MOPS-Net: A Matrix Optimization-driven Network for Task-Oriented 3D Point Cloud Downsampling

Yue Qian, Junhui Hou, *Senior Member, IEEE,*, Yiming Zeng, Qijian Zhang, Sam Kwong, *Fellow, IEEE,* and Ying He

Abstract—Downsampling is a commonly-used technique in 3D point cloud processing for saving storage space, transmission bandwidth and computational complexity. The classic downsampling methods, such as farthest point sampling and Poisson disk sampling, though widely used for decades, are independent of the subsequent applications, since the downsampled point cloud may compromise their performance severely. This paper explores the problem of task-oriented downsampling, which aims to downsample a point cloud and maintain the performance of subsequent applications as much as possible. We propose *MOPS-Net*, a novel end-to-end deep neural network which is designed from the perspective of matrix optimization, making it fundamentally different from the existing deep learning-based methods. Due to its *discrete* and *combinatorial* nature, it is difficult to solve the downsampling problem directly. To tackle this challenge, we relax the binary constraint of each variable and lift 3D points to a higher dimensional feature space, in which a constrained and differentiable matrix optimization problem with an implicit objective function is formulated. We then propose a deep neural network architecture to mimic the matrix optimization problem by exploring both the local and the global structures of the input data. With a task network, MOPS-Net can be end-to-end trained. Moreover, it is permutation-invariant, making it robust to input data. We also extend MOPS-Net such that a single network after one-time training is capable of handling arbitrary downsampling ratios. Extensive experimental results show that MOPS-Net can achieve favorable performance against state-of-the-art deep learning-based method over various tasks, including point cloud classification, retrieval and reconstruction.

Index Terms—Point cloud, Sampling, Optimization, Deep learning, Classification, Reconstruction.

1 Introduction

With recent advances in three-dimensional (3D) sensing technology (e.g., LiDAR scanning devices), 3D point clouds can be easily obtained. Compared with other 3D representations such as multi-view images, voxel grids and polygonal meshes, point clouds are a raw 3D representation, containing only 3D samples which are located on the scanned surface. Powered by deep learning techniques, the performance of many point cloud applications, such as classification, segmentation and reconstruction, has been improved significantly in recent years. However, processing large-scale and/or dense 3D point clouds is still challenging due to the cost of computation, storage, and communication load.

Point cloud downsampling is a popular and effective technique to reduce information redundancy, hereby improving the runtime performance of the downstream applications and saving storage space and transmission bandwidth. The classic downsampling approaches such as farthest point sampling (FPS) [53] and Poisson disk sampling (PDS) [1] iteratively generate uniformly distributed samples on the input shape, and thus they can preserve the geometry well. Such sampling methods, however, focus on reducing the geometry loss only and are completely independent of the downstream applications. As a result, the downsampled point clouds may degrade the performance of the subsequent applications severely.

A better way for downsampling is to generate samples that optimize the performance of a particular task, i.e., the resulting sparse point clouds will maintain the task performance as much as possible. Due to the task centric nature, we call it *task-oriented downsampling*. Moreover, an effective downsampling method should allow the user to freely specify the downsampling ratio to balance the task performance and computation efficiency.

As the deep learning technologies have proven effective in point cloud classification [2], [6], [7], [8], [42] and segmentation [13], [14], [15], [52], it is highly desired to combine downsampling methods with the deep neural networks. However, extending the existing network architectures to point cloud downsampling is non-trivial, since point selection process is discrete and non-differentiable. Recently, Dovrat et. al. pioneered a deep learning approach, called S-Net [12], that takes downsampling as a generative task and uses the extracted global features to generate samples. S-Net is flexible in that it can be combined with task-specific networks to produce an end-to-end network trained by a joint loss. Thanks to its task-oriented nature, S-Net outperforms FPS in various applications. However, since S-

- *This work was supported in part by the Hong Kong Research Grants Council under grants 9048123 and 9042820, and in part by the Natural Science Foundation of China under Grants 61871342.*
- *Y. Qian, J. Hou, Y. Zeng, Q. Zhang and S. Kwong are with the Department of Computer Science, City University of Hong Kong, Hong Kong. Email: yueqian4-c@my.cityu.edu.hk; jh.hou@cityu.edu.hk; ym.zeng@my.cityu.edu.hk; qijizhang3@cityu.edu.hk; cssamk@cityu.edu.hk*
- *Y. He is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore, 639798. Email: yhe@ntu.edu.hk*

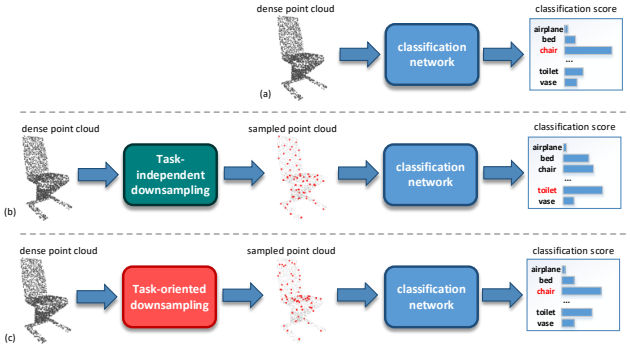

Fig. 1: Illustrations of task-independent and task-oriented point cloud downsampling using classification as an example. (a) A deep learning based point cloud classifier is trained on dense point clouds. (b) Classic downsampling methods, such as FPS, generate sparse point clouds without considering the nature of the task, hereby may compromise the performance of the classifier in (a) significantly. (c) The classification-oriented downsampling method produces sparse point clouds that maintain the performance of the classifier in (a). To develop such a classification-oriented downsampling, we take both the geometry of the input shape and the performance of the classifier into account.

Net solely relies on global features in its generative process, it does not utilize the point-wise high-dimensional local features, which limits the quality of synthesized points.

In this paper, we propose a novel deep learning approach, called MOPS-Net, for task-oriented point cloud downsampling. In contrast to the existing methods, we design MOPS-Net from a matrix optimization perspective. Viewing downsampling as a selection process, we first formulate a discrete and combinatorial optimization problem. As it is difficult to solve the 0-1 integer program directly, we relax the integer constraint for each variable and lift 3D points to a higher dimensional feature space, in which a constrained and differentiable matrix optimization problem with an implicit objective function is formulated. We then propose a deep neural network architecture to mimic the matrix optimization problem by exploring both the local and the global structures of the input data. Learning a relaxed sampling matrix in high-dimensional space allows us to better exploit the relation among points. With a task network, MOPS-Net can be end-to-end trained. MOPS-Net is permutation-invariant, making it robust to input data. Moreover, by restricting the invoking columns of the soft sampling matrix, we also extend MOPS-Net such that a single network with one-time training is capable of handling arbitrary downsampling ratios. Computational results show that MOPS achieves better performance than S-Net and the traditional methods in various tasks, including point cloud classification, retrieval and reconstruction. We also provide comprehensive analysis on each module to demonstrate the effectiveness of our design.

The rest of this paper is organized as follows. Section 2 reviews classic point cloud downsampling methods and recent deep learning techniques for 3D point clouds. In Section 3, we formulate and relax task-oriented downsampling as a constrained optimization problem, followed by an end-to-end task-oriented downsampling deep neural network which mimics the resulting optimization in Section 4. Extensive experiments and comparisons are presented in Section 5, as well as comprehensive ablation studies towards the proposed methods. Section 6 finally concludes this paper.

2 Related Work

Classic approaches, such as farthest point sampling (FPS) and Poisson disk sampling (PDS), generate samples in an iterative manner. Starting from a random sample, FPS repeatedly places the next sample point in the middle of the least-known area of the sampling domain. Using efficient geodesic distance computation tools (such as the fast marching method [54]), FPS generates m samples on a n-vertex mesh in $\sum_{i=1}^{m} f(\frac{n}{i}) = O(n \log n)$ time, where $f(x) = O(x \log x)$. FPS is easy to implement and becomes popular in designing neural networks that aggregate local features [2], [5], [6]. Poisson disk sampling produces samples that are tightly-packed, but no closer to each other than a specified minimum distance, resulting in a more uniform distribution than FPS. There are efficient implementations of PDS with linear time complexity in Euclidean spaces [3], [4]. However, it is expensive to generate Poisson disk samples on curved surfaces due to frequent computation of geodesic distances [1]. Moreover, the classic approaches focus only on preserving the geometry of the input shape and they do not consider the downstream tasks at all.

Deep learning for 3D point clouds. Due to the irregular and unordered nature of point clouds, the widely used convolutional neural networks (CNNs) on 2D images/videos [9], [10], [11] cannot be applied directly. PointNet, proposed by Qi *et al.* [42], maps a 3D point to a high dimensional space by point-wise multi-layer perceptrons (MLPs) and aggregates global features by a symmetric function, named max-pooling. As the first deep neural network that works for 3D raw points without projecting or parameterizing them to regular domains, PointNet quickly gained popularity and was successfully used as the foundamental feature extraction for point clouds. However, PointNet processes the points individually and does not consider the spatial relation among points. The follow-up works, such as PointNet++ [2], DGCNN [8] and PointCNN [6], improve PointNet by taking local geometry into account.

Inspired by the success of PointNet on classification, many other point cloud applications were studied in recent years, such as retrieval [17], [51], segmentation [13], [14], [15], [52], reconstruction [19], [38], [50], registration [39], [40], [43], [49], and object detection [44], [45], [46], [47], [48], just name a few. Although they have different problem settings, these networks can be combined with the point cloud downsampling network and jointly trained. In this paper, we evaluate the performance of the proposed downsampling framework on classification [42], retrieval [42] and reconstruction [19].

Opposite to downsampling, point cloud upsampling [21], [23], [24], [33] has also been investigated recently. Upsampling can be treated as either a 3D version of image super-resolution, or the inverse process of downsampling.

Despite the common word "sampling", the two tasks are completely different. Upsampling, as a generative task, requires informative feature expansion and can be trained by ground truth dense point clouds. In contrast, point cloud downsampling is close to feature selection, where a differentiable end-to-end framework should be carefully designed. Moreover, due to lack of ground truth, the downsampled points should be learned to optimize a specific task loss.

Deep learning-based point cloud downsampling is an emerging topic, on which there are only a few works. Nezhadary *et al.* [22] proposed to use critical points invoked in max-pooling as sampled points. In order to improve classification accuracy, Yang [20] adopted a gumbel softmax layer to integrate high level features. Recently, Dovrat *et al.* [12] proposed a data-driven point cloud downsampling framework named S-Net. After the point-wise feature extraction by PointNet, a global feature was obtained by the max-pooling operation. Then 3D coordinates of fewer points were regressed by fully-connected layers. Followed by a pre-trained task network, S-Net can be trained end-to-end. S-Net, however, regresses coordinates from global features directly and does not consider spatial correlation among points, which plays an important role in downsampling, since spatially close points have the tendency to be represented by the same downsampled point. In sharp contrast to S-Net, we propose a novel framework by exploring the local geometry of the input data in the perspective of matrix optimization.

3 Point cloud downsampling: A matrix optimization perspective

Denote by $\mathcal{P} = \{\mathbf{p}_i \in \mathbb{R}^3\}_{i=1}^n$ a dense point cloud with n points. $\mathbf{p}_i = \{x_i, y_i, z_i\}$ is the 3D Cartesian coordinates. Let $\mathcal{Q} = \{\mathbf{q}_i \in \mathbb{R}^3\}_{i=1}^m$ be the downsampled sparse point cloud with m ($< n$) points, i.e., $\mathcal{Q} \subset \mathcal{P}$.

As aforementioned, we consider task-oriented downsampling. That is, given $\mathcal{P}$, we compute the downsampled point cloud $\mathcal{Q}$ that maintain comparable or does not compromise the performance of the subsequent tasks severely, e.g., classification, retrieval, etc. Such a downsampling process is beneficial to computation, storage and transmission.

Mathematically speaking, the problem can be formulated as

$$\min_{\mathcal{Q}} L_{task}(\mathcal{Q}) \;\; s.t. \;\; \mathcal{Q} \subset \mathcal{P}, \tag{1}$$

where $L_{task}(\cdot)$ indicates the task-tailored loss, whose explicit form will be discussed in Section 4.4. We rewrite Eq. (1) in a more explicit manner

$$\begin{aligned} &\min_{\mathbf{S}} L_{task}(\mathbf{Q}) \\ &s.t.\ \mathbf{Q} = \mathbf{PS},\ \mathbf{1}_n^\mathsf{T}\mathbf{S} = \mathbf{1}_m^\mathsf{T},\ \mathbf{S}^\mathsf{T}\mathbf{S} = \mathbf{I}_m,\ s_{i,j} \in \{0, 1\}, \end{aligned} \tag{2}$$

where $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n] \in \mathbb{R}^{3\times n}$ and $\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_m] \in \mathbb{R}^{3\times m}$ are the matrix representations of $\mathcal{P}$ and $\mathcal{Q}$, respectively, constructed by stacking each point as a column in an *unodered* manner[1]; $s_{i,j}$ is the (i, j)-th entry of $\mathbf{S} \in \mathbb{R}^{n\times m}$; $\mathbf{1}_n = [1, \ldots, 1]^\mathsf{T} \in \mathbb{R}^{n\times 1}$ is the column vector with all elements equal to 1; and $\mathbf{I}_m$ refers to the identity matrix of size $m \times m$. The constraints limit $\mathbf{S}$ to be an ideal sampling matrix, i.e., $\mathbf{S}$ only contains m columns of a permutation matrix of size $n \times n$.

The challenge for solving Eq. (2) comes from the discrete and binary characteristics of matrix $\mathbf{S}$. To tackle this challenge, we relax the binary constraints Eq. (2) in a soft and continuous manner, i.e., the elements of $\mathbf{S}$ are continuous, ranging between 0 and 1. The relaxed variables indicate the probabilities of the corresponding points that will be sampled. In this relaxed framework, the sampled points in $\mathbf{Q}$ may not be the exact ones in $\mathcal{P}$. To preserve the geometry, we expect the points of $\mathbf{Q}$ to be close to those of $\mathbf{P}$ as much as possible. The shape difference between $\mathbf{P}$ and $\mathbf{Q}$ can be quantitatively measured by the distance between them $L_{dist}(\mathbf{P}, \mathbf{Q})$, whose explicit form will be discussed in Section 4.4. Minimizing such a distance forces the points of $\mathbf{Q}$ moving towards $\mathbf{P}$; hereby the resulting matrix $\mathbf{S}$ is promoted to approach the solution of Eq. (2). The *relaxed* and *continuous* optimization problem is expressed as

$$\begin{aligned} &\min_{\mathbf{S}} L_{task}(\mathbf{Q}) + \alpha L_{dist}(\mathbf{P}, \mathbf{Q}) \\ &s.t.\ \mathbf{Q} = \mathbf{PS},\ \mathbf{S} \geq 0,\ \mathbf{1}_n^\mathsf{T}\mathbf{S} = \mathbf{1}_m^\mathsf{T},\ \|\mathbf{S}^\mathsf{T}\mathbf{S} - \mathbf{I}_m\|_F < \epsilon, \end{aligned} \tag{3}$$

where $\|\cdot\|_F$ is the Frobenious norm of a matrix, $\epsilon > 0$ is a threshold, and $\alpha > 0$ is the penalty parameter to balance the two terms.

Remark. Relaxing the binary matrix $\mathbf{S}$ produces a point cloud which is not a subset of the input $\mathcal{Q} \not\subset \mathcal{P}$. This can be explained from the perspective of geometry processing. When presenting an object using two point clouds of different resolutions, the one with fewer points is generally not fully overlapping with the larger one since we have to re-distribute the points in order to preserve the geometry. Although $\mathcal{Q} \not\subset \mathcal{P}$, the objective function penalizes the points that are away from the input shape. Therefore, the points of $\mathcal{Q}$ are either on or close to the underlying object surface. Computational results in Section5 confirm this observation. If the subsequent task requires $\mathcal{Q} \subset \mathcal{P}$, we can apply a post-matching operation, i.e., assigning each point of $\mathcal{Q}$ the closest point in the input data.

Kernel methods are a powerful computational tool for pattern analysis, since they operate in a high-dimensional, implicit feature space without ever computing the coordinates of the data in that space [41]. Inspired by kernel methods, we lift the points to a high dimensional feature space by a typical mapping function $\phi(\cdot) : \mathbb{R}^3 \to \mathbb{R}^d$ and conduct sampling in $\mathbb{R}^d$. We denote by $\mathbf{\Phi}(\mathbf{P}) = [\phi(\mathbf{p}_1), \ldots, \phi(\mathbf{p}_n)]$ and $\mathbf{\Phi}(\mathbf{Q}) = [\phi(\mathbf{q}_1), \ldots, \phi(\mathbf{q}_m)]$ the transformed point clouds. The inverse function $\phi^{-1}(\cdot)$, which maps the transferred points back to 3D space, is required to produce the downsampled point cloud. Putting it all together, we formulate the relaxed downsampling problem as

$$\begin{aligned} &\min_{\mathbf{S}} L_{task}(\mathbf{\Phi}^{-1}(\mathbf{\Phi}(\mathbf{Q}))) + \alpha L_{dist}(\mathbf{P}, \mathbf{\Phi}^{-1}(\mathbf{\Phi}(\mathbf{Q}))), \\ &s.t.\ \mathbf{\Phi}(\mathbf{Q}) = \mathbf{\Phi}(\mathbf{P})\mathbf{S},\ \mathbf{S} \geq 0,\ \mathbf{1}_n^\mathsf{T}\mathbf{S} = \mathbf{1}_m^\mathsf{T},\ \|\mathbf{S}^\mathsf{T}\mathbf{S} - \mathbf{I}_m\|_F < \epsilon, \end{aligned} \tag{4}$$

where $\mathbf{\Phi}^{-1}(\mathbf{\Phi}(\mathbf{Q})) = [\phi^{-1}(\phi(\mathbf{q}_1)), \cdots, \phi^{-1}(\phi(\mathbf{q}_m))]$. See the experimental comparisons between sampling in 3D

1. Note $\mathcal{P}$ (resp. $\mathcal{Q}$) and $\mathbf{P}$ (resp. $\mathbf{Q}$) stand for the same data but in different forms, and there is no specific requirement on the order when stacking the points. The notations are used interchangeably in the paper.

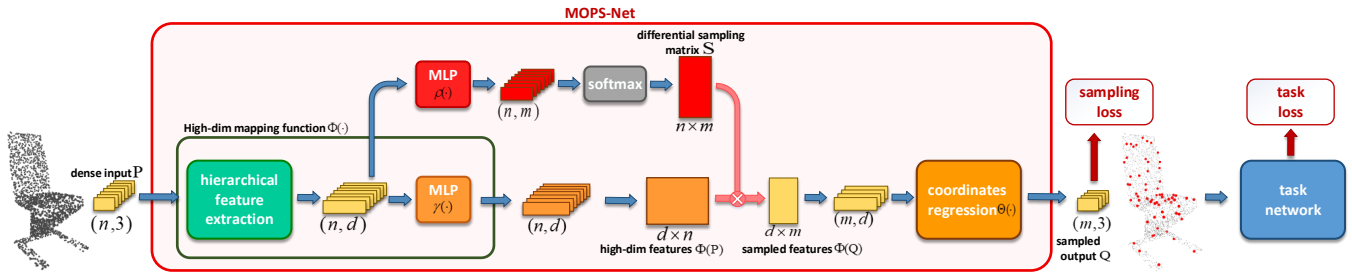

Fig. 2: The flowchart of MOPS-Net, a matrix optimization inspired deep learning method for task-oriented 3D point cloud downsampling. After mapping an input point cloud (represented by a set $\mathcal{P}$ or a matrix $\mathbf{P}$) into a high dimensional space via a hierarchical feature extraction module as well as additional MLPs ($\gamma(\cdot)$), we regress a differentiable sampling matrix $\mathbf{S} \in \mathbb{R}^{n\times m}$ ($m < n$) via an MLP $\rho(\cdot)$. Multiplying $\mathbf{S}$ to the high-dimensional representations of the input data $\mathbf{\Phi}(\mathbf{P})$ produces downsampled high-dimensional features $\mathbf{\Phi}(\mathbf{Q})$, which are then projected to 3D space via the coordinate regression module $\mathbf{\Theta}(\cdot)$, generating sampled point cloud ($\mathcal{Q}$ or $\mathbf{Q}$). MOPS-Net is permutation-invariant, which can be end-to-end trained together with a fixed task network. A joint loss, which considers both the performance of a specific task and the geometry of the sampled points, is used. A optional post-matching operation can be applied to ensure the downsampled set is a subset of the input set $\mathcal{Q} \subset \mathcal{P}$.

space and in high-dimensional feature space in Section 5.4 and Table 7.

4 Proposed Method

This section presents MOPS-Net, a novel end-to-end deep neural network that mimics the formulated optimization problem in Eq. (4) for task-oriented point cloud downsampling. Such a network unifies the high dimensional representation process $\phi(\cdot)$, the inverse process $\phi^{-1}(\cdot)$, and the downsampling process to optimize them jointly.

4.1 Overview

Figure 2 illustrates the flowchart of MOPS-Net. Given a point cloud $\mathcal{P}$ and a pre-trained task network, MOPS-Net uses a hierarchical feature extraction module, which corresponds to the mapping function $\phi(\cdot)$, to encode each point with high dimensional and informative features by exploring both the local and the global structures of $\mathcal{P}$ (Section 4.2). Based on the high dimensional features, MOPS-Net estimates a differentiable sampling matrix $\mathbf{S}$ via an MLP as well as the softmax operation which realizes the constraints on $\mathbf{S}$ in Eq. (4). Multiplying the high dimensional features and $\mathbf{S}$ produces the features of the downsampled points (Section 4.3), which are further fed into a coordinate regression module $\theta(\cdot)$ that mimics the inverse mapping function $\phi^{-1}(\cdot)$ to regress the coordinates of the sampled points (Section 4.4). Together with a fixed task network, MOPS-Net can be end-to-end trained with a joint loss (Section 4.5). Such a joint loss simultaneously penalizes the degradation of task performance and regularizes the distribution of sampled points, which is consistent with the objective function in Eq. (4). We show that MOPS-Net is flexible and it can be extended so that a *single* network with one-time training is capable of handling *arbitrary* downsampling ratio (Section 4.6). Last but not the least, we prove that MOPS-Net is *permutation-invariant*, which is a highly desired feature for point cloud applications. An optional post-matching operation may be applied to ensure the sampled points are the subset of the input point cloud.

Remark. The proposed MOPS-Net is fundamentally different from the existing deep learning framework S-Net [12]. S-Net formulates the sampling process as a point generation problem from global features, while MOPS-Net is designed from matrix optimization perspective to utilize the informative local (or point-wise) features. Experimental results confirmed the advantages of MOPS-Net over S-Net on classification, reconstruction and retrieval. See Section 5.

4.2 Hierarchical Feature Extraction

Given a point cloud $\mathcal{P} = \{\mathbf{p}_i \in \mathbb{R}^3\}_{i=1}^n$, we extract d-dimensional point-wise features, denoted by $\mathcal{C} = \{\mathbf{c}_i \in \mathbb{R}^d\}_{i=1}^n$, in a hierarchical manner. We use $\mathbf{C} = [\mathbf{c}_1, \ldots, \mathbf{c}_n] \in \mathbb{R}^{d\times n}$ to denote the matrix form of hierarchical features. In this paper, we make use of DGCNN [8] to extract point-wise features hierarchically, in which both local and global geometry information of $\mathcal{P}$ are explored. Note that other feature extraction networks for 3D point clouds such as PointNet and PointCNN can also be employed. See the experimental analysis in Section 5.4 and Table 5.

Specifically, at the l-th layer, for point $\mathbf{p}_i$ equipped a with d_l-dimensional feature $\mathbf{c}_i^l \in \mathbb{R}^{d_l}$, we first determine its k nearest neighbours $\mathcal{N}_i^l = \{\mathbf{p}_j\}_{j=1}^k$ by measuring Euclidean distances among the features, i.e., $\|\mathbf{c}_i^l - \mathbf{c}_j^l\|_2$ with $\|\cdot\|_2$ being the ℓ_2 norm of a vector and $\mathbf{c}_j^l \in \mathbb{R}^{d_l}$ being the corresponding feature of the j-th neighbour. A non-linear MLP $h_l(\cdot)$ with learnable parameters then encodes the local geometry information, followed by max-pooling, a symmetric aggregation operator $\square$, to generate d_{l+1}-dimensional feature for point $\mathbf{p}_i$ as the input of the successive layer. The process is formulated as

$$\mathbf{c}_i^{l+1} = \square_{\mathbf{c}_j^l \in \mathcal{N}_i^l} h_l(\mathbf{c}_i^l, \mathbf{c}_j^l - \mathbf{c}_i^l). \tag{5}$$

The whole hierarchical poin-twise feature extraction module $H(\cdot)$ is derived by performing such a process L times successively, i.e.

$$\mathcal{C}^L = H(\mathcal{P}) = h_L \circ h_{L-1} \circ \ldots \circ h_1(\mathcal{P}), \tag{6}$$

where $\mathcal{C}^L = \{\mathbf{c}_i^L \in \mathbb{R}^{d_L}\}_{i=1}^n$ is the set of point-wise features of the L-th layer, whose matrix form is written as $\mathbf{C}^L \in \mathbb{R}^{d_L \times n}$. Also, a global feature $\mathbf{g} \in \mathbb{R}^{d_g}$ can be obtained by applying max-pooling on $\mathcal{C}^L$, i.e.,

$$\mathbf{g} = \square_{i=1}^n \mathbf{c}_i^L. \tag{7}$$

Finally, the global knowledge of $\mathcal{C}^L$ is further enhanced by concatenating the global feature to each point-wise feature, leading to the final output of this module, i.e., $\mathbf{c}_i = \mathsf{concat}(\mathbf{c}_i^L, \mathbf{g}) \in \mathbb{R}^d$ where $\mathsf{concat}(\cdot)$ refers to the concatenation operation.

Since directly using $\mathcal{C}$ to construct both the high-dimensional representation of $\mathbf{P}$ and the sampling matrix $\mathbf{S}$ (Section 4.2) may introduce a coupling effect, we employ one more MLP, denoted by $\gamma(\cdot)$, to process $\mathcal{C}$. The resulting features are adopted as the high-dimensional representations of $\mathbf{P}$, i.e.,

$$\phi(\mathbf{p}_i) = \gamma(\mathbf{c}_i). \tag{8}$$

4.3 Learning Differentiable Sampling Matrix

As analyzed in Section 3, an ideal sampling matrix, which is a submatrix of a permutation matrix, is discrete and non-differentiable, making it challenging to optimize. Accordingly, such a non-differentiable sampling matrix cannot be implemented in a deep neural network. Fortunately, the relaxation on the sampling matrix in Eq. 4 leads to a continuous matrix with additional constraints, which approximates the ideal one and allows us to design a deep neural network.

According to Eq. 4, it is known that the optimized sampling matrix $\mathbf{S}$ will depend on $\mathcal{P}$ in a non-linear fashion. From the perspective of geometry processing, downsampling highly depends on the geometric structure of the input data. As the high-dimensional embeddings $\mathbf{c}_i$ produced by the hierarchical feature extraction module already encode such a structure locally and globally, we use them to predict a preliminary sampling matrix $\overline{\mathbf{S}} \in \mathbb{R}^{n \times m}$ row by row via an MLP $\rho(\cdot)$:

$$\overline{\mathbf{s}}_i = \rho(\mathbf{c}_i). \tag{9}$$

We further apply a column-wise softmax operation on $\overline{\mathbf{S}}$ to satisfy the constraints on the sampling matrix in Eq. (4)

$$\mathbf{S} = softmax(\overline{\mathbf{S}}). \tag{10}$$

The softmax operation $\mathsf{softmax}(\cdot)$ encourages $\mathbf{S}$ being non-negative and each column of $\mathbf{S}$ being dominated by a single element. We experimentally found that such an operation is able to realize the constraints in Eq. (4) well. See the visualization of $\mathbf{S}^\mathsf{T}\mathbf{S}$ in Figure 8.

In addition to differentiability, our design of the sampling matrix learning is also permutation-invariant. That is, the sampling result is independent of the point order of the input data. See the proof in Section 4.4.

4.4 Regression of the Sampled Set

When multiplying the high-dimensional representations of $\mathcal{P}$ with the sampling matrix $\mathbf{S}$ derived from the previous modules, we can obtain high-dimensional representations with a reduced size, i.e., $\mathbf{\Phi}(\mathbf{Q})$, which correspond to those of selected points. Naturally, the inverse of the mapping function $\phi(\cdot)$ could be computed to project $\mathbf{\Phi}(\mathbf{Q})$ to 3D space for generating the 3D coordinates of these points. However, since $\phi(\cdot)$ involves multiple non-invertible units such as ReLU, computing $\phi^{-1}(\cdot)$ is not feasible. Hence, we adopt an MLP, denoted by $\{\theta_i(\cdot), i = 1, \ldots, t\}$, to approximate $\phi^{-1}(\cdot)$ and to regress the 3D coordinates from $\mathbf{\Phi}(\mathbf{Q})$ point by point:

$$\mathbf{q}_i = \theta(\mathbf{\Phi}(\mathbf{P})\mathbf{s}_i) = \theta_t \circ \cdots \circ \theta_1(\mathbf{\Phi}(\mathbf{P})\mathbf{s}_i). \tag{11}$$

See the experimental analysis in Section 5.4 and Figure 10. As relaxation cannot guarantee $\mathcal{Q} \subset \mathcal{P}$, we can adopt an optional post-matching operation that maps each point of $\mathcal{Q}$ to its nearest point in $\mathcal{P}$. If there are fewer matched points than required, we adopt FPS to complete the sampled set.

Proposition 1. MOPS-Net is permutation-invariant.

Let $\mathbf{E} \in \mathbb{R}^{n \times n}$ be an arbitrary permutation matrix ($\mathbf{E}^\mathsf{T}\mathbf{E} = \mathbf{E}\mathbf{E}^\mathsf{T} = \mathbf{I}_n$), and $\widetilde{\mathbf{P}} = \mathbf{P}\mathbf{E}$ be the permutated input. Denote by $\widetilde{\mathbf{C}} \in \mathbb{R}^{d \times n}$, $\widetilde{\mathbf{C}}^L \in \mathbb{R}^{d_L \times n}$, $\widetilde{\mathbf{g}} \in \mathbb{R}^{d_g}$ and $\widetilde{\mathbf{Q}} \in \mathbb{R}^{3 \times m}$ the corresponding high-dimensional point-wise features, point-wise local features, global feature, and sampled points when feeding $\widetilde{\mathbf{P}}$ into MOPS-Net, respectively.

Notice that the non-linear function $H(\cdot)$ extracts point-wise features with shared MLPs; hence $\widetilde{\mathbf{C}}^L = H(\widetilde{\mathbf{P}}) = H(\mathbf{P}\mathbf{E}) = H(\mathbf{P})\mathbf{E} = \mathbf{C}^L\mathbf{E}$. Since the global feature is aggregated via max-pooling, which is a symmetric function[2], we have $\widetilde{\mathbf{g}} = \mathbf{g}$. Therefore, the high-dimensional features $\widetilde{\mathbf{C}}$, which concatenate $\widetilde{\mathbf{C}}^L$ and $\widetilde{\mathbf{g}}$, satisfy $\widetilde{\mathbf{C}} = \mathbf{C}\mathbf{E}$.

Similarly, we obtain $\mathbf{\Phi}(\widetilde{\mathbf{P}}) = \mathbf{\Phi}(\mathbf{P}\mathbf{E}) = \mathbf{\Phi}(\mathbf{P})\mathbf{E}$ and $\widetilde{\mathbf{S}}^\mathsf{T} = \mathbf{S}^\mathsf{T}\mathbf{E}$ (again due to the permutation-invariant property of softmax), implying

$$\mathbf{\Phi}(\widetilde{\mathbf{Q}}) = \mathbf{\Phi}(\widetilde{\mathbf{P}})\widetilde{\mathbf{S}} = \mathbf{\Phi}(\mathbf{P})\mathbf{E}\mathbf{E}^\mathbf{T}\mathbf{S} = \mathbf{\Phi}(\mathbf{P})\mathbf{S} = \mathbf{\Phi}(\mathbf{Q}). \tag{12}$$

After applying another point-wise MLP $\theta(\cdot)$ to regress the 3D coordinates, we have

$$\widetilde{\mathbf{Q}} = \mathbf{\Theta}(\mathbf{\Phi}(\widetilde{\mathbf{Q}})) = \mathbf{\Theta}(\mathbf{\Phi}(\mathbf{Q})) = \mathbf{Q}, \tag{13}$$

indicating that the sampled point set from $\widetilde{\mathbf{P}}$ is identical to those from $\mathbf{P}$, which completes the proof.

4.5 Joint Training Loss

As analyzed in the objective function (4), two types of losses are needed to train MOPS-Net, i.e., the task loss $L_{task}(\cdot)$ and the distance loss $L_{dist}(\cdot, \cdot)$. Specifically, $L_{task}(\cdot)$ aims to promote the network to learn downsampled point clouds that are able to maintain the high performance for a specific task. Let $\mathscr{F}_T(\cdot)$ be the network for a typical task, which was trained with the original point cloud data, and we have

$$L_{task}(\mathcal{Q}) = L_T(\mathscr{F}_T(\mathcal{Q}), y^*), \tag{14}$$

where y^* is the corresponding ground-truth data for $\mathcal{Q}$. Specifically, y^* will be the class label and the input point cloud when the task is classification and reconstruction, respectively.

The distance loss $L_{dist}(\cdot, \cdot)$ regularizes the network to learn downsampled point clouds that are close to the inputs. It consists of a subset regularization $L_{subset}(\cdot, \cdot)$ and a coverage regularization $L_{coverage}(\cdot, \cdot)$,

$$L_{dist}(\mathcal{P}, \mathcal{Q}) = L_{subset}(\mathcal{P}, \mathcal{Q}) + \beta L_{coverage}(\mathcal{P}, \mathcal{Q}), \tag{15}$$

2. See [42] for more details about the permutation-invariant properties of max-pooling.

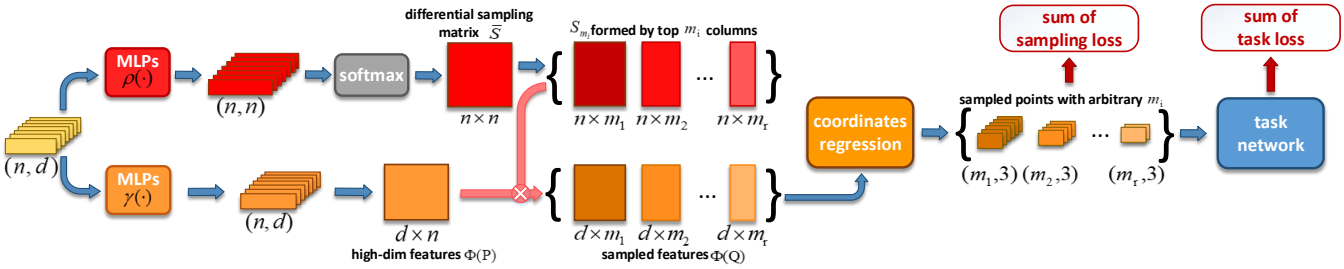

Fig. 3: FMOPS-Net extends MOPS-Net so that a single network with one-time training can handle arbitrary sampling size. FMOPS-Net learns a square matrix $\bar{\mathbf{S}} \in \mathbb{R}^{n\times n}$, whose m_i left-most columns are collected to form the differential matrix $\mathbf{S}_{m_i}$ to generate a sparse point cloud with m_i points. A multi-level loss is used to achieve the flexibility.

where β is the penalty parameter to balance the two parts. Specifically, $L_{subset}(\cdot,\cdot)$ is used to constrain the downsampled point cloud to approach the subset of the input, and both the average and maximum distances are considered, balanced by the parameter τ:

$$L_{subset}(\mathcal{P},\mathcal{Q}) = \frac{\tau}{M}\sum_{i=1,\dots,m}\min_{\mathbf{p}\in\mathcal{P}}||\mathbf{q}_i-\mathbf{p}||_2^2+\max_{i=1,\dots,m}\min_{\mathbf{p}\in\mathcal{P}}||\mathbf{q}_i-\mathbf{p}||_2^2, \tag{16}$$

where $\|\cdot\|_2$ is the ℓ_2 norm of a vector. Generally, a point cloud that represents the whole shape of an object could be better recognized/distinguished than that corresponding to part of the object. We thus use $L_{coverage}(\cdot,\cdot)$ to encourage the downsampled point clouds preserve the overall shapes of the original ones:

$$L_{coverage}(\mathcal{P},\mathcal{Q}) = \frac{1}{n}\sum_{i=1,\dots,n}\min_{\mathbf{q}\in\mathcal{Q}}||\mathbf{p}_i-\mathbf{q}||_2^2. \tag{17}$$

Therefore, the total loss $L_{total}(\cdot,\cdot)$ for end-to-end training of MOPS-Net is written as

$$L_{total}(\mathcal{P},\mathcal{Q}) = L_{task}(\mathcal{Q}) + \alpha L_{dist}(\mathcal{P},\mathcal{Q}). \tag{18}$$

See the experimental analysis about the loss in Section 5.4, Table 6 and Figure 7.

4.6 Flexible MOPS-Net for Arbitrary Ratios

In the previous sections, we construct MOPS-Net with a predefined sample size m, and a different network has to be trained for each m, which is tedious and unpractical for real-world applications. To solve this issue, we extend MOPS-Net and propose flexible MOPS-Net (FMOPS-Net), which is a single network that can achieve 3D point cloud downsampling with arbitrary sampling ratios after only one-time training. See Figure 3 for the flowchart of FMOPS-Net.

Specifically, instead of learning a rectangular sampling matrix in MOPS-Net, i.e., $\mathbf{S} \in \mathbb{R}^{n\times m}$ $(m < n)$, we consider a square matrix $\widehat{\mathbf{S}} \in \mathbb{R}^{n\times n}$ with other network components unchanged. Then given an arbitrary sample size m, the m left-most columns of $\widehat{\mathbf{S}}$ are selected to form the sampling matrix $\mathbf{S} \in \mathbb{R}^{n\times m}$, producing a point cloud with m points. Such a manner is equivalent to indirectly sorting the points of $\mathcal{P}$ according to their importance in a downsampled point cloud.

Denote by $\{m_1,\dots,m_r\}$ the sizes of r levels of downsampled point clouds we are interested in. The above mentioned flexibility can be achieved by training FMOPS-Net with a multi-level loss function $L_{sum}(\cdot,\cdot)$, which combines the joint loss for multiple sampled sets $\mathcal{Q}_{m_i}$ $(i = 1,\cdots,r)$:

$$L_{sum}(\mathcal{P},\{\mathcal{Q}_{m_i}\}) = \sum_{i=1,\dots,r}\lambda_i L_{total}(\mathcal{P},\mathcal{Q}_{m_i};\mathbf{S}_{m_i}), \tag{19}$$

where λ_i is the weight to balance the i-th sampled set.

5 Experiments

We validated the effectiveness of MOPS-Net and FMOPS-Net on three typical tasks, which are point cloud classification, retrieval and reconstruction. We used two widely used task-independent methods, random sampling (RS) and farthest point sampling (FPS), as baselines. We also compared with S-Net [12], the state-of-the-art deep learning-based task-oriented downsampling method. In addition, we conducted extensive ablation studies to comprehensively analyze and evaluate our networks.

5.1 Classification-oriented Downsampling

Implementation details and experiment settings. In MOPS-Net and FMOPS-Net, $H(\cdot)$ contains 3 EdgeConv layers used in DGCNN, and each layer consists of 3 MLPs of sizes $[64,64]$, $[64,64]$, and $[64,1024,128]$, respectively. We implemented $\rho(\cdot)$ by a single-layer MLP with output size m, $\gamma(\cdot)$ by a single-layer MLP with output size 128, and $\theta(\cdot)$ by a three-layer MLP of size $[256,256,3]$. We empirically set the parameters $\alpha = 10$, $\beta = 1$, $\tau = r+5$, and $\lambda_i = 1$, where r is the sampling ratio.

We used ModelNet40 [42], a benchmark dataset for classification, which covers 3D shapes in 40 categories and each point cloud has 1024 points. We adopted PointNet as the classification network for a fair comparison with S-Net. We also followed the training protocol of S-Net, which is illustrated in Figure 1, to train MOPS-Net/FMOPS-Net as well as S-Net on the same dataset. Specifically, we first trained the classification network on ModelNet40 with the same settings as in PointNet [42]. We then froze the trained classification network and trained MOPS-Net/FMOPS-Net on the same dataset[3]. The classification accuracy is 0.892

3. We also applied the same training protocol to retrieval and reconstruction. The reproduced results of S-Net on all 3 tasks are almost the same as those reported in [12].

TABLE 1: Comparisons of the classification accuracy by different downsampling methods. The larger, the better. Note the classification accuracy of original point clouds with 1024 points each is 0.892 when using the same classification network.

m	RS	FPS	S-Net	S-Net-M	MOPS-Net	MOPS-Net-M	FMOPS-Net	FMOPS-Net-M
512	0.878	0.879	0.857	0.882	0.883	0.883	0.876	0.883
256	0.797	0.846	0.860	0.847	0.874	0.867	0.872	0.862
128	0.607	0.760	0.852	0.793	0.872	0.850	0.859	0.833
64	0.348	0.564	0.855	0.708	0.871	0.810	0.844	0.761
32	0.154	0.289	0.854	0.610	0.861	0.776	0.741	0.524
16	0.061	0.149	0.794	0.349	0.847	0.512	0.576	0.236

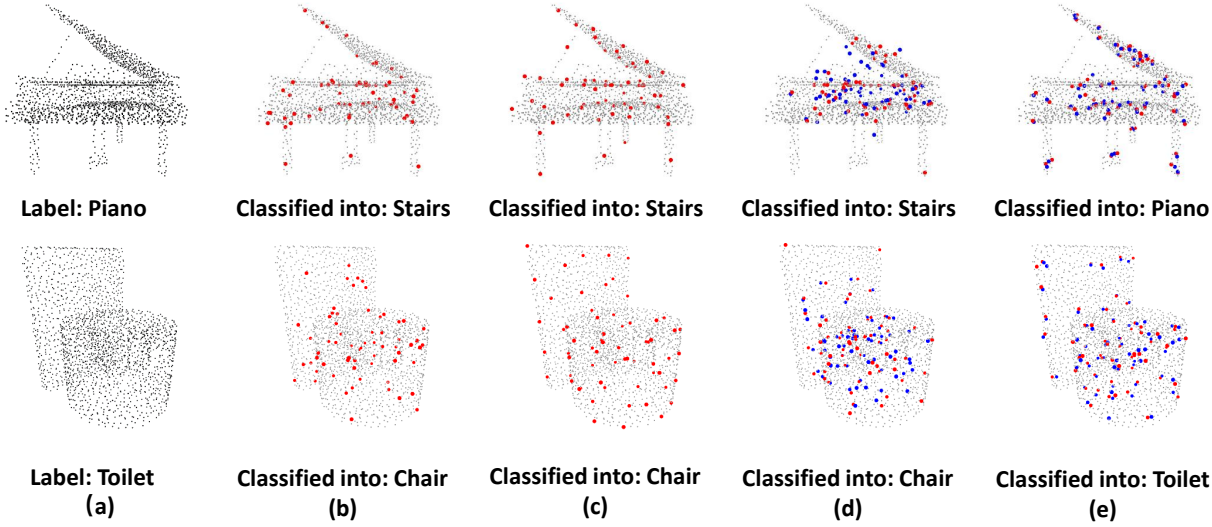

Fig. 4: Visual comparisons of sampled point clouds by different downsampling methods with $m = 64$ as well as the classification results. (a) The original dense point clouds (1024 points); (b) Random sampling results; (c) FPS results; (d) Results of S-Net (blue) and S-Net-M (red); and (e) Results of MOPS-Net (blue) and MOPS-Net-M (red). The classification results of (d) and (e) are those of S-Net-M and MOPS-Net-M.

when applying the trained classification network on the original point clouds of ModelNet40. We implemented our methods in Tensorflow, chose Adam [18] with batch normalization as the optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.99$, and set the batch size as 32. We initialized the learning rate as $1e-2$ and it gradually decreased to $1e-5$ during the training progress. We will make the code of MOPS-Net and FMOPS-Net publicly available.

Quantitative and qualitative comparisons. Table 1 reports the classification accuracy of different methods under various sample sizes m, ranging from 16 to 512. We use suffix "-M" to denote the deep neural networks with a post-matching operation.

We observe that with the sample size m decreasing, all methods suffer from performance degradation. The smaller the sample size, the worse the performance. Task-independent methods are much more sensitive to the sample size than task-oriented methods. When m is reduced to 16 from 1024, the accuracies of RS and FPS drop 83% and 74%, respectively. In contrast, S-Net and MOPS-Net roughly keep their performance, with accuracies dropping only 10% and 5%, respectively. In particular, MOPS-Net can still achieve a fairly good accuracy 84.7% for a very low sample size $m = 16$. This observation confirms the effectiveness of task-oriented methods.

We also observe that MOPS-Net and MOPS-Net-M consistently outperform S-Net and S-Net-M for all sample sizes and the performance gap becomes larger for smaller sample size. The reason is that S-Net extracts global features of the input data to synthesize new points. Due to lack of local geometry, their synthesized points may be far way from the underlying surface. MOPS-Net, in contrast, takes the local structure into account, i.e., linearly combining/aggregating only a few features of local points via the sampling matrix to regress 3D points. As a result, the generated points are either on or close to the underlying surface. See the experimental verification in Section 5.4.

Another interesting (but unsurprising) observation is MOPS-Net always outperforms MOPS-Net-M under all cases. This is because a true subset of the input is not the ideal way to represent the downsampled model. With fewer points, their locations should be computed via optimization in order to preserve the shape. This property does not hold for S-Net and S-Net-M. For example, S-Net-M has higher accuracy than S-Net for $m = 512$.

In addition to the quantitative comparisons, we also visually compared the downsampled results. As Figure 4 shows, the points computed by MOPS-Net are not evenly distributed over the shape, but they can capture the key geometric structures of the objects such that they can be correctly recognized by the subsequent classifier. However, the downsampled points generated by other methods are wrongly classified.

Evaluation of FMOPS-Net. Table 1 also lists the performance of FMOPS-Net/FMOPS-Net-M, a single network with one-time training for dealing with arbitrary sample

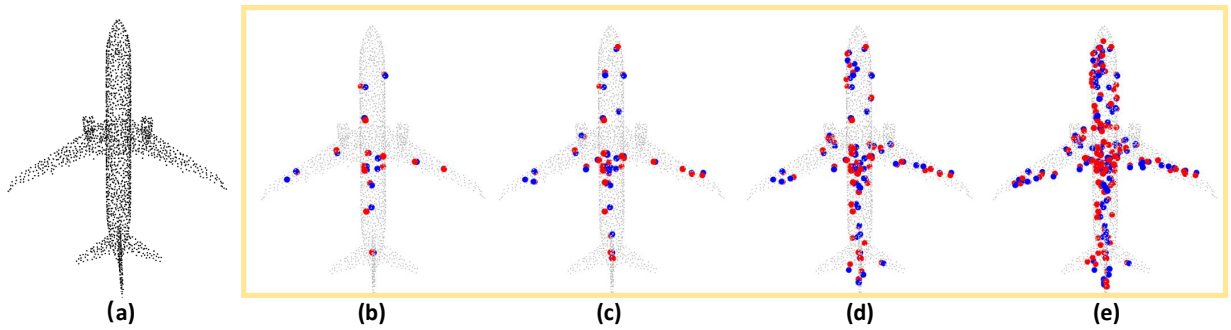

Fig. 5: Visual results of FMOPS-Net (blue points) and FMOPS-Net-M (red points) when the sample size gradually increases to 128 from 16. (a) The original point cloud with 1024 points (b) m=16 (c) m=32 (d) m=64 (e) m=128.

TABLE 2: Comparisons of classification accuracy of different downsampling methods when the classification network was trained with downsampled point clouds by each method.

m	RS	FPS	S-Net-M	MOPS-Net-M
512	0.869	0.886	0.885	0.886
256	0.873	0.883	0.881	0.883
128	0.863	0.875	0.874	0.885
64	0.827	0.863	0.865	0.879
32	0.764	0.841	0.864	0.879
16	0.668	0.812	0.844	0.873

sizes. Comparing FMOPS-Net (resp. FMOPS-Net-M) with MOPS-Net (resp. MOPS-M), we see that the flexibility only sacrifices the performance slightly except for very low sample sizes m=32 and 16, which demonstrates the effectiveness of FMOPS-Net. Besides, FMOPS-Net-M achieves comparable or even better performance than S-Net-M. For example, FMOPS-Net-M's accuracy is 76.1% for $m = 64$, improving S-Net-M by 5.3%. Figure 5 visualizes the downsampled points by FMOPS-Net/FMOPS-Net-M under various m. We also see that with increasing m, the sampled points are spreading out to cover the entire the shape and they are close to the input points.

Retrained classifier with downsampled point clouds. Previously, we trained the classification network with the original point clouds and then fixed the network. Here, for each method, we used previously obtained downsampled point clouds for training to *retrain* the classification network and then classify the downsampled ones obtained during testing. Such training and testing were conducted for each sample size separately. As illustrated in Table 2, MOPS-Net-M can achieve the highest accuracy under all cases. As the sample size decreases from 1024 to 16, the classification accuracy only drops 1.9%, which is more robust than FPS (8% decrements) and S-Net-M (4.8% decrements). The results show that MOPS-Net-M is not over-fitted to a particular classifier, and it can successfully sample points with meaningful information that benefits classification.

5.2 Retrieval-oriented Downsampling

Implementation details and experiment settings. Retrieval aims to search relevant 3D point clouds in a database, given a query point cloud. The output is the models in the same class and of similar shapes. This is usually done by finding the shapes with similar descriptors [42]. In this task, the trained classification network and MOPS-Net in Section 5.1 were adopted without retraining, and the benchmark dataset ModelNet40 was employed for evaluation. The features of the second-last layer of the classification network were collected as the shape descriptors. The mean average precision (mAP) was used to measure quantitative performance.

TABLE 3: Comparisons of the retrieval performance of different downsampling methods. The larger, the better. Note that the retrieval performance is 70.9 when original point clouds with 1024 points each were used under the same settings.

m	RS	FPS	S-Net	S-Net-M	MOPS-Net	MOPS-Net-M
512	68.1	69.4	65.3	69.1	66.3	69.7
256	61.2	65.9	66.5	64.7	67.4	66.6
128	51.7	60.2	66.2	60.1	67.2	64.1
64	37.9	53.1	66.8	56.5	65.7	62.0
32	24.8	42.9	67.6	55.2	69.9	63.5
16	16.2	33.1	64.3	48.1	73.5	61.8

Results. As Table 3 shows, MOPS-Net (resp. MOPS-Net-M) produces higher mAPs than RS, FPS, and S-Net (resp. S-Net-M) under almost all cases, especially for relatively small m. For example, MOPS-Net-M achieves mAP=61.8 when $m = 16$, which is 13.7 and 28.7 higher than those of S-Net-M and FPS, respectively. Moreover, MOPS-Net can achieve better performance even when the sample size decreases. For example, the mAP of the sampled points by MOPS-Net with $m = 16$ can attain 73.5, which is even higher than that of the original point clouds (i.e., 70.9). These observations demonstrate MOPS-Net is capable of extracting points whose features are more discriminative to retrieval.

5.3 Reconstruction-oriented Downsampling

The reconstruction task aims to reconstruct dense point clouds from sparsely sampled ones for representing underlying objects with more geometric details. Given a reconstruction method, the distribution of points of the sparse point cloud highly affects the reconstruction quality. Thus, we used such a task to evaluate different downsampling methods. Generally, a better downsampling method produces a smaller reconstruction error when using the downsampled point cloud to recover the input data.

TABLE 4: Comparisons of the normalized reconstruction error (NRE) of different downsampling methods. The smaller, the better.

m	RS	FPS	S-Net	S-Net-M	MOPS-Net	MOPS-Net-M	FMOPS-Net-M
1024	1.013	1.000	1.090	1.000	1.030	1.000	1.005
512	1.096	1.014	1.084	1.018	1.096	1.019	1.068
256	1.340	1.084	1.124	1.086	1.055	1.061	1.128
128	2.226	1.330	1.172	1.207	1.059	1.101	1.276
64	4.089	2.030	1.419	1.535	1.140	1.270	1.541
32	7.702	3.767	2.677	2.867	2.182	2.457	2.136

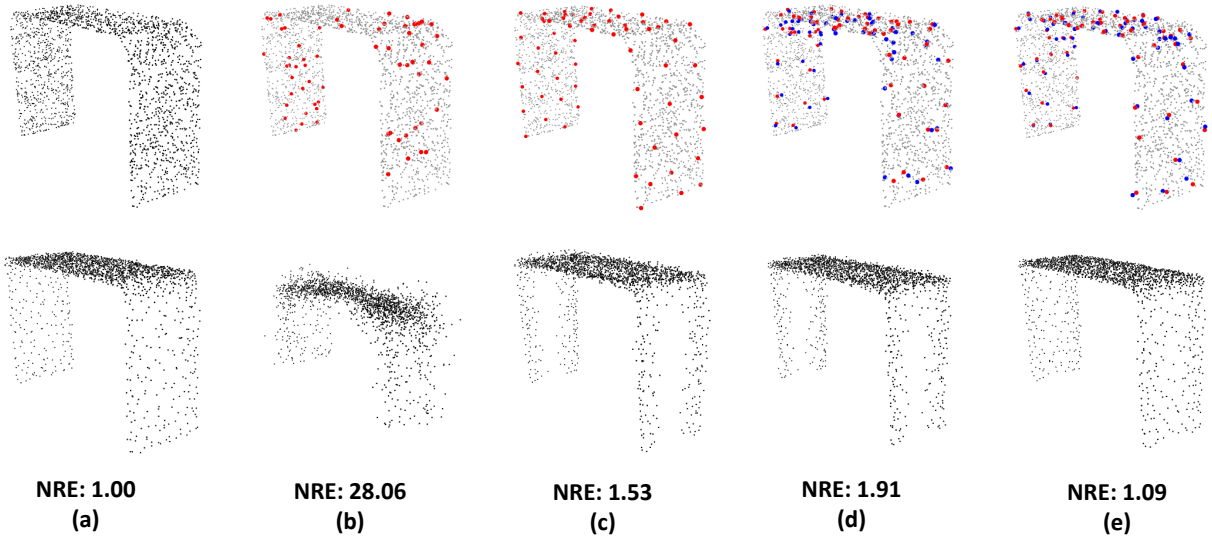

Fig. 6: Visual comparisons of the reconstructed point clouds by different downsampling methods with $m = 64$. The top row shows the sampled points (colored points) by different methods. The bottom row shows the reconstructed point clouds by different methods. (a) The original point cloud; (b) RS; (c) FPS; (d) S-Net (blue points) and S-Net-M (red points); and (e) MOPS-Net (blue points) and MOPS-Net-M (red points). Note the bottom row of (c) and (d) are the reconstructions by S-Net-M and MOPS-Net-M.

Implementation details and experiment settings. We used ShapeNetCore55 [34] as the benchmark, where each model contains 2048 points. Following S-Net, we adopted the auto-encoder in [19] as the reconstruction network for fair comparisons, where global features are first learned from the input data via an encoder and then fed into a decoder to regress coordinates of the output points. We fixed the number of output points by the reconstruction network as 2048, but the number of input points can be arbitrary. For fair comparisons, we adopted the same feature extraction architecture (i.e., PointNet) as S-Net in the hierarchical feature extraction module of our MOPS-Net/FMOPS-Net, where 1×1 convolutional layers with output of sizes $[64, 128, 128, 256, 128]$ were used. The architectures of other modules of MOPS-Net/FMOPS-Net were identical to those of the classification task in Section 5.1. The weights of the joint loss α, β and τ were empirically determined as 0.003, 1, and $\max(1, 4r - 18)$, respectively, where r is the sample ratio. For FMOPS-Net, we set $\lambda_i = 1$. We trained the MOPS-Net/FMOPS-Net with the Adam optimizer with batch normalization decay 0.9. We set the batch size as 50 and initialized the learning rate to $5e-4$, which is decreased to $1e-5$ during training.

Quantitative and qualitative comparisons. To quantitatively compare the reconstruction quality, we adopted the normalized reconstruction error (NRE), defined as

$$\mathsf{NRE}(\mathcal{Q}, \mathcal{P}) = \frac{\mathsf{CD}(\mathcal{P}, \mathscr{F}_T(\mathcal{Q}))}{\mathsf{CD}(\mathcal{P}, \mathscr{F}_T(\mathcal{P}))}, \tag{20}$$

where $\mathsf{CD}(\cdot, \cdot)$ measures the Chamfer distance between two inputs [32], and $\mathscr{F}_T(\cdot)$ is the reconstruction network. Table 4 reports the results. We observed that when the sample size m is relatively large, e.g. $m = 1024$ or 512, MOPS-Net (resp. MOPS-Net-M) has similar performance with RS, FPS and S-Net (resp. S-Net-M), and the NREs are close to 1 (i.e., lossless), indicating that 512 points are sufficient to represent the objects, and a uniform downsampling method works well. However, as the sample size decreases, the NREs of RS, FPS, and S-Net (resp. S-Net-M) increase much more rapidly than that of our MOPS-Net (resp. MOPS-Net-M), especially for relatively small m, which demonstrates the advantage of our downsampling architecture.

MOPS-Net (resp.S-Net) produces smaller NREs than MOPS-Net-M (resp. S-Net-M) when the sample size is relatively small, e.g., $m = 32, \cdots, 256$, due to the same reason as mentioned in Section 5.1. Besides, FMOPS-Net-M is comparable to S-Net-M, but slightly worse than MOPS-Net-M. However, FMOPS-Net-M can achieve the lowest NRE among all methods when $m = 32$. These results validate the effectiveness of our manner for achieving flexibility.

We also visualized the reconstruction results on a toy model with sample size $m = 64$ in Figure 6. It can be

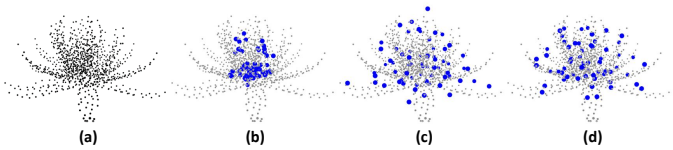

Fig. 7: Visual illustrations of the sampled points by MOPS-Net when trained with different settings of L_{dist}. $m = 64$. (a) The origin point cloud with 1024 points; (b) The sampled points by MOPS-Net when $L_{dist} = L_{subset}$; (c) The sampled points by MOPS-Net when $L_{dist} = L_{coverage}$; and (d) The sampled points by MOPS-Net when $L_{dist} = L_{subset} + \beta L_{coverage}$.

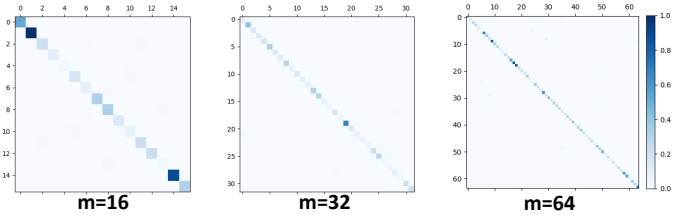

Fig. 8: Visual illustrations of the learned $\mathbf{S}^\mathsf{T}\mathbf{S}$ by MOPS-Net under various sample sizes.

seen that the reconstructed point cloud by MOPS-Net-M contains fewer outliers and is *closest* to the ground-truth. However, RS is unable to reconstruct the shape, since it does not consider geometry in sampling. FPS and S-Net also produce results with artifacts, such as outliers on the horizontal plane, and non-uniformally distributed points on the vertical planes.

5.4 Ablation Study

In this section, we conducted a series of ablation studies to comprehensively analyze and verify the contribution of each module. Specifically, we examined and analyzed the feature extraction module, the design of the loss functions for training, and the learned sampling matrix. Finally, we demonstrated the advantage of conducting downsampling in high dimensional feature space over that in 3D coordinate space.

Hierarchical feature extraction: DGCNN vs. PointNet. We investigated how the feature representation affects the performance of our methods. Here we used PointNet, a smaller network than DGCNN, to implement the hierarchical feature extraction module, which consists of an MLP with the output size $[64, 64, 64, 128, 128]$. Compared with DGCNN, PointNet lacks the ability to comprehend local information, thus leading to a performance degradation, which can be seen in Table 5. It is also worth noting that even with this simple feature extraction module, MOPS-Net still surpasses the other compared methods, including FPS and S-Net, under all cases, which demonstrates the advantage of our framework design.

Analysis of the loss function $L_{dist}(\cdot, \cdot)$. L_{dist} measures the distance between sampled points and original points during training. We analyzed the loss function using the classification task. Table 6 compares the classification accuracy of MOPS-Net and MOPS-Net-M when MOPS-Net was trained with different settings for L_{dist}. Although MOPS-Net achieves higher accuracy when trained with only the subset loss $Lsubset$ or the coverage loss $L_{coverage}$, the corresponding visualization of the sampled points under these two settings (see Figures 7(b) and (c)) show that the geometry of sampled points is seriously damaged, i.e., the sampled points either cover only part of the object or are far away from the object. This observation indicates that the sampled points tend to be 3D feature-like encodings that can be well distinguished by the classifier, rather than the 3D modeling of the object. Besides, the significant drop of the accuracy of MOPS-Net-M also reveals this issue. However, when both terms are considered, i.e., $L_{dist} = L_{subset} + \beta L_{coverage}$, the sampled points are much better distributed to preserve the shape and the classification accuracy is still comparable. Therefore, we conclude both L_{subset} and $L_{coverage}$ play critical roles in regularizing the distribution of sampled points.

TABLE 5: Classification accuracy of our methods equipped with different feature extraction modules.

sample size m	MOPS-Net (PointNet)	MOPS-Net-M (PointNet)	MOPS-Net (DGCNN)	MOPS-Net-M (DGCNN)
512	0.872	0.884	0.883	0.883
256	0.873	0.862	0.874	0.867
128	0.867	0.836	0.872	0.850
64	0.852	0.785	0.871	0.810
32	0.823	0.671	0.861	0.776
16	0.765	0.353	0.847	0.512

TABLE 6: Classification accuracy of MOPS-Net and MOPS-Net-M when MOPS-Net was trained with different settings of $L_{dist}(\cdot, \cdot)$. Here, $m = 64$.

Setting of L_{dist}	MOPS-Net	MOPS-Net-M
$L_{dist} = L_{subset}$	0.883	0.457
$L_{dist} = L_{coverage}$	0.881	0.652
$L_{dist} = L_{subset} + \beta L_{coverage}$	0.871	0.810

Analysis of the sampling matrix S. To study the learned sampling matrix $\mathbf{S}$, we check how close $\mathbf{S}^\mathsf{T}\mathbf{S}$ is to an identity matrix. The visualization in Figure 8 confirms that $\mathbf{S}^\mathsf{T}\mathbf{S}$ is close to an identity matrix under all sampling sizes. Therefore, $\mathbf{S}$ indeed mimics a binary sampling matrix, which confirms our theoretical analysis in Section 3.

Moreover, based on the learned matrix $\mathbf{S}$, we visualized the prominent points (the colored points of the right subfigures of Figure 9) that make contributions to each sampling location (the red points in the left most subfigure of Figure 9). We observe that each sampled point is only related its local neighborhoods, verifying the sampling property of the learned matrix $\mathbf{S}$. Also, such an observation demonstrates that our method aggregates local features to generate sampled points, which is totally different from S-Net which uses global features to synthesize new points.

Analysis of the coordinate regression module. In Section 4.4, we introduced the regression module $\theta(\cdot)$ to re-project downsampled high-dimensional features to 3D coordinates. We claim such a regression module realizes $\phi^{-1}(\cdot)$. Here, we experimentally justified such a statement by replacing the sampling matrix $\mathbf{S}$ with an identity matrix $\mathbf{I}_n \mathbb{R}^{n \times n}$. As shown in Figure 10, the generated point clouds is very close to the inputs, although we did not impose any restriction for achieving this target during training. Thus,

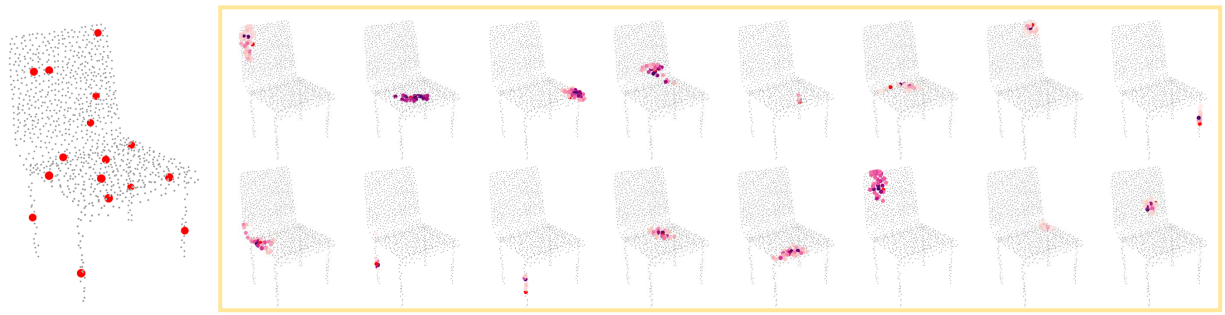

Fig. 9: Left: 16 sampled points (in red) by MOPS-Net from an input point cloud with 1024 points. Right: Each subfigure corresponds a sampled points; Based on the learned sampling matrix $\mathbf{S}$, we visualized the prominent points with colors, which contribute to the sampled points during feature aggregation.

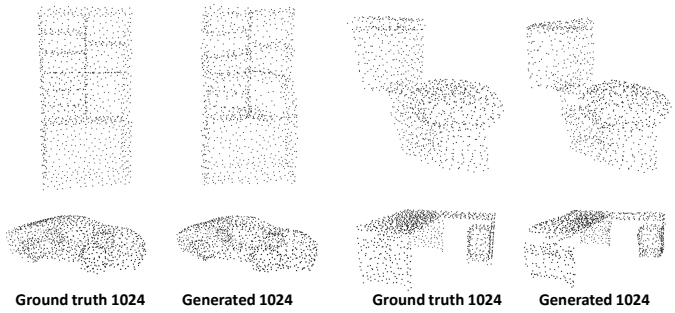

Fig. 10: Experimental verification that the learned function $\theta(\cdot)$ is an approximation of $\phi^{-1}(\cdot)$ in MOPS-Net.

we can conclude that $\theta(\cdot)$ indeed approximates $\phi^{-1}(\cdot)$, i.e. $\theta(\cdot) \approx \phi^{-1}(\cdot)$.

Advantage of the downsampling in high dimensional feature space. To demonstrate such an advantage, we modified MOPS-Net, leading to RMOPS-Net, as illustrated in Figure 11, where the learned sampling matrix is applied to the input point cloud directly, i.e., 3D coordinates of points, instead of high-dimensional features. Accordingly, RMOPS-Net with a post-matching operation is denoted by RMOPS-Net-M. We compared RMOPS-Net (resp. RMOPS-Net-M) with MOPS-Net (resp. MOPS-Net-M) under the classification task, and Table 7 lists the corresponding results. One can observe that MOPS-Net outperforms RMOPS-Net under all cases, demonstrating the advantage of such a process in feature space, which is coincident with the common-sense in machine learning. However, the gap between RMOPS-Net and RMOPS-Net-M is larger than that between MOPS-Net and MOPS-Net-M. The reason is that RMOPS-Net generates sampled points as a linear combination of the local points of the original point cloud, and there is a dominant weight among the weights, making the resulting points are close to the original points. Thus, they are slightly affected by the post-matching operation. However, MOPS-Net generates sampled points by regressing the locally aggregated high dimensional features. Although the generated points are either on or close to the underlying object surface, they are not necessary to be close to the original points. So, the post-matching operation changes the positions more seriously than that of RMOPS-Net, and thus compromises the accuracy more significantly.

TABLE 7: Comparisons of the classification accuracy of RMOPS-Net (resp. RMOPS-Net-M) and MOPS-Net (resp. MOPS-Net-M).

m	RMOPS-Net	RMOPS-Net-M	MOPS-Net	MOPS-Net-M
512	0.873	0.883	0.883	0.883
256	0.864	0.871	0.874	0.867
128	0.854	0.845	0.872	0.850
64	0.858	0.843	0.871	0.810
32	0.844	0.813	0.861	0.776
16	0.798	0.592	0.847	0.512

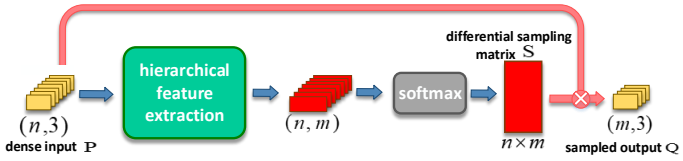

Fig. 11: The flowchart of RMOPS-Net, which is constructed by modifying MOPS-Net, i.e., applying the sampling matrix $\mathbf{S}$ to 3D coordinates instead of high dimensional features.

6 Conclusion & Future Work

In this paper, we presented MOPS-Net, a novel end-to-end deep learning framework for task-oriented point cloud downsampling. In contrast to the existing methods, we designed MOPS-Net from the perspective of matrix optimization. As the original discrete and combinatorial optimization problem is difficult to solve, we obtained a continuous and differentiable form by relaxing the 0-1 constraint of each variable. MOPS-Net elegantly mimics the function of the resulting matrix optimization problem by exploring both local and global structures of input data. MOPS-Net is permutation invariant and can be end-to-end trained with a task network. We applied MPOS-Net to three typical applications, classification, retrieval and reconstruction, and observed that MPOS-Net produced better results than the state-of-the-art methods. Moreover, MOPS-Net is flexible in that with simple modification, a single network with one-time training can handle arbitrary downsampling ratios. We justified our optimization driven design principle and

demonstrated the efficacy of MPOS-Net through extensive evaluations and comparisons.

The promising results of MOPS-Net inspire several interesting future directions. For example, the widely used FPS in feature extraction of current networks for 3D point clouds can be replaced to boost performance. Though MOPS-Net is designed for point cloud downsampling, increasing the dimension of differential sampling matrix allows us to handle upsampling as well. MOPS-Net links matrix optimization and deep learning in an elegant way. We believe the matrix optimization idea is general and it can be applied to other selection and ranking problems, such as key frame selection in videos [35], [36], band selection in hyperspectral images [28], [29] and view selection in light field images [37].

REFERENCES

[1] An Intrinsic Algorithm for Parallel Poisson Disk Sampling on Arbitra. *Ieee Trans. Vis. Comput. Graph.*. **19**, 1425–1437 (2013)

[2] Qi, C., Yi, L., Su, H., Guibas, L. Pointnet++: Deep hierarchical feature learning on point sets in a metric space.

[3] Wei, L. Parallel Poisson disk sampling. *Acm Transactions On Graphics (tog)*. **27**, 1–9 (2008)

[4] Bridson, R. Fast Poisson disk sampling in arbitrary dimensions.. *Siggraph Sketches*. **10** pp. 1278780–1278807 (200)

[5] Wu, W., Qi, Z., Fuxin, L. Pointconv: Deep convolutional networks on 3d point clouds.

[6] Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B. Pointcnn: Convolution on x-transformed points.

[7] Shen, Y., Feng, C., Yang, Y., Tian, D. Mining point cloud local structures by kernel correlation and graph pooling.

[8] Wang, Y., Sun, Y., Liu, Z., Sarma, S., Bronstein, M., Solomon, J. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*. **38**, 1–12 (2019)

[9] Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A. Xnor-net: Imagenet classification using binary convolutional neural networks.

[10] Krizhevsky, A., Sutskever, I., Hinton, G. Imagenet classification with deep convolutional neural networks.

[11] He, K., Zhang, X., Ren, S., Sun, J. Deep residual learning for image recognition.

[12] Dovrat, O., Lang, I., Avidan, S. Learning to sample.

[13] Li, J., Chen, B., Heelee, G. So-net: Self-organizing network for point cloud analysis.

[14] Tchapmi, L., Choy, C., Armeni, I., Gwak, J., Savarese, S. Segcloud: Semantic segmentation of 3d point clouds.

[15] Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M., Kautz, J. Splatnet: Sparse lattice networks for point cloud processing.

[16] Hua, B., Tran, M., Yeung, S. Pointwise convolutional neural networks.

[17] Angelinauy, M., Heelee, G. Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition.

[18] Kingma, D., Ba, J. Adam: A method for stochastic optimization. *Arxiv Preprint arXiv:1412.6980*. (201)

[19] Achlioptas, P., Diamanti, O., Mitliagkas, I., Guibas, L. Learning representations and generative models for 3d point clouds. *Arxiv Preprint arXiv:1707.02392*. (201)

[20] Yang, J., Zhang, Q., Ni, B., Li, L., Liu, J., Zhou, M., Tian, Q. Modeling point clouds with self-attention and gumbel subset sampling.

[21] Yu, L., Li, X., Fu, C., Cohen-or, D., Heng, P. Pu-net: Point cloud upsampling network.

[22] Nezhadarya, E., Taghavi, E., Liu, B., Luo, J. Adaptive Hierarchical Down-Sampling for Point Cloud Classification. *Arxiv Preprint arXiv:1904.08506*. (201)

[23] Yifan, W., Wu, S., Huang, H., Cohen-or, D., Sorkine-hornung, O. Patch-based progressive 3d point set upsampling.

[24] Li, R., Li, X., Fu, C., Cohen-or, D., Heng, P. Pu-gan: a point cloud upsampling adversarial network.

[25] Wolf, W. Key frame selection by motion analysis.

[26] Dirfaux, F. Key frame selection to represent a video.

[27] Zhuang, Y., Rui, Y., Huang, T., Mehrotra, S. Adaptive key frame extraction using unsupervised clustering.

[28] Guo, B., Gunn, S., Damper, R., Nelson, J. Band selection for hyperspectral image classification using mutual information. *Ieee Geoscience And Remote Sensing Letters*. **3**, 522–526 (2006)

[29] Wang, Q., Lin, J., Yuan, Y. Salient band selection for hyperspectral image classification via manifold ranking. *Ieee Transactions On Neural Networks And Learning Systems*. **27**, 1279–1289 (2016)

[30] Li, S., Qi, H. Sparse representation based band selection for hyperspectral images.

[31] Viola, I., Maretic, H., Frossard, P., Ebrahimi, T. A graph learning approach for light field image compression.

[32] Barrow, H., Tenenbaum, J., Bolles, R., Wolf, H. Parametric correspondence and chamfer matching: Two new techniques for image matching.

[33] Qian, Y., Hou, J., Kwong, S., He, Y. PUGeo-Net: A Geometry-centric Network for 3D Point Cloud Upsampling. *Arxiv Preprint arXiv:2002.10277*. (202)

[34] Chang, A., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H. Shapenet: An information-rich 3d model repository. *Arxiv Preprint arXiv:1512.03012*. (201)

[35] Wu, Z., Xiong, C., Ma, C., Socher, R., Davis, L. Adaframe: Adaptive frame selection for fast video recognition.

[36] Mei, S., Ma, M., Wan, S., Hou, J., Wang, Z., Feng, D. Patch based Video Summarization with Block Sparse Representation. *Ieee Transactions On Multimedia*. **PP** pp. 1-1 (2020)

[37] Jin, J., Hou, J., Chen, J., Zeng, H., Kwong, S., Yu, J. Deep coarse-to-fine dense light field reconstruction with flexible sampling and geometry-aware fusion. *Arxiv Preprint arXiv:1512.03012*. (201)

[38] Sun, Y., Wang, Y., Liu, Z., Siegel, J., Sarma, S. Pointgrow: Autoregressively learned point cloud generation with self-attention.

[39] Elbaz, G., Avraham, T., Fischer, A. 3D point cloud registration for localization using a deep neural network auto-encoder.

[40] Aoki, Y., Goforth, H., Srivatsan, R., Lucey, S. PointNetLK: Robust & efficient point cloud registration using PointNet.

[41] Kung, S. Kernel methods and machine learning. (Cambridge University Pres,2014)

[42] Qi, C., Su, H., Mo, K., Guibas, L. Pointnet: Deep learning on point sets for 3d classification and segmentation.

[43] Yew, Z., Lee, G. 3DFeat-Net: Weakly supervised local 3D features for point cloud registration.

[44] Zhou, Y., Tuzel, O. Voxelnet: End-to-end learning for point cloud based 3d object detection.

[45] Li, B. 3d fully convolutional network for vehicle detection in point cloud.

[46] Shi, S., Wang, X., Li, H. Pointrcnn: 3d object proposal generation and detection from point cloud.

[47] Engelcke, M., Rao, D., Wang, D., Tong, C., Posner, I. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks.

[48] Lang, A., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O. Pointpillars: Fast encoders for object detection from point clouds.

[49] Zaganidis, A., Sun, L., Duckett, T., Cielniak, G. Integrating deep semantic segmentation into 3-d point cloud registration. *Ieee Robotics And Automation Letters*. **3**, 2942–2949 (2018)

[50] Li, C., Zaheer, M., Zhang, Y., Poczos, B., Salakhutdinov, R. Point cloud gan. *Arxiv Preprint arXiv:1810.05795*. (201)

[51] Kuang, Z., Yu, J., Fan, J., Tan, M. Deep point convolutional approach for 3D model retrieval.

[52] Hu, Q., Yang, B., Xie, L., Rosa, S., Guo, Y., Wang, Z., Trigoni, N., Markham, A. RandLA-Net: Efficient semantic segmentation of large-scale point clouds. *Arxiv Preprint arXiv:1911.11236*. (201)

[53] None. *The Farthest Point Strategy For Progressive Image Sampling*. **6**, 1305-1315 (1997)

[54] Kimmel, R., Sethian, J. Computing geodesic paths on manifolds. *Proceedings Of The National Academy Of Science*. **95**, 8431–8435 (1998)