Point is a Vector: A Feature Representation in Point Analysis

Xin Deng[1], WengYu Zhang[1], Qing Ding[1], XinMing Zhang[1]
[1] University of Science and Technology of China

Abstract

The irregularity and disorder of point clouds bring many challenges to point cloud analysis. PointMLP suggests that geometric information is not the only critical point in point cloud analysis. It achieves promising results based on a simple multi-layer perception (MLP) structure with a geometric affine module. However, these MLP-like structures aggregate features only with fixed weights, while differences in the semantic information of different point features are ignored. Therefore, we propose a novel Point-Vector Representation of the point feature to improve feature aggregation by using inductive bias. The direction of the introduced vector representation can dynamically modulate the aggregation of two point features according to the semantic information. Based on this, we design a novel Point2Vector MLP architecture. Experiments show that it achieves state-of-the-art performance on the classification task of the ScanObjectNN dataset, with an 1% increase compared with the previous best method. It also achieves a competitive performance on the ShapeNetPart dataset. We hope our method can help people better understand the role of semantic information in point cloud analysis and lead to exploring more and better feature representations or other ways.

1 Introduction

The point cloud is a data structure generated by 3D scanning devices such as LiDAR and RGB-D sensors that scan objects in realistic scenes. Since point clouds are usually sparse, irregular, and disordered, it is very difficult to directly apply proven techniques from image processing to point cloud analysis. A classical approach attempts to convert irregular point clouds into regular voxels ([1],[2],[3],[4]) or render point clouds to Bird’s Eye View ([5],[6],[7]). However, this method produces additional memory overhead, and inevitably introduces quantification error that can obscure the nature properties of data.

To reduce the information loss, some researchers suggest processing irregular point clouds directly. PointNet [8] is a pioneering work in point-based methods, that applies multi-layer perception (MLP) at each point to extract features and then aggregates them via symmetric functions such as max-pooling. However, PointNet only considers global features, while local features are missing. PointNet++ [9] extracts hierarchical point set features by setting abstraction on the basis of PointNet. However it still handles each point in the local point set separately and does not extract local relationships. Locality is a priori knowledge of point clouds, and many recent works have attempted to explore local geometric information through convolution ([10],[11]), graph structure ([12],[13],[14]), and attention mechanisms ([15],[16],[17]). Some works ([18],[19],[20]), as shown in the Figure 1(a), use attention mechanisms to assign weights, and attempt to define the importance of these features for aggregation methods. Many recent works ([21],[14]) tend to design more dynamic kernels or anisotropic methods, as shown in Figure 1(b). However these well-designed modules usually represent excessive computational and memory access overhead.

Preprint. Under review.
Figure 1: Illustrate the differences between the various types of aggregation operations and our method. (a) Dynamic assignment of weights based on feature importance. (b) Generate unique kernels ($\hat{e}_i$ and different arrows) for each point feature. (c) Differently, our method generates a point-vector representation for each feature by semantic information, and then assigns weights to the summation.

PointMLP\textsuperscript{[22]} designs a simple MLP structure with a geometric affine module. Furthermore, in the image processing domain, the latest WaveMLP\textsuperscript{[23]} proposes a new form of feature representation, and achieves good results with a simple residual MLP structure. It proposes a wave-like token representation, and designs a novel token-mixing module. Inspired by this, we decide to abandon the complex network structure design and return to MLP’s simple design concept. PointMLP\textsuperscript{[22]} and our method can be counted as one type (see Figure 1\textsuperscript{(c)}) with a difference in feature processing. It performs geometric affine transformations on the features, and we propose a new feature representation to better aggregate semantic information. Previous works\textsuperscript{[24],\textsuperscript{25]} treat the values of features as scalars and then assign weights for aggregation. Previous works have also used the concept of vectors to extract features. Just as vector neurons\textsuperscript{[26]} directly treat the input coordinates as vector coordinates, the whole network performs the vector transformation. GeoCnn\textsuperscript{[27]} uses the vector direction of the original space to guide the fusion of features. However, we associate the direction of the vector to represent the semantic content. Our vector representation is based on the intermediate features and has no direct relationship with the original space.

In this paper, we propose a point analysis MLP architecture (called Point2Vector MLP), which implements vector-like representation and local feature aggregation. We represent the geometric information between points by a Point Feature Generator module rather than the complex geometric feature extractor. We explore the feature aggregation aspect of point cloud analysis. The contributions of this paper are summarized as follows:

- We introduce a new representation of the point feature – Point-Vector representation, which replaces each value of the point feature with a three-dimensional vector. This forces the network to focus on the impact of semantic information on feature aggregation.
- Based on the point-vector representation, we design a Point2Vector MLP network with the implementation of the local Point-Vector feature aggregation.
- We apply the proposed method to point cloud classification and part segmentation tasks, and quantitatively evaluate the results. Our Point2Vector MLP model achieves state-of-the-art classification performance on the real-world ScanObjectNN dataset, and outperforms related works by 1.0% accuracy. It achieves a competitive performance on the ShapeNetPart dataset.

2 Related Work

Local feature extraction. Many recent works\textsuperscript{[28],\textsuperscript{21],\textsuperscript{11]} proposes novel local feature extractors to efficiently learn local features. Inspired by 2D image processing, current methods are divided into three categories: convolution-based, graph-based and attention-based. PointCNN\textsuperscript{[10]} aligns point clouds by predicting the local point set transformation matrix. However, such an operation is not permutation invariant, which leads to sensitivity to the order of input points. Subsequent convolution-based methods began to move toward dynamic convolution kernels, e.g. PointConv\textsuperscript{[11]} uses MLP for density estimation to generate density-adaptive convolution kernels. KPConv\textsuperscript{[20]} considers the effect of coordinates of points in local regions on convolution kernels, and propose to learn location-based convolution kernels directly. The latest PAConv\textsuperscript{[21]} abandons the direct generation of convolutional kernels. The graph-based approach treats points as nodes of a graph and creates edges based on their
spatial and feature relationships. The DGCNN \cite{12} aggregates the nearest neighbor points in the feature space, then EdgeConv \cite{12} generates edge features to represent the relationship between a node and its neighboring nodes. CurveNet \cite{29} aggregates a series of arcs on the isomorphic map of the point cloud to complement the geometric information. The latest AdaptConv \cite{14} proposes adaptive graph convolution rather than simply assigning weights to individual neighboring points. Transformer, which has recently become more popular in the field of 2D images, also performs well on point clouds. PCT \cite{30} and PointTransformer \cite{16} has the advantage of exploring local geometric structures, reflecting that the attention mechanism is good at modeling the relationship between neighbors. However, those local feature extraction networks are more complex and demanding in terms of computational effort and space occupation.

**MLP-like architectures.** Point cloud analysis and the development of image processing techniques are closely related. Recently MLP structures \cite{25}, \cite{31} in the image processing domain have developed very rapidly, rivalling CNN structures and transformer structures. In the field of point cloud analysis, PointNet \cite{8} and PointNet++ \cite{9} pioneer point-based methods based on the MLP structure, and many works turn to convolution, graph and transformer. Now, it returns to the simplicity of MLP structure. PointMixer \cite{24} proposed a generic MLP-based point set operator for information sharing between 3D unstructured points. PointMLP \cite{22} proposes a geometric affine module that can be inserted into very simple ResMLP (Residual MLP) structure. The latest work in the field of image processing, WaveMLP \cite{23} treats tokens as waves, thus designs a very novel WaveMLP architecture by taking into account the influence of semantic information of the image patch. It dynamically adjusts token-mixing by superimposing those waves. Inspired by previous works but different from them, we treat each value of the point feature as a projection of a 3D spatial vector in 1D space. In this paper, we explore new representations of features as 3D vectors with modulus and angle properties, and use summation of vectors to implement local feature aggregation.

### 3 Method

![Figure 2: The basic block of the Point2Vector MLP model. Norm means normalization method. Channel MLP means MLP composed of Channel-FC. The details of each module are described in Section 3.2, Section 3.3.](image)

In this section, we discuss the proposed Point2Vector MLP in detail and compare it with other point-based models. Our overall structure follows the simplicity of the MLP-like structure, it is stacked by the basic blocks in the Figure 8 and samples the point cloud step by step. To enhance the performance of the deep network, we add an additional residual structure on the main path. For the sake of clarity, we revisit point-based methods and the idea of WaveMLP in Section 3.1. Then we show in detail the vector-related point feature aggregation module (VPFA) in Section 3.2. At last, in Section 3.3, we describe the Point Feature Generator. Note that the point features we mention in this paper are the intermediate features. The intermediate features contain the semantic information of a region of the original space. Since the vectors in the original space go through a complex neural network with nonlinear factors, we can no longer obtain the mapping between the intermediate vectors and the original vectors. Therefore, our approach is not directly related to the original space.

#### 3.1 Preliminary

Similar to PointNet++ \cite{9}, previous MLP-like work treats each value of the feature vector as a *scalar*, and aggregates features by linearly combining values on the channels as well as on the space. Their
aggregation operations are represented as follows:

\[
\text{Channel-FC} \left( x_j, W^c \right) = x_j W^c, j = 1, 2, \ldots, n,
\]

(1)

\[
\text{Token-FC} \left( X, W \right)_i = \sum_{k}^{N_i} W_{ik} \odot X_k, i = 1, 2, \ldots, n,
\]

(2)

where \( W^c \) is a \( c_{\text{in}} \times c_{\text{out}} \) learnable matrix, \( x_j \) is a \( c_{\text{in}} \) dimensional feature vector of point \( j \), \( W_{ik} \) denotes a \( c \)-dimensional learnable parameter, \( N_i \) denotes the neighbors of point \( i \), \( X_k \) is a \( c \)-dimensional feature vector, and \( \odot \) is the Hadamard product. In PointNet, \( W_{ik} \) is a one-hot vector, the entire equation represents max-pooling. In other cases, such as PointMix, \( W_{ik} \) means the dynamic weight related to other information such as location or distance. The simple weighted aggregation operation is the bottleneck of the representation ability of the MLP network. In image processing, WaveMLP\(^{[23]} \) gives a new perspective on features, and its representation can be abstracted as:

\[
\tilde{f}_j = |f_j| \odot e^{i\theta_j}, j = 1, 2, \ldots, n,
\]

(3)

where the amplitude \( |f_j| \) is a real-value feature representing the content of each token, \( i \) is the imaginary unit satisfying \( i^2 = -1 \), and \( \theta_j \) indicates the phase representing the effectiveness of semantic information. WaveMLP views the token as a wave, with both amplitude and phase, and the actual value of token features as the amplitude of this wave exhibited at some phase \( \theta \). The feature vector summation manifests itself as a superposition of waves as:

\[
\tilde{f}_k = \sum_{j \in M_k} W_1 \odot |f_j| \odot \cos \theta_j + i W_2 \odot |f_j| \odot \sin \theta_j
\]

(4)

where \( M_k \) is the set of image patches around patch \( k \). Finally, they project the wave onto the real part to obtain the final aggregated result. Similarly to WaveMLP\(^{[23]} \), on the point cloud we can explore the representation of point features, thus enhancing the representation of MLP structure. This shows that semantic information is critical in image processing. In the same way, we can explore the representation of semantic information in point cloud analysis.

### 3.2 Vector-Related Point Feature Aggregation

Note that when we talk about each channel of a vector representation, we are referring to a three-dimensional vector. We consider that the \( c \)-dimensional features can be projected to 3-dimensional space, then 3-dimensional vector is projected to 1-dimensional scalar. In previous works, the \( c \)-dimensional feature is often projected directly to 1-dimensional scalar and then aggregated. Instead, we propose an intermediate representation to aggregate features first, and then project them. There is no doubt that higher dimensional representations can also give better results, but the higher resource consumption limits their application in neural networks. So we choose a 3D vector as our intermediate vector, which is more expressive compared to a 1D scalar and has acceptable consumption. For each channel of our vector representation, the \( c \)-dimensional feature is projected into a 3-dimensional vector. To better aggregate the features of neighboring points and dynamically adjust the relationship between weights and features, we first explore the 3D vector representation of the value of features to better aggregate features. This section presents details of vector representation and feature aggregation. The details of our VPFA module are shown in the Figure\([\ldots]\)

**Point-Vector representation.** We talked about feature projection in Sec 3.2 to obtain a 3D vector by first projecting into 3D space, and then the 3D vector is projected into a 1D scalar value. We consider that each value of the result should be the projection of a three dimensional vector into one dimensional space. We assume that the basis vectors of our vector representation are orthogonal, so we can borrow the Cartesian coordinate system to represent the 3D vectors. The projection operation is formulated as follows:

\[
f^k_i = [x, y, z] W^k, (x, y, z) \in \mathbb{R}^3, k = 1, 2, \ldots, c,
\]

(5)

where \( (x, y, z) \) is the coordinate of our assumed three-dimensional vector, \( c \) is the number of channels, \( W^k \) denotes the learnable \( 3 \times 1 \) matrix, and \( f^k_i \) is the value of \( k \)-th channel of point \( i \). From there, in our model, a three-dimensional vector is obtained from the point feature. It is expressed in Figure\([\ldots]\)
Figure 3: The process of VPFA module. FC means Fully-Connect layer. As shown in Figure 9 and Vector triplet representation are identical. For each channel of the Point-Vector representation, it means \( c \)-dimensional feature is projected into a 3-dimensional vector.

Figure 4: Illustrate the point-vector expression. For the explanation of each attribute, see Section 3.2

We denote \( r \) as the modulus of the vector, \( \alpha \) as the angle between the projection of the vector in the \( x \)-\( y \) plane of the Cartesian coordinate system and the \( x \)-axis, and \( \beta \) as the angle between the vector and the \( z \)-axis. Another representation of the orthogonal basis of a vector is analyzed in detail in Section 4.2. The whole point feature is formulated as:

\[
\vec{r}_i = (r_i \circ \sin \beta_i \circ \cos \alpha_i, r_i \circ \sin \beta_i \circ \sin \alpha_i, r_i \circ \cos \beta_i), \quad i \in N,
\]

where \( \vec{r}_i \) is the vector representation of point \( i \), \( r_i \) is the \( c \)-dimensional modulus vector, \( \circ \) denotes the product by element, and \( \alpha_i \) and \( \beta_i \) are the \( c \)-dimensional angle vectors. Due to the properties of the \( \cos \) and \( \sin \) functions, we can guarantee that the basis vectors are orthogonal.

The angle between two vectors has a huge effect on the result of adding the vectors. When the two vectors are in the same direction, \(< \vec{r}_1, \vec{r}_2 > = 0 + i \times 2\pi\), the final result is the sum of their two modulus \( \|\vec{r}_1\| = \|\vec{r}_2\| \) as Figure 10(a), and Figure 10(b) shows the two vectors are reversed, \(< \vec{r}_1, \vec{r}_2 > = 0 + i \times \pi, i = \pm 2k + 1, k = 0, 1, 2, \ldots \), the final result is the difference between the two vector modulus \( \|\vec{r}_1\| = \|\vec{r}_2\| \). When the angle between the two vectors is otherwise like Figure 10(c), the enhancement or weakening of the two features depends on the angle between the two vectors when they are aggregated. For consistency, the final vector needs to be projected into a one-dimensional space.

**Modulus of vector.** We need to obtain the \( c \) modulus of 3-dimensional vectors from the original \( c \)-dimensional feature. If \( X, Y \in \mathbb{R}^c \), the transformation matrix can be formulated as follows:

\[
A = \frac{YX^T}{\|X\|^2},
\]
Figure 5: Illustrate the different cases of two vectors summation: $\vec{r} = \vec{r}_1 + \vec{r}_2$. The sum of two vectors is calculated by the parallelogram rule. The dashed lines of different colors represent the components of different vectors. (a) shows two vectors in the same direction. (b) shows two vectors in opposite directions. (c) shows the general case.

where $A$ satisfies $AX = Y$, $X$ denotes the $c$-dimensional feature vector and is not zero vectors, and $Y$ denotes the $c$-dimensional modulus of the vector representation. According to Equation[7] we can prove that the linear transformation matrix does exist. Therefore, we were able to approximate this transformation matrix by Channel-FC. Denoting $X = [X_1, X_2, \ldots, X_c]$ as the input of a module, we can obtain the vector modulus $|\vec{r}_i|$ based on the plain Channel-FC in Equation[1] i.e.,

$$ |\vec{r}_i| = \text{Channel-FC}(W_i, X), i = 1, 2, \ldots, N, $$

where $W_i$ is $c \times c$ transformation matrix, and $|\vec{r}_i|$ is the $c$-dimensional modulus vector.

Direction of vector. The direction represents the influence of semantic information on feature aggregation. We use the angles $\alpha$ and $\beta$ in Figure[3] to define the vector direction. The properties of the cos and sin functions ensure that the basis vectors are orthogonal. For the calculation of the angle, we consider two strategies. The first is the static angle generation strategy, which uses fixed parameters to represent the angle of vector. This also distinguishes between different types of points, but ignores the variability of the input point cloud. Another is dynamic estimation in which the angle of vector is dynamically related to the input features, i.e., $\alpha \cos \beta = \Theta(X, W^\theta)$, where $W^\theta$ as the learnable parameter. We have tentatively chosen Channel-FC for $\Theta(\cdot)$. To maintain the simplicity of MLP structure, we did not design a very complicated prediction function. We will explore the importance of the angle later on in Section[4.2].

Feature Aggregation. After we obtain the Point-Vector representation, we convert it to a triplet as in Equation[6] Then the vectors $\vec{r}'_j$ of the neighboring points of point $i$ are summed as following:

$$ r'_i = \text{Token-FC}(\vec{r}', W) \times_i $$

$$ = \sum_{j \in N_i} W^1_j \odot r_j \odot \sin \beta_j \odot \cos \alpha_j, \sum_{j \in N_i} W^2_j \odot r_j \odot \sin \beta_j \odot \sin \alpha_j, \sum_{j \in N_i} W^3_j \odot r_j \odot \cos \beta_j, $$

where $N_i$ is the set of neighbors of point $i$ to be aggregated, $W^1_j$ is the $c$-dimensional learnable weight matrix, and $r_j$ means $c$-dimensional vector that consists of the modulus. This formula represents the summation of each channel for each vector representation separately. The summation of vectors is the summation of the components of the triplet separately.

We can see that the vectors obtained are triplet representations. According to our assumptions, the vectors need to be projected into scalars. We generate the result by Channel-FC to project the vector into a one-dimensional space, i.e., result $= \vec{r}W$, where $W$ is a $3 \times 1$ matrix, and $\vec{r}$ we regard as a $c \times 3$ vector representation.

In our proposed Point2Vector MLP model, the values of each channel in the feature are replaced by three-dimensional vector. A representation is first obtained by using the above equations to obtain $r$ and $\alpha$ and $\beta$ respectively. Then the point features are aggregated using the weighted sum of this set of 3D vectors like Equation[9] Finally by projecting the three-dimensional vector to the one-dimensional space, we obtain the results of neighbor feature aggregation. The feature representation is then enhanced by a Channel-FC. We used a skip-connection to enhance the network robustness.
3.3 Point Feature Generator

Due to the influence of the sparse and irregular structure of the point cloud, our module has difficulty converging and does not achieve very good results with the original input feature. It is difficult for us to make the network fit the relationship between several coordinates. The influence of the large space occupation and uneven distribution of point clouds does pose a great challenge. To solve this problem, we use furthest sampling to select the centroids. This operation maximize the preservation of the point cloud in the overall structure while preserving more of the sparse regions, and mitigates the uneven distribution while reducing computational difficulty. Let \( f_{i,j} \in \mathbb{R}^{k \times c} \) be the grouped local neighbors of \( f_i \in \mathbb{R}^c \) containing k points, each neighbor point \( f_{i,j} \) is a \( c \)-dimensional vector. It can be formulated as:

\[
    f_i = \text{Maxpool} \left( \text{Channel-FC} \left( \text{Batchnorm} \left( (f_{i,j} - f_i), (f_i) \right) \right), j = 1, 2, \ldots, k, f_{i,j} \in \mathbb{R}^{k \times c} \right), \quad (10)
\]

where Batchnorm is a normalization method. By Batchnorm, we narrow the gap between the range of relative and absolute features. Thus the network converges more easily. To maintain the simple nature of the MLP structure, we used Channel-FC and Max-pooling to process, thus, retaining most of the features of the original point cloud while highlighting the salient features. The differences in the construction of features are shown in the supplementary materials.

4 Experiments

In this section, we evaluate our Point2Vector model on several benchmarks, and provide a detailed ablation experiment to quantitatively and qualitatively analyze the effectiveness of our model.

4.1 Shape Classification On ScanObjectNN

| Method      | mACC(%) | OA(%) |
|-------------|---------|-------|
| PointNet[8] | 63.4    | 68.2  |
| SpiderCNN[32]| 69.8    | 73.7  |
| PointNet++[9]| 75.4    | 77.9  |
| DGCNN[12]  | 73.6    | 78.1  |
| PointCNN[10]| 75.1    | 78.5  |
| DRNet[33]  | 78.0    | 80.3  |
| SimpleView[34]| -       | 80.5  |
| GBNet[35]  | 77.8    | 80.5  |
| PRA-Net[36]| 79.1    | 82.1  |
| MVTN[37]   | -       | 82.8  |
| Point-BERT[38]| -      | 83.1  |
| Point-MAE[39]| -      | 85.2  |
| PointMLP[22]| -      | -     |
| **Point2Vector MLP** (ours) | **84.9 ± 0.3** | **86.4 ± 0.3** |

Many 3D object classification methods report high performance on CAD model datasets such as the ModelNet40[40] benchmark. Meanwhile other methods (PACon[21], PointMLP[22] etc.) mention that the accuracy of ModelNet40 fluctuates particularly. Therefore, we conduct experiments on the ScanObjectNN[41] benchmark and report the results.

The dataset is a scanned dataset of real-world indoor scenes and contains about 15,000 objects that are divided into 15 categories and cover 2902 unique objects. Since real-world scanned objects are usually mixed with the background or incomplete due to occlusion, this poses a huge challenge for point cloud analysis. We use the most difficult variant \( PB_{T50\_RS} \) in our experiments. We train 200 epochs using SGD optimizer with a batch size of 32. For the sake of fairness, our parameters are set exactly as the baseline PointMLP. To increase reproducibility, we fix the random seed to be \( I \). For comparison we still use the results given in the paper as a baseline. To accurately illustrate the effectiveness of our model, we train and test 4 rounds and report the mean ± standard deviation in
According to the experiments, our method has a significant improvement on both class mean accuracy (mAcc) and overall accuracy (OA) metrics compared to all other methods. For example, we outperform PointMLP by 0.6% mAcc and 1.0% OA. We achieve higher accuracy, although we are approximately 1/3 slower than baseline in inference. Moreover, our method achieves the smallest gap between class mean accuracy and overall accuracy. This phenomenon indicates that our method is more robust and not biased towards specific classes in comparison.

4.2 Ablation Studies

To better demonstrate our approach, we design several sets of ablation experiments to demonstrate the effectiveness of the components of our model. The experiments are conducted on the ScanObjectNN dataset. If the difference between the comparison results is too large, we do not use mean ± standard deviation to present the results of our experiments. For a detailed ablation experimental setup and more ablation studies, refer to the supplementary material.

Table 2: The difference between different vector representations. For more information on each representation, see 4.2

| Representation               | mAcc (%) | OA (%)  |
|-----------------------------|----------|---------|
| coordinate projection       | 83.4 ± 0.4 | 85.1 ± 0.3 |
| orthogonal coordinate      | 83.4 ± 0.6 | 85.0 ± 0.6 |
| planar projection           | 84.9 ± 0.3 | 86.4 ± 0.3 |

The representation of vector. In section 3.2, we give the point-vector representation of the triplet form. We chose \((r \sin \beta \cos \alpha, r \sin \beta \sin \alpha, r \cos \beta)\) instead of \((x, y, z)\) or \((r \cos \alpha, r \cos \beta, r \cos \gamma)\). We consider the former as the planar projection representation and the latter two as the orthogonal coordinate representation, and the coordinate projection representation, respectively. When the latter two representations are taken, their aggregation operations are shown in the supplementary material. The comparison results of the representations are shown in the Table 9. It can be seen that the planar projection representation performs far better (≈ 1%) than the other two representations. And the deviation value of the latter two is larger. The reason is that the latter two representations have no constraints on the components of the triplet, so the components learned by the network may not be orthogonal.

Table 3: Effectiveness of model components. Our modules are defined in Section 3 and new modules are defined in Section 4.2

| knn-MAX | Channel-FC | PFG | LocalGrouper(PointMLP) | mAcc (%) | OA (%)  |
|---------|-------------|-----|------------------------|----------|---------|
| X       | ✓           | X   | ✓                      | 83.9 ± 0.5 | 85.4 ± 0.3 |
| ✓       | X           | X   | ✓                      | 83.6 ± 0.1 | 85.6 ± 0.1 |
| ✓       | ✓           | X   | ✓                      | 84.4 ± 0.3 | 86.1 ± 0.3 |

| VPFA | k-MLP | knn-MAX | PFG | mAcc (%) | OA (%)  |
|------|-------|---------|-----|----------|---------|
| X    | ✓     | X       | ✓   | 84.4 ± 0.3 | 86.1 ± 0.3 |
| X    | ✓     | X       | ✓   | 84.2      | 86.2    |
| ✓    | ✓     | X       | ✓   | 85.9 ± 0.3 | 86.4 ± 0.3 |

Effectiveness of model components. The knn-MAX module consists of kNN+Max pooling, and k-MLP module consists of kNN+MLP aggregation. The LocalGrouper(PointMLP) module consists of the original LocalGrouper, MLP and Max pooling in PointMLP. The original PointMLP consists of LocalGrouper(PointMLP) and Channel-FC. As shown in the table, adding the k-MAX module
directly to pointMLP results in a slight performance improvement. However it obtains a higher (OA-mAcc) score, which means a greater bias toward specific categories. The PFG module improved 0.5% compared to the LocalGrouper(pointMLP) module. We use a simple BN in PFG for dequantization, which reduces the difference between the values of relative features and absolute features. It shows that our PFG module is better than the LocalGrouper(pointMLP) module.

We use k-MAX as the baseline for dynamic weight aggregation and k-MLP as the baseline for static weight aggregation. The k-MAX and k-MLP modules have a small difference in OA score, but the latter module has a lower mAcc score. This illustrates that the dynamic weight aggregation approach is less biased toward specific categories. The VPFA module has a slight improvement in OA score compared to the k-MAX module. However, our module obtained a higher mAcc score and a lower (OA-mAcc) score. This reflects the robustness of our approach relative to the dynamic aggregation baseline, which means there is little bias toward specific categories.

| module                  | mACC(%) | OA(%) |
|-------------------------|---------|-------|
| No angle                | 76.3    | 79.2  |
| Identity                | 82.6    | 84.5  |
| Fixed Parameter         | 83.4    | 85.2  |
| Dynamic(Group Conv)     | 84.8 ± 0.4 | 86.1 ± 0.4 |
| Dynamic(Channel-FC)     | 84.9 ± 0.3 | 86.4 ± 0.3 |

The effectiveness of angle. We give the meaning of 2 angles and explain their necessity in the previous section. The difference with the angle estimation can be seen in the table. No angle means that our model without VPFA. Even the ability is weaker than other models. It shows that it is unreliable when the fixed parameters is used to estimate the direction of the vector corresponding to each point feature. Different point features contain different semantic information, and our estimation method of the angle should be dynamic. As the table shows, the difference between the effect of Group Conv and Channel-FC is relatively small. We set the number of channels to groups when we use Group Conv, which indicates that the angle is strongly correlated with its own channel instead of other channels.

4.3 Part Segmentation.

We extend the proposed presentation to the part segmentation task, and evaluate it on the ShapeNetPart benchmark. The ShapeNetPart dataset contains 16881 objects divided into 16 categories. There are 50 categories of parts labels in total. Each shape contains 2 ~ 6 part categories. We use the PointNet++ setting which randomly selects 2048 points as input points. The results are displayed in Table. Some visualization results are shown in Supplementary material.

We directly replace the curve_generation and curve_aggregation structures in CurveNet with the Point Feature Mixing module in Fig. We use the same training strategy and hyperparameters as CurveNet. Since we did not use the vote strategy, we only compare the result(w/o vote) with CurveNet. Our model gets a competitive performance.

| Methods          | Input | #point | mIoU↑ |
|------------------|-------|--------|-------|
| PointNet [8]     | xyz   | 2048   | 83.7  |
| DGCNN [12]       | xyz   | 2048   | 85.1  |
| PointCNN [10] *  | xyz   | 2048   | 86.1  |
| PointASNL [15]   | xyz   | 2048   | 86.1  |
| RS-CNN [42] *    | xyz   | 2048   | 86.2  |
| PACConv [21]     | xyz   | 2048   | 86.0  |
| PACConv [21] *   | xyz   | 2048   | 86.1  |
| PCT [30] *       | xyz   | 2048   | 86.4  |
| PointTransformer [16] | xyz | 2048 | 86.6 |
| PointNet++ [17]  | xyz, nr | 2048 | 85.1 |
| SO-Net [43]      | xyz, nr | 1024 | 84.6 |
| CurveNet w/o curves [29] | xyz | 2048 | 85.9 |
| CurveNet [29]    | xyz   | 2048   | 86.6  |
| CurveNet [29] *  | xyz   | 2048   | 86.8  |
| Point2Vector MLP(Ours) | xyz | 2048 | 86.6 |
Please see the appendix for a detailed description of the segmentation model. Our model is able to achieve similar effects as the curves module of CurveNet.

5 Conclusion

In this paper, we propose a novel Point-Vector representation, which focuses on the impact of semantic information on feature aggregation. In this model, semantic information is introduced to improve the feature aggregation. The modulus is related to actual point feature, and the angle represents the influence of semantic information. Based on this representation, a MLP-like architecture for point cloud analysis is designed. The Experiments show that our Point-Vector feature representation improves greatly on the object classification task and is well adapted to global feature extraction. At the same time, our proposed method can achieve competitive performance on the part segmentation task. But our method still has the limitation that we obtain the same basis vector for the vector representation. We believe that for dense prediction tasks, the basis vectors of the local region should differ from the global basis vectors. Our work can inspire people to pay attention to improve the quality of feature expression.

Acknowledgments and Disclosure of Funding

Acknowledgment. This work was supported in part by the National Key Research and Development Program of China under Grant 2020YFB2103803 and by the National Natural Science Foundation of China under Grant 62072425 and Grant 61672485.

References

[1] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 4490 – 4499, Salt Lake City, UT, United states, 2018.

[2] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 10526 – 10535, Virtual, Online, United states, 2020.

[3] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, volume 2019-June, pages 12689 – 12697, Long Beach, CA, United states, 2019.

[4] Qingdong He, Zhengming Wang, Hao Zeng, Yi Zeng, and Yijun Liu. Syga-net: Sparse voxel-graph attention network for 3d object detection from point clouds. arXiv, 2020.

[5] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In Proc. ICCV, 2015.

[6] Yan Wang, Wanxia Zhong, Hang Su, Fujiang Zheng, Yiran Pang, Hongchuan Wen, and Kun Cai. An improved mvccn for 3d shape recognition. In Proceedings of 2021 IEEE International Conference on Emergency Science and Information Technology, ICESIT 2021, pages 469 – 472, Chongqing, China, 2021.

[7] Jongyoun Noh, Sanghoon Lee, and Bumsub Ham. Hvpr: Hybrid voxel-point representation for single-stage 3d object detection. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 14600 – 14609, Virtual, Online, United states, 2021.

[8] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, volume 2017-January, pages 77 – 85, Honolulu, HI, United states, 2017.

[9] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017.

[10] Yangyan Li, Rui Bu, Mingchoa Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In Advances in Neural Information Processing Systems, volume 2018-December, pages 820 – 830, Montreal, QC, Canada, 2018.
[11] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, volume 2019-June, pages 9613 – 9622, Long Beach, CA, United states, 2019.

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

[13] Qiangeng Xu, Xudong Sun, Cho-Ying Wu, Panqu Wang, and Ulrich Neumann. Grid-gcn for fast and scalable point cloud learning. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 5660 – 5669, Virtual, Online, United states, 2020.

[14] Haoran Zhou, Yidan Feng, Mingsheng Fang, Mingqiang Wei, Jing Qin, and Tong Lu. Adaptive graph convolution for point cloud analysis. In Proceedings of the IEEE International Conference on Computer Vision, pages 4945 – 4954, Virtual, Online, Canada, 2021.

[15] Xu Yan, Chaoda Zheng, Zhen Li, Sheng Wang, and Shuguang Cui. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 5588 – 5597, Virtual, Online, United states, 2020.

[16] Nico Engel, Vasileios Belagiannis, and Klaus Dietmayer. Point transformer. IEEE Access, 9:134826 – 134840, 2021.

[17] Xumin Yu, Yongming Rao, Ziyi Wang, Zuyan Liu, Jiwen Lu, and Jie Zhou. Pointinr: Diverse point cloud completion with geometry-aware transformers. In Proceedings of the IEEE International Conference on Computer Vision, pages 12478 – 12487, Virtual, Online, Canada, 2021.

[18] Lei Wang, Yuchun Huang, Yaolin Hou, Shennan Zhang, and Jie Shan. Graph attention convolution for point cloud semantic segmentation. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, volume 2019-June, pages 10288 – 10297, Long Beach, CA, United states, 2019.

[19] Matan Atzmon, Haggai Maron, and Yaron Lipman. Point convolutional neural networks by extension operators. ACM Transactions on Graphics, 37(4), 2018.

[20] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, Francois Goulette, and Leonidas Guibas. Kpconv: Flexible and deformable convolution for point clouds. In Proceedings of the IEEE International Conference on Computer Vision, volume 2019-October, pages 6410 – 6419, Seoul, Korea, Republic of, 2019.

[21] Mutian Xu, Runyu Ding, Hengshuang Zhao, and Xiaojuan Qi. Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 3172 – 3181, Virtual, Online, United states, 2021.

[22] 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, 2022.

[23] Yehui Tang, Kai Han, Jianyuan Guo, Chang Xu, Yanxi Li, Chao Xu, and Yunhe Wang. An image patch is a wave: Phase-aware vision mlp. arXiv preprint arXiv:2111.12294, 2021.

[24] Jaesung Choe, Chunghyun Park, Francois Rameau, Jaesik Park, and In So Kweon. Pointmixer: Mlp-mixer for point cloud understanding. arXiv, 2021.

[25] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. Advances in Neural Information Processing Systems, 34, 2021.

[26] Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas J. Guibas. Vector neurons: A general framework for so(3)-equivariant networks. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pages 12200–12209, October 2021.

[27] Yuedong Chen, Guoxian Song, Zhiwen Shao, Jianfei Cai, Tat-Jen Cham, and Jianmin Zheng. Geoconv: Geodesic guided convolution for facial action unit recognition. Pattern Recognition, 122, 2022.

[28] Siqi Fan, Qiulei Dong, Fenghua Zhu, Yisheng Lv, Peijun Ye, and Fei-Yue Wang. Scf-net: Learning spatial contextual features for large-scale point cloud segmentation. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 14499 – 14508, Virtual, Online, United states, 2021.

[29] Tiange Xiang, Chaoyi Zhang, Yang Song, Jianhui Yu, and Weidong Cai. Walk in the cloud: Learning curves for point clouds shape analysis. In Proceedings of the IEEE International Conference on Computer Vision, pages 895 – 904, Virtual, Online, Canada, 2021.

[30] 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(2):187 – 199, 2021.
Shoufa Chen, Enze Xie, Chongjian GE, Runjian Chen, Ding Liang, and Ping Luo. CycleMLP: A MLP-like architecture for dense prediction. In *International Conference on Learning Representations*, 2022.

Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11212 LNCS, pages 90 – 105, Munich, Germany, 2018.

Shi Qiu, Saeed Anwar, and Nick Barnes. Dense-resolution network for point cloud classification and segmentation. In *Proceedings - 2021 IEEE Winter Conference on Applications of Computer Vision, WACV 2021*, pages 3812 – 3821, Virtual, Online, United states, 2021.

Ankit Goyal, Hei Law, Bowei Liu, Alejandro Newell, and Jia Deng. Revisiting point cloud shape classification with a simple and effective baseline. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 3809–3820. PMLR, 2021.

Shi Qiu, Saeed Anwar, and Nick Barnes. Geometric back-projection network for point cloud classification. *IEEE Transactions on Multimedia*, 24:1943 – 1955, 2022.

Silin Cheng, Xiwu Chen, Xinwei He, Zhe Liu, and Xiang Bai. Pra-net: Point relation-aware network for 3d point cloud analysis. *IEEE Transactions on Image Processing*, 30:4436 – 4448, 2021.

Abdullah Hamdi, Silvio Giancola, and Bernard Ghanem. Mvtn: Multi-view transformation network for 3d shape recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1 – 11, Virtual, Online, Canada, 2021.

Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. arXiv, 2021.

Yatian Pang, Wenxiao Wang, Francis E.H. Tay, Wei Liu, Yonghong Tian, and Li Yuan. Masked autoencoders for point cloud self-supervised learning. arXiv, 2022.

Zhirong Wu, Shurang Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xianchao Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 07-12-June-2015, pages 1912 – 1920, Boston, MA, United states, 2015.

Li Yi, Vladimir G. Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics*, 35(6), 2016.

Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2019-June, pages 8887 – 8896, Long Beach, CA, United states, 2019.

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

Li Yi, Vladimir G. Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics*, 35(6), 2016.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. volume 32, Vancouver, BC, Canada, 2019.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. Toulon, France, 2017.

### Appendix

#### A.1 Summation of vectors

When aggregating two point features, the angle between them regulates the aggregation method. Now we assume that two vectors \( \vec{r}_1 \) and \( \vec{r}_2 \) are added to obtain the result \( \vec{r} = \vec{r}_1 + \vec{r}_2 \) (for simplicity, we set the weight of the sum of the two vectors to 1), several intuitive situations are shown in the Figure [10]. The modulus of \( \vec{r} \) in the Euclid Space and the angle between the two vectors can be calculated as following:

\[
|\vec{r}| = \sqrt{|\vec{r}_1|^2 + |\vec{r}_2|^2 + 2|\vec{r}_1| \cdot |\vec{r}_2| \cos \theta},
\]

where \( \theta \) is the angle between \( \vec{r}_1 \) and \( \vec{r}_2 \).
\[
\langle \vec{r}_1, \vec{r}_2 \rangle = \arccos \left( \frac{\vec{r}_1 \cdot \vec{r}_2}{|\vec{r}_1| \ast |\vec{r}_2|} \right),
\]

where \( \langle \vec{r}_1, \vec{r}_2 \rangle \) is the angle between the vector \( \vec{r}_1 \) and the vector \( \vec{r}_2 \), and \( \arccos (\cdot) \) denotes the inverse of the cos function. It can be found that the summation of vectors is strongly influenced by the direction. And the angle of a vector can easily represent the relationship between two vectors. Whether the two vectors promote or inhibit each other depends on the angle of the vectors.

### B The Wave Representation of 2D Image Patch

In the image processing, WaveMLP\[23\] gives a new perspective on features, its representation can be abstracted as:

\[
\tilde{f}_j = |f_j| \odot e^{i\theta_j}, \quad j = 1, 2, \ldots, n,
\]

where the amplitude \( |f_j| \) is a real-value feature representing the content of each token, \( i \) is the imaginary unit satisfying \( i^2 = -1 \), \( \theta_j \) indicates the phase representing the effectiveness of semantic information. WaveMLP views the token as a wave, with both amplitude and phase, and the actual value of token features as the amplitude of this wave exhibited at some phase \( \theta \). The feature vector summation manifests itself as a superposition of waves as:

\[
\tilde{f}_k = \sum_{j \in M_k} W_1 \odot |f_j| \odot \cos \theta_j + iW_k \odot |f_j| \odot \sin \theta_j
\]

where \( M_k \) is the set of image patches around patch \( k \). Finally, they project the wave onto the real part to obtain the final aggregated result.

### C Others Point-Vector Representations

We give two other representations of the vectors in Figure 6. The coordinate projection representation is shown in Figure 6(a), and the orthogonal coordinate representation is shown in Figure 6(b). The former can be formulated as:

\[
\vec{r}_i = (r_i \odot \cos \alpha_i, r_i \odot \cos \beta_i, r_i \odot \cos \gamma_i), \quad i \in \mathbb{N},
\]

where \( r_i \) is the modulus of vector, \( \alpha, \beta, \gamma \) denote the angle of the vector with the x-axis and y-axis and z-axis respectively. The vector summation form under this representation can be formulated as:

\[
\vec{r}_i = \sum_{j \in M_i} W_j \odot \vec{r}_j,
\]
\[ \vec{r}_i = \left( \sum_{j \in M_i} W_j^1 \odot r_j \odot \cos \alpha_j, \sum_{j \in M_i} W_j^2 \odot r_j \odot \cos \beta_j, \sum_{j \in M_i} W_j^3 \odot r_j \odot \cos \gamma_j \right), \] (17)

\[ f_{ik} = W_k \cdot \vec{r}_{ik}, k = 1, 2, \ldots, c, \] (18)

where \( M_i \) is the neighbors of point \( i \), \( W_j \) is the learnable matrix, \( W_k \) is a 1 \( \times \) 3 vector, \( r_{ik} \) is the \( k \)-th channel of vector \( \vec{r}_i \). Then the last representation can be formulated as:

\[ \vec{r}_i = (x_i, y_i, z_i), i \in N, \] (19)

where \( x_i, y_i, z_i \) are the xyz coordinates of the vector in the Cartesian coordinate system. The form of vector summation differs in the case of using coordinates to represent vectors. We formulate it as following:

\[ \vec{r}_i = (\sum_{j \in M_i} W_j^1 \odot x_j, \sum_{j \in M_i} W_j^2 \odot y_j, \sum_{j \in M_i} W_j^3 \odot z_j), \] (20)

\[ f_{ik} = w_k \cdot \sqrt{x_{ik}^2 + y_{ik}^2 + z_{ik}^2}, k = 1, 2, \ldots, c, \] (21)

where \( w_k \) is a learnable parameter. We use the coordinates to estimate the vector modulus length and then get the projection as the final value.

### D Network Architecture

#### D.1 Overall Network

![Figure 7: Structure of classification and segmentation tasks.](image)

The figure above shows the overall structure of our network, the classification task is directly stacked Down-block. The part segmentation task takes the strategy of baseline, using point feature propagation to upsample, then propagating the semantic features back to the original point cloud, obtaining the feature vector of each point, adding class information and global pooling features, and finally for each point classification to obtain the part prediction. In order to demonstrate the usefulness of our module, we have essentially kept it consistent with the baseline.

#### D.2 Point Feature Generator

The structure of PFG is briefly mentioned in previous section and the formula is given. The above figure shows the input feature generation process. PointMLP normalizes the features for all
channels and all knn groups of the feature map. To distinguish from it, we take the intuitive BatchNorm to normalize the feature map. This means that we only normalize all local regions for each channel. We have realized that the maxpooling operation loses information in non-extreme regions, which has an impact on the overall semantic prediction, and we will subsequently extend the Point-Vector representation to the downsampling part to create an adaptive pooling module.

E Experiments

Our experiments are conducted on two v100 gpus. We implement the Point2Vector MLP in PyTorch[45]. We use the SGD optimizer with momentum and weight decay set to 0.9 and 0.0001, respectively. For classification on ScanObjectNN[41], we train 200 epochs with initial learning rate 0.01, using CosineAnnealing[46] scheduler with one cycle. For part segmentation task on ShapeNetPart[44] benchmark, we train 350 epochs with Adam optimizer, the initial rate is 0.003, using CosineAnnealing scheduler with one cycle.

E.1 More Ablation Studies

| VPFA | Channel MLP | PFG(Layernorm) | PFG(Batchnorm) | mAcc(%) | OA(%) |
|------|-------------|---------------|---------------|--------|-------|
| X    |            |               |               | 76.3   | 79.2  |
| ✓    | ✓           |               |               | 82.6   | 84.8  |
| ✓    | ✓           | ✓             |               | 81.7   | 84.0  |
| ✓    | ✓           | ✓             | ✓             | 84.0 ± 0.4 | 85.7 ± 0.3 |
| ✓    | ✓           |               | ✓             | 84.9 ± 0.3 | 86.4 ± 0.3 |

Effectiveness of model components. The table reports the classification accuracy of the model on ScanObjectNN dataset after removing any of the independent components. PFG(Layernorm) means Point Feature Generator with Layernorm. Not using PFG means directly splicing features without using normalization, more details can be found at Section 3.3. We find that the difference brought by using different normalization methods is not particularly significant when constructing the input features. Normalization can reduce features of different magnitudes to the same range without changing their properties, thus promote network converge speed. The difference between the two accuracies (OA − mAcc) changes 2% to 1%, shows that normalization also enhances the stability of the network. After removing the VPFA module, the network performance drop a lot (≈ 8% mAcc and 7% OA). It shows that this representation can guide the aggregation of neighboring point features well. Channel MLP increases network depth and enhances expressiveness.

Part segment network. We directly replace the curve_generation and curve_aggregation structures in Curvenet with the Point Feature Mixing module in Fig.8. As the table[7] shows, Point2Vector_CurveNet is 0.7% mIOU higher to Point2Vector_PointMLP. This indicates that there
Table 7: Part segment network analysis. '_PointMLP' denotes it’s based on PointMLP[22]. '_CurveNet' denotes it’s based on CurveNet[29].

| Method                     | Ins. mIOU(%) |
|----------------------------|--------------|
| CurveNet w/o curves        | 85.9         |
| Point2Vector_POINTMLP      | 85.9         |
| PointMLP                   | 86.1         |
| CurveNet w/o vote          | 86.6         |
| Point2Vector_CURVENET      | 86.6         |

are some problems with other modules that affect performance. Point2Vector_CurveNet is 0.7% higher than CurveNet(w/o curves), it shows that our module can have a similar effect as CurveNet’s key module. By comparing Curvenet’s other modules with our Point2Vector_POINTMLP network, the following factors can be found to affect the performance of our Point2Vector_POINTMLP network:

1. Inspired by PointMLP, we originally use location information by directly splicing location and features. Then, it is sent to the MLP for processing. We now realize that there is a gap between the distribution of positions and features, which affects the performance. Curvenet uses the LPFA module to specifically perform the fusion of positions and features.

2. The Curvenet upsampling process also fuses the location information. In contrast, our Point2Vector_POINTMLP upsampling is simply a linear interpolation and MLP fusion of features.

We also analyze the impact of our improvements to point feature mixing:

1. In vector aggregation, the vector representation of point j in region 1 is the same as that in region 2, which is undesirable in the part segment task that requires fine local information. Therefore, we use \( \text{vector}_j - \text{vector}_i \) now in aggregation. \( \text{vector}_j - \text{vector}_i \) means the vector representation of point j in the i-th local region. This changes the above problem to some extent.

2. Before vector aggregation, we introduce the relative position vectors of points j and i and linearly transform them into the coordinate system of the feature. Finally, the results are summed with feature vector representation.

| Model          | mAcc\(\%\) | OA\(\%\) |
|----------------|------------|----------|
| Point2Vector_L | 84.2       | 85.8     |
| Point2Vector_P | 84.6       | 85.9     |
| Point2Vector_M | 84.9±0.3   | 86.4±0.3 |
| Point2Vector_XL| 84.4       | 85.4     |

Table 8: Effect of different number of VPFA modules

The number of VPFA blocks. Point2Vector_L means the numbers of VPFA blocks in 4 stages are \([2, 2, 2, 2]\) respectively. Point2Vector_P means the numbers of VPFA blocks in 4 stages are \([4, 2, 2, 2]\) respectively. Point2Vector_M means the numbers of VPFA blocks in 4 stages are \([2, 4, 6, 3]\) respectively. The result in ScanObjectNN benchmark is shown in Table 8. Our modules can be very effective even in small numbers. We note that stacking more modules in the previous layers brings a certain boost, while the mAcc can reach a higher level. It can be seen that our module gives better guidance in the early stage of feature extraction. Using more modules in the later stages does not lead to a boost and generates overfitting.

Skip connection for VPFA. The network architecture is shown in Figure 9, we explore the weight for skip connection and main path. We denote MLP result as \( a \), then the weight can be calculated as \( \alpha, \beta = \text{softmax}(a/k) \), where k means the scale factor. As shown in Table 9, even without the skip connection structure, our module achieves a good result, which means that our module is very effective. We find that the weight in each channel results higher Acc, but it causes larger variance. So we try to get the weight in batch dimension, it shows great potential. We find that the scale factor variation leads to better results, it’s become more generalized and more stable. However, the
results without weighting performed normally. We conclude that weighting in the channel dimension achieves better results but is not stable enough and tends to be specific to certain categories, while weighting in the batch dimension achieves better generalization.

| method              | mAcc(%) | OA(%) |
|---------------------|---------|-------|
| No weight           | 84.3    | 85.9  |
| No residual         | 84.6 ±0.4 | 85.8 ±0.2 |
| Batch weight(k = 1) | 84.2    | 85.6  |
| Batch weight(k = 30)| 85.15 ±0.05 | 86.3 ±0.1 |
| Channel weight      | 84.9 ±0.3 | 86.4 ±0.3 |

Table 9: Different summation strategies.

E.2 Accuracy and speed comparison

We introduce a new expression on top of the baseline, and the MLP layers stacked to construct this expression will affect the inference speed. The comparison is shown in table 10, we are slow by about 100 items/s, but get 1% acc elevation. Because pointmlp reasoning is much faster than other methods, we are already ahead of most of others methods.

| Method               | mAcc(%)   | OA(%)   | inference speed(items/s) |
|----------------------|-----------|---------|--------------------------|
| PointMLP(baseline)   | 83.9 ±0.5 | 85.4 ±0.3 | 230.5                    |
| Point2Vector(Ours)   | 84.9 ±0.3 | 86.4 ±0.3 | 131                      |

Table 10: Accuracy and Inference speed.

E.3 Visualization

Feature vector comparison. In order to compare the feature aggregation effect, we take 20 samples of each class and downscaled the feature vectors from the final maxpooling to 2 dimensions using PCA, and then displayed them in the figure 10. The horizontal and vertical coordinates are the values output by PCA, the red, black, and yellow colors represent the labels of the samples respectively. The figure 10(a) shows the feature of PointMLP, the figure 10(b) shows the result of ours. As shown in the figure, the result of baseline has many outliers, and the outliers are scattered far away. In contrast, our results have outliers but are closer to the center, and our different categories are farther apart from the center. The result in comparison is that ours are more easily separated and our network works better for semantic information aggregation.

PartSegment visualization. As shown in figure 11, our method does not perform very well for the junction of the part inlay. We focused only on semantic information, while positional and geometric structure information is crucial for the prediction of part junctions. We have recognized the disadvantage of our approach in dense prediction and tried to embed geometric and positional information into our representation. And we will subsequently try to generalize our feature representation in other areas such as semantic segmentation.
Figure 10: Feature vector comparison. (a) is the visualization of baseline. (b) is the result of ours.

Figure 11: Part segmentation results on ShapeNetPart. Top line is ground truth and bottom line is our prediction.
E.4 Limitations

Even if we use $\text{vector}_j - \text{vector}_i$ to obtain the vector representation of $j$ in the local region $i$. Note that such subtraction does not change the basis vectors. The basis vectors of points in that local region remain the same as the basis vectors of points in the global region. I believe that local regions should have a separate set of basis vectors, which will be explored in our next work.

We did not analyze the effect of position information on the representation of the vectors of features and used the most intuitive vector linear transformation to obtain position information. However, I think this does not take into account the gap between the position vector and the feature vector, which will be explored in subsequent work.