb: A simple approach that questions our progress in continual learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. Springer, 524–540.

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

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

[43] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. 2022. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. *Advances in Neural Information Processing Systems* 35 (2022), 23192–23204.

[44] Kegan GG Samuel and Marshall F Tappen. 2009. Learning optimized MAP estimates in continuously-valued MRF models. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 477–484.

[45] Mert Bulent Sariyildiz and Ramazan Gokberk Cinbis. 2019. Gradient matching generative networks for zero-shot learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2168–2178.

[46] Sebastian Schwarz, Marius Preda, Vittorio Baroncini, Madhukar Budagavi, Pablo Cesar, Philip A Chou, Robert A Cohen, Maja Krivokuća, Sébastien Lasserre, Zhu Li, et al. 2018. Emerging MPEG standards for point cloud compression.

IEEE Journal on Emerging and Selected Topics in Circuits and Systems 9, 1 (2018), 133–148.

[47] Ozan Sener and Silvio Savarese. 2018. Active Learning for Convolutional Neural Networks: A Core-Set Approach. In *International Conference on Learning Representations*.

[48] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. 2018. Mining point cloud local structures by kernel correlation and graph pooling. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4548–4557.

[49] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. 2020. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10529–10538.

[50] Yuge Shi, Jeffrey Seely, Philip Torr, N Siddharth, Awni Hannun, Nicolas Usunier, and Gabriel Synnaeve. 2021. Gradient Matching for Domain Generalization. In *International Conference on Learning Representations*.

[51] Mikiya Shibuya, Shinya Sumikura, and Ken Sakurada. 2020. Privacy preserving visual SLAM. In *European Conference on Computer Vision*. Springer, 102–118.

[52] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. 2019. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision*. 6411–6420.

[53] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. 2019. 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*. 1588–1597.

[54] Kai Wang, Bo Zhao, Xiangyu Peng, Zheng Zhu, Shuo Yang, Shuo Wang, Guan Huang, Hakan Bilen, Xinchao Wang, and Yang You. 2022. CAFE: Learning to Condense Dataset by Aligning Features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 12196–12205.

[55] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. 2019. Dataset Distillation. In *International Conference on Learning Representations*.

[56] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. 2019. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)* 38, 5 (2019), 1–12.

[57] Ziqi Wang, Fei Luo, Xiaoxiao Long, Wenxiao Zhang, and Chunxia Xiao. 2023. Learning long-range information with dual-scale transformers for indoor scene completion. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 18523–18533.

[58] Ziwei Wei, Benben Niu, Haodong Xiao, and Yun He. 2023. Isolated Points Prediction via Deep Neural Network on Point Cloud Lossless Geometry Compression. *IEEE Transactions on Circuits and Systems for Video Technology* 33, 1 (2023), 407–420. https://doi.org/10.1109/TCSVT.2022.3197370

[59] Max Welling. 2009. Herding dynamical weights to learn. In *Proceedings of the 26th annual international conference on machine learning*. 1121–1128.

[60] Louis Wiesmann, Andres Milioto, Xieyuanli Chen, Cyrill Stachniss, and Jens Behley. 2021. Deep compression for dense point cloud maps. *IEEE Robotics and Automation Letters* 6, 2 (2021), 2060–2067.

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

[62] Hang Xu, Chen Long, Wenxiao Zhang, Yuan Liu, Zhen Cao, Zhen Dong, and Bisheng Yang. 2024. Explicitly Guided Information Interaction Network for Cross-modal Point Cloud Completion. *arXiv preprint arXiv:2407.02887* (2024).

[63] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. 2018. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 87–102.

[64] Wei Yan, Shan Liu, Thomas H Li, Zhu Li, Ge Li, et al. 2019. Deep autoencoder-based lossy geometry compression for point clouds. *arXiv preprint arXiv:1905.03691* (2019).

[65] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. 2019. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4541–4550.

[66] Xun Yang, Tianyu Chang, Tianzhu Zhang, Shanshan Wang, Richang Hong, and Meng Wang. 2024. Learning Hierarchical Visual Transformation for Domain Generalizable Visual Matching and Recognition. *International Journal of Computer Vision* (2024), 1–27.

[67] Xun Yang, Fuli Feng, Wei Ji, Meng Wang, and Tat-Seng Chua. 2021. Deconfounded video moment retrieval with causal intervention. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1–10.

[68] Xun Yang, Shanshan Wang, Jian Dong, Jianfeng Dong, Meng Wang, and Tat-Seng Chua. 2022. Video moment retrieval with cross-modal neural architecture search. *IEEE Transactions on Image Processing* 31 (2022), 1204–1216.

[69] Xun Yang, Jianming Zeng, Dan Guo, Shanshan Wang, Jianfeng Dong, and Meng Wang. 2024. Robust Video Question Answering via Contrastive Cross-Modality Representation Learning. *SCIENCE CHINA Information Sciences* (2024).

[70] Ruonan Zhang, Jingyi Chen, Wei Gao, Ge Li, and Thomas H. Li. 2022. PointOT: Interpretable Geometry-Inspired Point Cloud Generative Model via Optimal Transport. *IEEE Transactions on Circuits and Systems for Video Technology* 32, 10 (2022), 6792–6806. https://doi.org/10.1109/TCSVT.2022.3170588

[71] Wenxiao Zhang, Zhen Dong, Jun Liu, Qingan Yan, Chunxia Xiao, et al. 2022. Point Cloud Completion Via Skeleton-Detail Transformer. *IEEE Transactions on Visualization and Computer Graphics* (2022).

[72] Wenxiao Zhang, Chengjiang Long, Qingan Yan, Alix LH Chow, and Chunxia Xiao. 2020. Multi-stage point completion network with critical set supervision. *Computer Aided Geometric Design* 82 (2020), 101925.

[73] Wenxiao Zhang, Hossein Rahmani, Xun Yang, and Jun Liu. [n. d.]. Reverse2Complete: Unpaired Multimodal Point Cloud Completion via Guided Diffusion. In *ACM Multimedia 2024*.

[74] Wenxiao Zhang and Chunxia Xiao. 2019. PCAN: 3D attention map learning using contextual information for point cloud based retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12436–12445.

[75] Wenxiao Zhang, Qingan Yan, and Chunxia Xiao. 2020. Detail preserved point cloud completion via separated feature aggregation. In *European Conference on Computer Vision*. Springer, 512–528.

[76] Wenxiao Zhang, Huajian Zhou, Zhen Dong, Qingan Yan, and Chunxia Xiao. 2022. Rank-PointRetrieval: Reranking Point Cloud Retrieval via a Visually Consistent Registration Evaluation. *IEEE Transactions on Visualization and Computer Graphics* (2022).

[77] Xiang Zhang and Wen Gao. 2021. Adaptive Geometry Partition for Point Cloud Compression. *IEEE Transactions on Circuits and Systems for Video Technology* 31, 12 (2021), 4561–4574. https://doi.org/10.1109/TCSVT.2021.3101807

[78] Bo Zhao and Hakan Bilen. 2021. Dataset condensation with differentiable siamese augmentation. In *International Conference on Machine Learning*. PMLR, 12674–12685.

[79] Bo Zhao and Hakan Bilen. 2023. Dataset condensation with distribution matching. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 6514–6523.

[80] Baoquan Zhao, Weisi Lin, and Chenlei Lv. 2021. Fine-Grained Patch Segmentation and Rasterization for 3-D Point Cloud Attribute Compression. *IEEE Transactions on Circuits and Systems for Video Technology* 31, 12 (2021), 4590–4602. https://doi.org/10.1109/TCSVT.2021.3101126

[81] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. 2021. Dataset Condensation with Gradient Matching. *ICLR* 1, 2 (2021), 3.

[82] Lili Zhao, Kai-Kuang Ma, Zhili Liu, Qian Yin, and Jianwen Chen. 2022. Real-Time Scene-Aware LiDAR Point Cloud Compression Using Semantic Prior Representation. *IEEE Transactions on Circuits and Systems for Video Technology* 32, 8 (2022), 5623–5637. https://doi.org/10.1109/TCSVT.2022.3145513

[83] Wenjie Zhu, Yiling Xu, Dandan Ding, Zhan Ma, and Mike Nilsson. 2021. Lossy Point Cloud Geometry Compression via Region-Wise Processing. *IEEE Transactions on Circuits and Systems for Video Technology* 31, 12 (2021), 4575–4589. https://doi.org/10.1109/TCSVT.2021.3101852