Rethink Geometry Feature Representation for Point Cloud Classification

Wang Jiangyi, A0236307J
National University of Singapore
e0732857@u.nus.edu

Abstract

Point cloud analysis is challenging due to its unorderness, sparsity and irregularity. Recent works attempt to capture local geometry via sophisticated local geometry extractors. These methods rely on convolution operation or attention mechanism to exploit geometric information from coordinates directly. However, we argue that, these approaches are insufficient to describe the local geometry. In this report, we present novel Modified Point Feature Histogram, a novel representation of point clouds to capture local orientation information. This Point Feature Histogram arises from tradition computer vision society, and we conduct modification to enhance its feature representation. Modified Point Feature Histogram can be a plug-and-play module for most previous models. We carefully examine the optimal way to combine Modified Point Feature Histogram with existed basic models, including PointNet and PointNet++. Based on the baseline of PointNet, Modified Point Feature Histogram improves the instance accuracy from ***89.6%*** *to* ***92.4%*** *(+2.8%) on the classification task of ModelNet40 benchmark. Moreover, combining with PointNet++, the instancy accuracy can be improved from* ***90.6%*** *to* ***93.3%*** *(+2.7%). Meanwhile, this method can attain 91.2% mean accuracy, which is comparable with previous state-of-the-art RepSurf.*

1. Introduction

Recently, point cloud analysis has rasied great interest for the development of sensors like LiDAR and photogrammetry. Point clouds can be used for various purposes, including autonomous dribing, robotics and virtual and augmented reality. However, compared with other data type like images, the main obstacles for point cloud data are, unorderness, sparsity and irregularity. On one hand, tne point cloud data is composed of N unordered and irregular set of data $\mathbb{P} \in \mathbb{R}^{N \times 3}$, which makes it infeasible to apply those image processing techniques directly. On the other hand, if we naively translate point cloud data into 3d axis and process with 3d convolutions, then it is computationally intractable due to the sparsity.

According to the success from Convolution Neural Network [5, 6, 20] in image processing community, great efforts have been made for the ideal network architecture of point clouds [14, 15]. The pioneering work of network architecture which directly works with point cloud is, PointNet [14]. Take point cloud classification task as one example, PointNet utilizes mutiple shared-MLP followed by max pooling to work as the feature extractor. It is able to generate one global feature vector for each point cloud object. Until now, it is still the backbone for many advanced network architecture [7, 15, 22]. The main drawbacks of PointNet is, the loss of local geometry information due to the max pooling operator. To release this issue, PointNet++ [15] attempts to extract local geometry information in a hierarchical manner. It mimics the design of Convolution Neural Network and achieves great success. Lately, great progress has been made by exploring local geometry via convolution operation [7, 9, 22] or attention mechanism [4, 24, 26]. Among these approaches, they both designed sophisticated feature extractor to enhance the quality of local feature representation.

However, the booming sophisticated extractors will saturate the performance since they have already described the local geometry pretty well from the network architecture side [12]. Therefore, we turn to rethink the sucess of Convolution Neural Network. The convolution layer in Convolution Neural Network has its origin in the filter from tradition computer vision community. This may illustrate that, Convolution Neural Network introduces the inductive bias efficiently via convolution operation. While for point cloud task, tradition computer vision will not utilize architectures like convolution or attention mechanism to capture local geometry from coordinates. Based on this intuition, we emphasize that, we will not let Multi-Layer Perceptron (MLP) learn high-level features from coordinates only.

In this report, we modify the Point Feature Histogram and propose a novel geometry feature, Modified Point Feature Histogram from tradition computer vision, which can capture local orientation information via histogram representation. Furthermore, based on the above reasoning, we concatenate Modified Point Feature Histogram with coordinates and let

MLP learn high-level features from them together. Although it might not be the best solution to capture local geometry, we believe that it is a more reasonable approach than extracting features from coordinates only via MLP.

The contributions of our work are:

- We propose a novel geomety feature, Modified Point Feature Histogram, which is able to capture local orientation information via histogram representation.
- We parallelize the computation for Modified Point Feature Histogram, which allows us to attain the feature representation efficiently.
- We design a hierarchical approach to combine Modified Point Feature Histogram with PointNet and PointNet++.

Even though the design is not complicated, experiments show that our proposed approach is highly competitive against other network architecture in classification task. All experiments are conducted on ModelNet40 [23] Dataset, which is a benchmark dataset for classification task. With the help of Modified Point Feature Histogram, we increase the instance accuracy of PointNet baseline from **89.6%** to **92.4%** (+2.8%). Moreover, we also improve the instance accuracy of PointNet++ from **90.6%** to **93.3%** (+2.7%).

2. Related Work

One popular approach to work with point clouds is, voxelizing them into volumetric grids by quantization and then apply 3D convolution networks. This type of works is constrained by the computational costs of 3D convolution due to the sparsity of point clouds. Many works [13, 23] are aimed to design efficient approaches to implement convolution opeartions. However, if the point cloud is sparsely sampled, particularly with an uneven sampling rate, regions that are sparsely sampled may not have any neighbors within the volumetric convolutional filter. This will lead to potentially significant issues.

Instead of voxelization, many recent works attempt to work with raw point clouds directly. PointNet [14] proposed to use several shared-MLP to extract features for each point, followed by one max pooling layer, which augments local features to one global feature. However, the limitation of PointNet is, the last max pooling layer will suffer from the loss of local features. PointNet++ [15] mimics the design of CNN, which applies a hierarchical structure to extract features. It proposed novel Set Abstraction (SA) Layer, which allows us to extract features from small local regions and gradually extending to larger regions. Since PointNet++ still utilizes PointNet as its local feature extractor, it will still potentially lose local geometry information due to max pooling operation. Based on this limitation of PointNet++, many works [7, 9, 22] tend to build sophisticated local feature extractor. PointCNN [9] attempted to learn a χ-transformation from input point clouds in order to re-organize inputs into canonical order. After that, utilizing normal convolution operation to extract features. Instead of reordering the input point clouds, PointConv [22] focused on approximating continuous weight and density functions in convolutional filters via MLP. Since it learnt convolution weights from input points, it satisfied the permutation-invariance property because the weight of each point is only related to its coordinates.

More recently, Transformer and self-attention models [2, 16, 21] have revolutionized machine translation and natural language processing. Moreover, many works [3, 10] have succeeded in applying attention mechanism into 2D image processing. ViT [3] viewed one image as several tokens and applied Transformer on these tokens directly. Swin-Transformer [10] tended to apply Transformer in the local patch, which can greatly shrink the computational costs. There are also a number of previous works [4, 24, 26] apply attention mechanism [21] on point clouds. Point Cloud Transformer [4] applied Transfomer architecture on point cloud directly. It utilized global attention within point clouds. To avoid heavy computational costs, Point Transformer [26] applied attention mechanism on local regions to enhance the local feature representation. There are also some works [12, 17] which do not attempt to design novel architectures. PointMLP [12] argued the effectiveness of sophisticated local feature extractor. It proposed deep shared-MLPs to extract features. RepSurf [17] attempted to utilize umbrella surface to characterize relations between points via quantities like normals and angles, followed by MLPs to enhance feature representation. Both of these two approaches achieve the state-of-the-art result without the help of sophisticated local feature extractor.

3. Data

Our report presents experiments conducted on the ModelNet40 [23] Dataset. The ModelNet40 dataset is a popular benchmark dataset used for 3D object recognition and classification, particularly in the context of point cloud data and deep learning-based approaches. It consists of over 12,000 Computer-Aided Design (CAD) models of 40 different object categories, with each object represented as a point cloud consisting of a set of approximately 10,000 points in 3D space. The dataset has been widely used to evaluate the performance of neural network architectures for point cloud classification tasks, such as PointNet and its variants. The diverse range of object categories in ModelNet40, along with variations in shape and size within each category, presents a challenging task for point cloud classification algorithms, and makes it an important dataset for researchers to develop and evaluate new approaches for this task.

In Modelnet40 Dataset, some categories may share great similarity, which makes the classification extremely difficult.

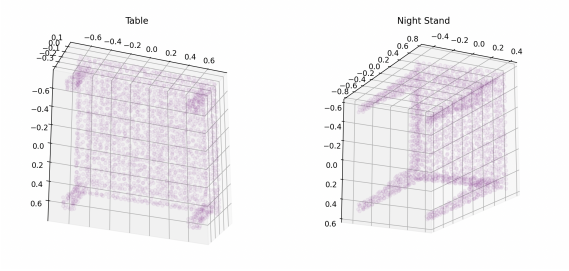


Figure 1. **'Table' and 'Night Stand' Point Clouds**. We visualize these two similar objects on ModelNet40 Dataset.

Fig. 1 shows two visualization examples on ModelNet40 Dataset. For one human-being, it is also a difficult task to distinguish tables and night stands. From these two examples, we should understand that, the point cloud classification task is not as easy as we thought.

4. Method

4.1. Tradition Approach

We propose a novel geometry feature, **Modified Point Feature Histogram (MPFH)**, which extends from Point Feature Histogram in Tradition Computer Vision community. Sec. 4.1.1 introduces the normal estimation via PCA [1]. Sec. 4.1.2, 4.1.3, 4.1.4 introduce Point Feature Histogram-based methods in details.

4.1.1 Normal Estimation

Normal estimation is required for the computation of Point Feature Histogram. Generally speaking, there are two approaches to derive normal estimation. First is through the local quadratic plane fitting [25], which will give the closed-form normal vector. Second is through Principal Component Analysis [1] (PCA). Here, we apply the second approach since it is computationally efficient and more robust to the irregular shape of local plane.

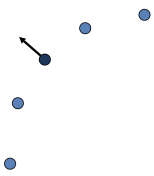


Figure 2. **Point Cloud and Normal**. The vector is normal direction for the middle point cloud data. It corresponds to the principal component with smallest eigenvalue for these five points.

Fig. 2 illustrates the intuition of normal estimation via PCA. Due to the fact that, principal component corresponding to the smallest eigenvalue represents the direction that explains the least amount of data variance. This exactly corresponds to the normal vector we want. Therefore, we can use PCA to estimate the normal approximately.

4.1.2 Point Feature Histogram

In traditonal computer vision framework, most algorithms [11] can be decomposed into two stages. The first stage is keypoint detection and the second is keypoint description. Here, we mainly focus on the keypoint description since our aim is to combine Deep Learning models with stronger geometry feature representation. One famous keypoint descriptor for point cloud processing is Point Feature Histogram [19].

Generally speaking, Point Feature Histogram [19] representation is based on the relationships between the points in the k-neighborhood and their estimated surface normals. It mainly focuses on captureing the k-neighborhood local curvature information for one specific point via angle hitogram.

The derivation of Point Feature Histogram can be decomposed into the following the following 3 steps:

- For interested point p^*, find all pairs $\{(p_i, p_j) : i<j\}$ within its k-neighborhood.
- For all $\{(p_i, p_j) : i<j\}$ pairs, calculate Point Features.
- Construct histogram based on Point Features and record as one vector f^* for total $\{(p_i, p_j) : i<j\}$ pairs.

Here, each entry of f^* represents the frequency of Point Features in each bin. Moreover, this vector f^* is exactly the Point Feature Histogram for the given interested point p^*.

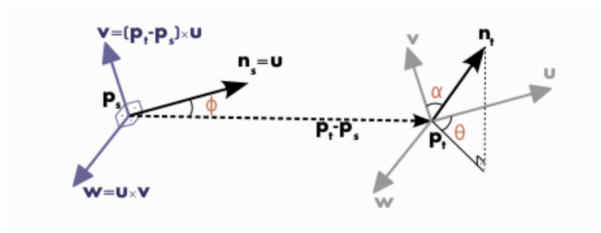


Figure 3. **PFH Derivation**. (u, v, w) is the local reference frame achieved via local direction $p_t - p_s$ and normal n_s. PFH captures local curvature information via Point Features (α, ϕ, θ).

The most sophisticated part is, how to derive the Point Feature for given point pair (p_i, p_j). The idea is illustrated in Fig. 3. Here, we introduce local reference frame (u, v, w) to characterize the relative location between two points. Then, we utilize three relative angles to encode local curvature

information, which are the Point Features we need.

Before calculating the three angles, the first thing we should do is to determine (p_s, p_t) for an unordered point set (p_i, p_j) such that it is irrelevant to order of points (p_i, p_j). Here, p_s represents the starting point and p_t represents the terminating point.

One approach to achieve that is, for each point, calculate the angle between its normal vector and direction to the other point. After that, without loss of generality, we can assign the point with smaller angles as starting point. Following this scheme, we can guarantee that (p_s, p_t) is invariant to the order of (p_i, p_j).

To dervie the three Point Features, we construct the Local Reference Frame (u, v, w) first. It can be derived from coordinates (p_s, p_t) and normals (n_s, n_t) as:

$$\begin{cases} u = n_s \\ v = u \times \dfrac{(p_t - p_s)}{||p_t - p_s||_2} \\ w = u \times v \end{cases} \tag{1}$$

Based on this, we construct the Local Reference Frame, and we are able to derive the Point Features as:

$$\begin{cases} \alpha = v \cdot n_t \\ \phi = u \cdot \dfrac{(p_t - p_s)}{||p_t - p_s||_2} \\ \theta = \arctan(w \cdot n_t / u \cdot n_t) \end{cases} \tag{2}$$

To gain more insights from Equ. 2, we make more explanations for these three angles from geometry perspective.

In Equ. 2, v is the perpendicular to the normal of starting point n_s and direction from starting point to terminating point $p_t - p_s$. Equivalently, v is the normal vector of the plane spanned by normal n_s and relative direction $p_t - p_s$. For illustration purpose, we denote it as **Critical Plane**. Therefore, Point Feature α measures whether the normal of terminating n_t is closer to the Critical Plane. Similarly, ϕ characterizes the relationship between starting point normal n_s and relative direction $p_t - p_s$. Lastly, θ represents how terminating point normal is oriented in $(u, w)-$plane.

Until now, for a given interested point p^*, we can calculate all Point Features F^* for $\{(p_i, p_j) : i<j\}$ within k-neighborhood, namely $F^* = \{f_k : k = 1, ..., \frac{k(k-1)}{2}\}$. By determining the number of bins N_{bin}, the final Point Feature Histogram for p^* is $f^* \in \mathbb{R}^{3 \cdot N_{bin}}$. Moreover, from the statistics perspective, the Point Feature Histogram characterzies the empirical marginal distribution of (α, ϕ, θ) within the k-neighborhood.

Notice that, here we have three hyper-parameters to tune: number of neighbors k, number of bins N_{bin} and number of total points N_{total} to generate Point Feature Histogram. Actually these three hyper-parameters will greatly influence the quality of feature representation. We will conduct comprehensive experiments in Sec. 5.

4.1.3 Fast Point Feature Histogram

The main issue for Point Feature Histogram is the computational costs. Assume that we use k-neighborhood to compute the Point Feature Histogram, we are required to take all $\frac{k(k-1)}{2}$ neighbor pairs into consideration for a specific interested point p^*. Therefore, for one interested point p^*, the computational cost is $O(k^2)$, which is unaffordable.

To release this issue, Fast Point Feature Histogram [18] (FPFH) proposes $O(k)$ algorithm to approximate Point Feature Histogram. The key idea is, we only consider those k pairs which are directly connected with p^* within k-neighborhood. We use these pairs to construct the Simple Point Feature Histogram (SPFH), which shrinks the neighbor pairs from $\frac{k(k-1)}{2}$ to k. After that, we construct the FPFH for p^* via inversely weighted average scheme of SPFH within its k-neighborhood. The detailed derivation is:

$$FPFH(p^*) = SPFH(p^*) + \frac{1}{k}\sum_{i=1}^{k}\frac{1}{w_i}SPFH(p_i) \tag{3}$$

In FPFH, our choice of w_i is, $w_i = ||p_i - p^*||_2$. It means that, for those points closer to interested point p^*, they will contribute more to the FPFH feature, which is a reasonable setting.

4.1.4 Modified Point Feature Histogram

We propose our Modified Point Feature Histogram based on these two modifications for FPFH [18]:

- Normalize on Local Reference Frame to attain Angle Point Features.
- γ-Weighted Average Scheme.

Firstly, instead of using dot-product values to characterize the relation between two vectors, we turn to use relative angle, which is a higher-level feature. To achieve that, we normalize the reference frame as:

$$\begin{cases} u = n_s \\ v = \dfrac{\hat{v}}{||\hat{v}||_2}, \text{where } \hat{v} = u \times \dfrac{(p_t - p_s)}{||p_t - p_s||_2} \\ w = \dfrac{u \times v}{||u \times v||_2} \end{cases} \tag{4}$$

With the help of orthonormal reference frame, we can easily

derive the corresponding angle with high-level semantic meanings as:

$$\begin{cases} \alpha = \arccos(v \cdot n_t) \\ \phi = \arccos(u \cdot \dfrac{(p_t - p_s)}{||p_t - p_s||_2}) \\ \theta = \arctan(w \cdot n_t / u \cdot n_t) \end{cases} \quad (5)$$

Secondly, we propose γ-weighted average scheme to guarantee that our MPFH is still a well-defined histogram representation. That is, for Modified Point Feature Histogram $MPFH(p^*) \in \mathbb{R}^{3 \cdot N_{bin}}$, we want to guarantee:

$$\sum_{j=0}^{N_{bin}-1} MPFH(p^*)[i \cdot N_{bin} + j] = 1, \text{for } i = 0, 1, 2 \quad (6)$$

Therefore, one possible way to achieve this is, constructing MPFH via convex combination of SPFH, which is defined in Equ. 7:

$$MPFH(p^*) = \gamma \cdot SPFH(p^*) + (1-\gamma) \cdot \sum_{i=1}^{k} \frac{1}{k} SPFH(p_i) \quad (7)$$

In Equ. 7, we introduce hyper-parameter $\gamma \in [0, 1]$ to balance our attention on interested point p^* and its k-neighborhood $\{p_i\}_{i=1}^{k}$. When $\gamma = 0$, MPFH degrades to SPFH feature. In the following experiments, we set $\gamma = \frac{1}{2}$, which places equal attention to the interested point p^* itself and its neighbors.

4.2. Deep Learning Approach

We propose an approach which combines MPFH with PointNet [14] and PointNet++ [15] via concatenating MPFH with coordinates together. Sec. 4.2.1 introduces the pioneering work PointNet. Sec. 4.2.2 introduces PointNet++, the modification of PointNet. Sec. 4.2.3 introduces our proposed method.

4.2.1 PointNet

PointNet [14] is a pioneering work in deep learning architecture, which is developed for point cloud processing tasks. The main challenge in network design for point cloud data is its unorderness and irregularity. To overcome this issue, PointNet proposes 'max pooling' module to extract global features from local point features.

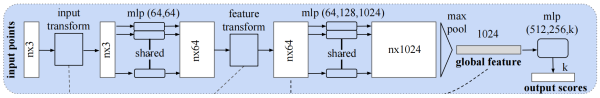


Figure 4. **PointNet**. It generate high-level features via three MLPs followed by one max pooling layer from coordinates only.

Fig. 4 illustrates the main architecture of PointNet. Generally speaking, it can be decomposed into 2 stages. In Stage 1, PointNet utilizes three shared-MLPs to extract features from individual points. We hope to generate high-level features from coordinates via MLP. In Stage 2, when we achieve the high-level feature representation for individual points, it utilizes one max pooling layer to achieve the global features. Lastly, the output of the max pooling layer can be fed into one MLP classifier for classification tasks.

Furthermore, PointNet introduces T-net architecture, which is a learnable transformation matrix. It is aimed to align the point clouds into canonical coordinates, in order to increase the robustness of the network.This will improve the quality of final global features.

While PointNet is a popular method for analyzing point clouds, it has the limitation that can lead to the loss of local geometry information. This is due to the use of a max pooling operation, which ignores information from neighboring points. As a result, PointNet relies solely on the shared-MLP to extract local features. This will lower the quality of our features.

4.2.2 PointNet++

PointNet++ [15] is proposed to release the loss of local geometry information issue of PointNet [14]. PointNet++ utilizes a hierarchical neural network architecture to capture local geometry at different scales. Generally speaking, PointNet++ utilizes a set of PointNet modules at each hierarchical level. Each PointNet module in one specific level processes only one region of point cloud data and generates the feature representation for this region. This approach enables PointNet++ to capture the local geometry information and augment features in a hierarchical manner.

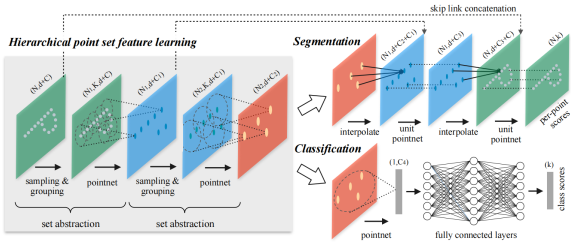
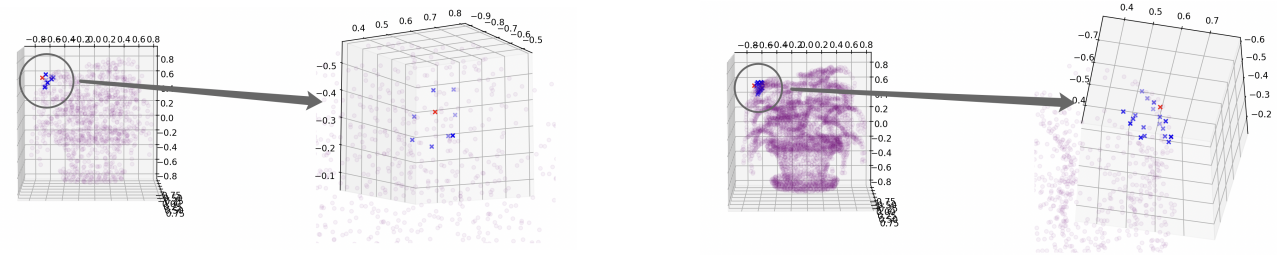


Figure 5. **PointNet++**. It utilizes hierarchical approach to extract features with the help of Set Abstraction Layer.

Fig. 5 shows the general architecture of PointNet++. It proposes Set Abstraction Layer ('Sampling & Grouping ' in Fig. 5) to extract features in a hierarchical manner. It utilizes Farthest Point Sampling (FPS) algorithm to select a set of representative points and aggregates the local features around those points via PointNet. Moreover, Set Abstraction



(a) $N_{total} = 1000, k = 8$.

(b) $N_{total} = 4000, k = 18$.

Figure 6. **Local neighborhood comparison**. (a) it seems that eight neighbors lie on one quadratic plane; (b) eighteen neighbors are distributed within two clusters, corresponding to two leaves in flower pot.

Layer mimics the design of CNN in 2D image processing area. To be more specific, FPS algorithm corresponds to pooling layer in CNN, which is able to down-sample the point clouds. Cooperating with k-Nearst-Neighbors algorithm, PointNet mimics the convolution operation for a local region. Combining these two designs together, we achieve our 'Set Abstraction' Layer, which allows us to extract features hierarchically.

Compare with PointNet, PointNet++ is able to capture the local geometry information in a hierarchical manner. However, it still has the limitation that, it is difficult to capture the local features efficiently since it applies vanilla PointNet as its local feature extractor.

4.2.3 Proposed Approach

Many recent works [4, 7, 9, 22, 24, 26] attempted to design sophisticated architecture to capture local geometry information. When we rethink the design of CNNs, their convolution layer has its origins in filters from tradition computer vision. Based on this, we think that, the design of convolution layers in CNN efficiently introduces the inductive bias. Therefore, it is easier for CNNs to achieve high-quality feature representation.

However, in point cloud tasks, we will not utilize architectures like MLP or attention in tradition computer vision. Instead, since the point clouds are intrinsically embedded in the 3D world coordinate system, we will try to understand them via geometry in a mathematical way. This is totally different from the case in CNN. Therefore, we believe that it is **inefficient** to let MLPs learn high-level features from coordinates only.

Based on this consideration, instead of designing sophisticated architecture to capture local geometry, we propose our approach that, combining tradition computer vision with deep learning techniques directly. We attempt to utilize novel MPFH from traditon computer vision society, which is able to let models understand the geometry information easier. In implementation, we concatenate MPFH with coordinates (or high-level features encoded from coordinates), followed by some MLPs to enhance the feature representation. We believe that, this approach will place more prior knowledge to our neural network with respect to local geometry information. Similar with the design of CNN, due to the correct inductive bias, our model is able to generalize better with smaller size of dataset.

5. Discussion

5.1. Experiments on MPFH

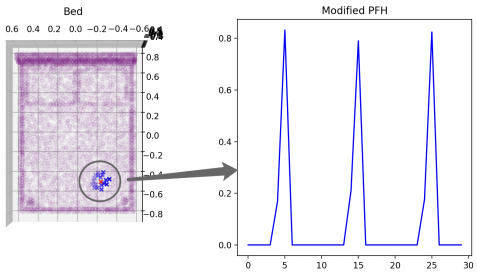
To combine MPFH with PointNet [14] and PointNet++ [15], we try to understand the MPFH first. Here, we are interested in the following two questions:

- How to determine number of input point cloud data N_{total} and number of neighbors k?
- How to interpret MPFH features?

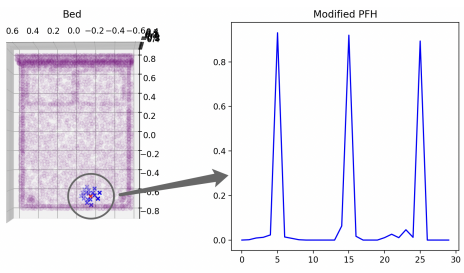
5.1.1 MPFH Hyper-parameters Selection

Here, we carefully examine how hyper-parameters influence the quality of MPFH features. We mainly focus the following two hyper-parameters, number of total point cloud data N_{total} and number of neighbors k. For simplicity, we fix the number of bins $N_{bin} = 10$, which means each bin contains 18-degree angle.

Here, we use the 'flower pot' sample in ModelNet40 Dataset to visualize the influence of different hyper-parameters. Fig. 6 illustrates the local neighborhood differences under different hyper-parameters setting. We compare two choices of hyper-parameters, $N_{total} = 1000, k = 8$ and $N_{total} = 4000, k = 18$ via plotting the interested point p^* in red and its k-neighborhood in blue. In Fig. 6a, it shows that all eight neighbors around p^* come from one quadratic plane approximately. However, the ground truth is, the interested point p^* comes from one leaf. Therefore, it is more reasonable to expect the neighbors of p^* to have cluster property. This is exactly what we have in Fig. 6b, all eighteen



(a) p^* in the Center of Flat Plane.



(b) p^* in the Boundary of Flat Plane.

Figure 7. **MPFHs corresponding to different locations of a flat plane**. (a) interested point p^* lies in the center of flat plane, which leads to the sparse MPFH with three skews; (b) interested point p^* is near the boundary of flat plane, which makes MPFH non-sparse and fluctuate slightly, but still with three skews.

neighbors are distributed within two clusters, which corresponds to the two leaves in flower pot.

It is worthwhile to mention that it is difficult to classify the 'flower pot' class correctly since it has complicated patterns in local regions, which will confuse our model. If we directly apply PointNet/PointNet++, many samples in this class will be misclassified. With the help of MPFH under appropriate hyper-parameters, it will be easier for model to capture the complicated patterns of 'flower pot', , which is able to benefit the model performance.

This emphasizes the significance of choosing appropriate hyper-parameters. If we do not select enough points, then our neighborhood is biased. This will greatly decreases our quality of MPFH. However, if we enlarge N_{total}, then we need to enlarge k to maintain the scale of neighborhood, which is computationally inefficient. Therefore, the guideline is, we should choose the intermediate value of N_{total} to achieve the trade-off between quality of local neighborhood and computational costs. Furthermore, the choice of k should be comparable to N_{total} to guarantee the scale of neighborhood.

5.1.2 MPFH Interpretation

Here, we attempt to interpret MPFH features. This will help us us understand how MPFH helps our model make prediction. To illustrate this, we consider the simplest case, that is, the interested point p^* exactly lies on a flat plane. Fig. 7 examines how different locations of flat plane will influence the MPFH feature representation. The point cloud data we visualize is the 'bed' in ModelNet40 Dataset. We set $N_{total} = 4000, k = 18, N_{bin} = 10$ to generate MPFH. Moreover, we choose $\gamma = \frac{1}{2}$ in Equ. 7, which places equal attention on interested point p^* and its k-neighbors.

Fig. 7a shows the situation that, when interested point p^* lies in the flat plane and is far away from boundary, the MPFH is sparse and there is exactly one skew in histogram for each angle α, ϕ, θ. This phenomenon is mainly caused by the fact that, for those points on a flat plane, they will share the same *relative orientation tendency*. Here, for arbitrary two points, we use *relative orientation tendency* to stand for the patterns between their relative direction and their normals. Since these patterns are exactly the same for those points on a flat plane, for each α, ϕ, θ, angles should be concentrated on one bin of histogram. Therefore, MPFH is sparse and there is exactly one skew for each relative angle.

Fig. 7b corresponds to the secnario that interested point p^* is near the boundary of a flat plane. We also have three skews in MPFH. However, the difference is, now MPFH is no longer sparse. Instead, in most bins, it has non-zero but small values. This phenomenon is because, our intersted p^* will have some neighbors that exactly lie on the boundary or on some other planes. Now, the *relative orientation tendency* will be different between neighbors. On one hand, since most of the points lie on the flat plane, we will still achieve three skews in MPFH. On the other hand, due to the existence of neighbors lying on the boundary, MPFH is non-sparse and have some small values in most bins.

To conclude, those large skews will explain the global curvature information of one plane. Those small fluctuations will capture how those neighbors are oriented from the local perspective. From this perspective, MPFH can be viewed as one tag for interested point p^*, which encodes local geometry information via $(3 \cdot N_{bin})$-dim vector. With the help of MPFH, it will be easir for MLP to generate high-level features compared with utilizing coordinates only.

5.2. Experiments on PointNet and PointNet++

We comprehensively examine the optimal approach to combine MPFH with basic neural network models, PointNet [14] and PointNet++ [15]. We conduct experiments on ModelNet40 [23] Dataset, which is one benchmark for point cloud classification task. There are 12,311 CAD models from 40 man-made object categories. We use the official

split with 9,843 shapes for training and 2,468 for testing. In all experiments, we implement the models with PyTorch on one A40 GPU. The optimizer we use is Adam [8] with default parameters $\alpha = 0.9$ and $\beta = 0.999$. We apply exponential learning decay scheme, of which step size $d = 20$ and decay rate $\gamma = 0.7$. Moreover, we conduct point cloud normalization on ModelNet40 Data such that they are in a unit sphere. During training, we augment input point clouds via randomly rotating and dropout. We will also jitter the position of each point by a Gaussian noise with zero mean and 0.02 standard deviation.

For point cloud classification task, there are two popular performance metrics, instance accuracy and class accuracy. Instance accuracy is the common accuracy measure, which explains the percentage of samples are correctly classified. Class accuracy is the mean instance accuracy with resepect to total classes. Generally speaking, class accuracy is a more robust performance metric since it will compensate for minority class.

Sec. 5.2.1 conducts ablation studies on the design of PointNet and PointNet++ with MPFH. We propose a hierarchical approach to work with MPFH. Sec. 5.2.2 compares between PointNet++ baseline and PointNet++ with MPFH. Sec. 5.2.3 conducts comparison between PointNet++ with MPFH and RepSurf.

5.2.1 Ablation Studies

Firstly, we determine the optimal choice of MPFH hyper-parameters. Based on the guideline in Sec. 5.1.1, we predefine four choices of hyper-parameters as shown in Fig. 8.

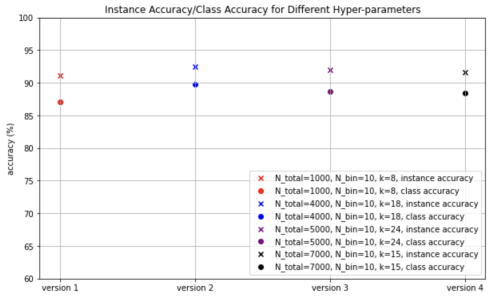


Figure 8. **Instance Accuracy and Class Accuracy Comparisons**. Dot sign represents for class accuracy and cross sign represents for instance Accuracy. Different colors correspond to different choices of hyper-parameters.

Fig. 8 compares the instance accuracy and class accuracy between different choices of hyper-parameters. The dot sign stands for class accuracy and cross sign stands for instance accuracy. We use four different colors to represent four different choices of hyper-parameters. It can be observed that from Fig. 8 that, $(N_{total} = 4000, N_{bin} = 10, k = 18)$ achieves the best instance accuracy and class accuracy and outperforms other choices of hyper-parameters. This illustrates that, under this hyper-parameter setting, MPFH features tend to have the richest semantic meanings, at least in ModelNet Dataset. Therefore, in our following experiments, we will apply $(N_{total} = 4000, N_{bin} = 10, k = 18)$ for better model performance.

Secondly, we attempt to figure out the optimal approach to combine MPFH with PointNet [14] together. In PointNet architecture, which in shown in Fig. 4, we have three shared-MLP layers followed by one pooling layer. This means that, we totally have four positions where we can concatenate our MPFH features. We conduct ablation study on how the concatenation position influences the model performance. In Table 1, we report instance accuracy and class accuracy for different concatenations of MPFH.

Method	ModelNet40 Dataset	
	Instance Accuracy	Class Accuracy
PointNet (Baseline)	89.6	86.3
PointNet + Normal (Layer 1)	91.3	87.7
PointNet + MPFH (Layer 1)	91.9	89.2
PointNet + MPFH (Layer 3)	**92.4**	**89.9**
PointNet + MPFH (Pooling Layer)	90.9	87.6

Table 1. **Ablation Study on Design of PointNet with MPFH**. Metrics are instance accuracy and class accuracy. We compare these two metrics under different model designs. Concatenating MPFH in Layer 3 outperforms other designs significantly.

There are several observations from Table 1:

- **Observation 1**: MPFH is a superior feature representation compared to normal vector as it contains more geometry information for local regions.
- **Observation 2**: MPFH feature representation can be enhanced via MLPs. With concatenating MPFH in pooling layer, the instance accuracy and class accuracy is poorer than concatenating in previous MLP layers. This emphasizes that, MLP still can generate feature with richer semantic meanings based on MPFH.
- **Observation 3**: Concatenating MPFH in Layer 3 outperforms that in Layer 1. This is a surprising but reasonable result due to the fact that, MPFH is the feature representation generated from coordinates. This is the intrinsically hierarchical relationship between coordinates and MPFH. It means MPFH is a higher-level feature representation compared with coordinates. If we simply concatenate MPFH in Layer 1, then MLP is likely to focus too much on the high-level feature MPFH. This will restrict our model from learning better feature representation. By concatenating MPFH in later Layer, MLP is able to make full use of coordinates,

which is a better design for the feature representation learning process.

To conclude, Table 1 suggests us to combine MPFH with PointNet in a hierarchical manner, which allows us to achieve the best performance. Based on this design, we can improve the instance accuracy of PointNet from **89.6%** to **92.4%** (+2.7%).

To verify this suggestion, we conduct one more experiment which attempts for the optimal approach to combine MPFH and normal with PointNet. In Table 2, we report instance accuracy and class accuracy for different concatenations of MPFH and normal.

Method	ModelNet40 Dataset	
	Instance Accuracy	Class Accuracy
PointNet (Baseline)	89.6	86.3
PointNet + Normal (Layer 1)	91.3	87.7
PointNet + Normal (Layer 1) + MPFH (Layer 1)	92.1	89.7
PointNet + Normal (Layer 1) + MPFH (Layer 3)	**93.0**	**90.8**
PointNet + Normal (Layer 1) + MPFH (Pooling Layer)	92.0	87.6

Table 2. **Ablation Study on Design of PointNet with MPFH and Normal**. Metrics are instance accuracy and class accuracy. Concatenating normal in Layer 1, MPFH in Layer 3 will also result in best performance.

In Table 2, similar with the case in Table 1, we achieve the best performance when we concatenate MPFH in share-MLP Layer 3. This result corresponds to our **Observation 3**, that is, we should combine MPFH with PointNet hierarchically.

5.2.2 Comparison between PointNet++ baseline and PointNet++ with MPFH

Since PointNet++ [15] utilizes PointNet [14] module as its local feature extractor, we follow the design suggestion from Sec. 5.2.1 to concatenate MPFH hierarchically. Based on this design, we conduct experiments to compare the performance with PointNet++ baseline. Fig. 9 reports testing instance accuracy comparison results.

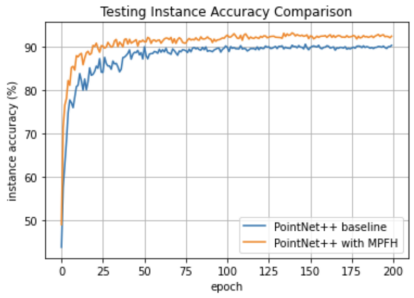


Figure 9. **Testing Instance Accuracy Comparison on PointNet++**. Blue color corresponds to PointNet++ baseline; Orange color corresponds to our proposed design.

Fig. 9 illustrates the effectiveness of our proposed PointNet++ with MPFH compared with PointNet++ Baseline. Due to MPFH features, our proposed PointNet++ converges faster to a better testing instance accuracy value. This means that, our proposed PointNet++ is not only more powerful with respect to model performance, but also more efficient. More detailed comparison result is shown in Table 3.

Model	Instance Accuracy
PointNet++ (Baseline)	90.6
PointNet with MPFH	92.4
PointNet++ with MPFH	**93.3**

Table 3. **Comparison Results for PointNet++**. Metric is instance accuracy. Concatenating MPFH will result in the increase of instance accuracy by 2.7%.

Table 3 reports the detailed instance accuracy comparison results. With the help of MPFH, we can improve the instance accuracy of PointNet++ from **90.6%** to **93.3%** (+2.7%).

5.2.3 Comparison between PointNet++ with MPFH and RepSurf

Lastly, we conduct comparison between our proposed PointNet++ [15] with MPFH and RepSurf [17], which achieves the state-of-the-art result in point cloud data classification on ModelNet40 Dataset. In fact, both of us share the similarity that, we attempt to utilize geometry information via introducing more inputs with richer semantic meanings. The main difference is, RepSurf utilizes MLP to learn the umbrella geometry via some basic geometry quantities like normals and relative angles. However, in our proposed method, we use histogram for more robust feature representation. Table 4 lists the detailed comparison results.

Model	ModelNet40 Dataset	
	Instance Accuracy	Class Accuracy
PointNet++ with MPFH	93.3	**91.2**
RepSurf	**94.0**	91.1

Table 4. **Comparison Results between PointNet++ with MPFH and RepSurf**. Metric is instance accuracy and class accuracy. Our proposed method is able to achieve comparable results from RepSurf, especially in class accuracy.

Table 4 shows that, via utilizing MPFH directly, we are able to achieve comparable results with respect to RepSurf. Moreover, our proposed method performs even better in class accuracy. This is possibly due to the fact that, MPFH is more sensitive to local geometry information, which allows us to classify those difficult classes correctly in ModelNet40 Dataset.

6. Conclusion

In this work, we proposed a novel Modified Point Feature Histogram (MPFH) for point cloud classification task, which can effectively capture local curvature information. Moreover, in order to improve the performance of PointNet and PointNet++ via MPFH, we proposed a hierarchical approach, which allows network to generate better geometry feature representation from MLPs only. We demonstrated the effectiveness of our proposed method through experiments on ModelNet40 Dataset. It achieved comparable result with the state-of-the-art RepSurf model. In future work, we would like to conduct experiments on more benchmark datasets and examine more on how to capture geometry feature representation effectively.

References

[1] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010. 3

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 2

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2

[4] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *Computational Visual Media*, 7:187–199, 2021. 1, 2, 6

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[6] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 1

[7] Mingyang Jiang, Yiran Wu, Tianqi Zhao, Zelin Zhao, and Cewu Lu. Pointsift: A sift-like network module for 3d point cloud semantic segmentation. *arXiv preprint arXiv:1807.00652*, 2018. 1, 2, 6

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 8

[9] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31, 2018. 1, 2, 6

[10] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021. 2

[11] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110, 2004. 3

[12] Xu Ma, Can Qin, Haoxuan You, Haoxi Ran, and Yun Fu. Rethinking network design and local geometry in point cloud: A simple residual mlp framework. *arXiv preprint arXiv:2202.07123*, 2022. 1, 2

[13] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 922–928. IEEE, 2015. 2

[14] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 1, 2, 5, 6, 7, 8, 9

[15] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 1, 2, 5, 6, 7, 9

[16] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018. 2

[17] Haoxi Ran, Jun Liu, and Chengjie Wang. Surface representation for point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18942–18952, 2022. 2, 9

[18] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE international conference on robotics and automation*, pages 3212–3217. IEEE, 2009. 4

[19] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, and Michael Beetz. Persistent point feature histograms for 3d point clouds. In *Proc 10th Int Conf Intel Autonomous Syst (IAS-10), Baden-Baden, Germany*, pages 119–128, 2008. 3

[20] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019. 1

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2

[22] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 9621–9630, 2019. 1, 2, 6

[23] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 2, 7

[24] Jiancheng Yang, Qiang Zhang, Bingbing Ni, Linguo Li, Jinxian Liu, Mengdie Zhou, and Qi Tian. Modeling point clouds

with self-attention and gumbel subset sampling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3323–3332, 2019. 1, 2, 6

[25] Xiaopeng Zhang, Hongjun Li, Zhanglin Cheng, et al. Curvature estimation of 3d point cloud surfaces through the fitting of normal section curvatures. *Proceedings of ASIAGRAPH*, 2008:23–26, 2008. 3

[26] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16259–16268, 2021. 1, 2, 6