Geometric Feedback Network for Point Cloud Classification

Shi Qiu¹,², Saeed Anwar¹,² and Nick Barnes¹
¹Australian National University  ²Data61 - CSIRO
{shi.qiu, saeed.anwar}@data61.csiro.au, nick.barnes@anu.edu.au

Abstract

As the basic task of point cloud learning, classification is fundamental but always challenging. To address some unsolved problems of existing methods, we propose a network designed as a feedback mechanism, a procedure allowing the modification of the output via as a response to the output, to comprehensively capture the local features of 3D point clouds. Besides, we also enrich the explicit and implicit geometric information of point clouds in low-level 3D space and high-level feature space, respectively. By applying an attention module based on channel affinity, that focuses on distinct channels, the learned feature map of our network can effectively avoid redundancy. The performances on synthetic and real-world datasets demonstrate the superiority and applicability of our network. Comparing with other state-of-the-art methods, our approach balances accuracy and efficiency.

1. Introduction

Point clouds are one of the fundamental representations of 3D data, and widely used for both academia and industry because of the development of 3D sensing technology and relevant applications. Generally, 3D point clouds can be collected by 3D scanners [4] utilizing physical touch or non-contact measurements e.g. light, sound, LiDAR etc. Particularly, LiDAR scanners [9] are in service in many areas including agriculture, biology, and robotics, etc. Due to its tremendous contributions, point cloud analysis attracts much interest for further investigation.

Previously, traditional algorithms [30, 22, 29, 38] that operate on 3D data incorporated estimated geometric information and reconstructed models. With the help of deep learning, recent works on 3D data focus on data-driven approaches via Convolutional Neural Networks (CNNs). Classical works can be generally categorized as: multi-view images with 2D CNNs (e.g. MVCNN [32]), volumetric/mesh data with 3D CNNs (e.g. VoxNet [20]), 3D point cloud with multi-layer perceptrons (MLP) (e.g. [26]), etc.

Currently, many works are exploring different methods to improve the processing of 3D point clouds, but there remains some unsolved issues: (1) How can we force the network to automatically learn a better representation from the abstract high-level embedding space? (2) How do we refine the output features in order to focus on the crucial information? (3) Besides the regular features, can we learn from more geometric clues for a comprehensive analysis?

To investigate the possible answers to the above concerns, here we present a novel attentional feedback structure for point cloud learning via the incorporation of geometric context. As supported by substantial biological evidence, feedback mechanisms are very helpful for visual tasks, because feedback paths in our brain especially in the visual cortex consist of more neurons than the forward paths. Besides, feedback mechanisms also have been successfully utilized for stable and responsive systems in industry. Some early works managed to involve this mechanism in CNNs to deal with computer vision problems like 2D visual segmentation [5] or 3D hand pose estimation [24] etc. However, for point clouds, PU-GAN [13] to some extent used a feedback-like unit but for feature expansion inside its generator module. Our primary motivation to investigate feedback is to automatically refine the output feature map by comparing the difference between the input and the corresponding feedback signal. Using this approach, we enable the network to generate a better learning output.

Regarding high-level feature presentation for 3D point
clouds, another widely applied mechanism: Attention, can assist the network to put more emphasis on useful information [36]. Attention modules are used in many 2D visual problems (e.g. image segmentation [42, 8, 7], image denoising [1]), etc. For 3D point cloud analysis, we design a Channel-wise Affinity Attention module for feature map enhancement based on affinity between channels.

To deal with the unorderedness of regular point cloud data, PointNet [26] proposed to use symmetric functions to aggregate features, while most of the subsequent works apply max-pooling to extract prominent features. Despite the fact that prominent features are representative, they lack some details especially in local areas, hence, are insufficient for precise classification task. To address this problem, we propose a simple but effective way to learn fine-grained features via a shared fully connected (FC) layer on each local neighborhood. Although recent work [27, 18, 41, 47, 12] shows that CNN approaches can benefit from more geometric information, it could also negatively affect the performance since redundant or useless features can be incorporated. To maximize its advantage, we shall carefully form a low-level descriptor with explicit physical meaning to enrich the geometric information.

The contributions of our work can be summarized as:

- We design a feedback CNN mechanism for local prominent feature learning on point clouds.
- We introduce a Channel-wise Affinity Attention module to better refine high-level features of point clouds.
- We propose an intuitive but effective structure to extract local fine-grained features as complement. And we also show that our estimated mesh descriptor can significantly improve the performance of the network.
- We present experimental results showing that our proposed network on both synthetic and real-world 3D point cloud classification benchmarks outperform state-of-the-art methods.

2. Related Work

Estimating geometric relations. Although 3D scattered point cloud data has many advantages, the main drawback is the lack of geometric information. In order to acquire more underlying knowledge of point clouds, conventional methods [23, 21, 38] tried to estimate the geometry of point cloud e.g. face, normals, curvature, etc. and also proposed many hand-crafted features for recognition and matching, (e.g. shape context[11], point histograms[29], etc.). Besides, recent works [27, 18, 43, 46] with CNNs have better performance thanks to the permutation invariance of low-level geometry. To give advantages for the needs of modern methods, here we expand low-level geometric information from the given 3D coordinates, then integrate this as an explicit geometric descriptor for network processing.

Learning local features. PointNet [26] passes point cloud data through MLPs for a high-level feature representation of every single point, and successfully solved the unordered problem of point cloud data with a symmetric function. Due to the effectiveness, later works [27, 15, 43, 18] also adopt MLP based operation for point cloud processing. Meanwhile, researchers realize that local features are promising because they contribute additional characteristics to global features. Although the points are unordered in point cloud data, we may group points based on various metrics. Generally, one approach selects seed points as centroids, and then applies a query algorithm (e.g. Ball Query in [27]) based on 3D Euclidean distance to group points for local clusters. After extracting local features, the network may further process the centroids.

Another track is to find each point’s neighbors in embedding space based on N-dimensional Euclidean distance [43] and then group each point’s neighbors in the form of high dimensional vectors. In contrast to the previous type, this method can avoid sparsity and update dynamically in different feature dimensions. In terms of feature aggregation, max-pooling [26] is widely employed since it can solve the issue of unorderedness and gather information sufficiently. In spite of the benefits, there are some weak points of the current max pooling approach: it may lose local details or involve bias. To overcome these problems, we add complementary local fine-grained features and feedback structures to reduce possible bias.

Attention mechanism for CNN. The idea of attention has been successfully used in many areas of Artificial Intelligence (AI). Like human beings, the computational resource of machine is also limited. Thus we need to focus on important aspects. Previously, Vaswani et al. [36] proposed different types of attention mechanisms for neural machine translation. Subsequently, attention mechanisms were incorporated in visual tasks, for example, Wang et al. [42] extended the idea of Self-attention in spatial domain for computer vision problems. Also, SENet [8] credits winning the ImageNet [28] challenge to its channel-wise attention module. Other works [44, 7, 14] derive benefits from both spatial and channel domains of 2D images.

In terms of 3D point clouds, attention modules contribute to point clouds detection [25], generation [33], segmentation [48, 16, 49], etc. However, limited work is done in well-designed attention mechanisms targeting 3D point clouds classification. On this front, Xie et al. [46] utilized a spatial self-attention module for the shape context [11] of point clouds. Subsequent works [40, 6] also applied Graph Attention [37] module for the constructed graph features on point clouds. Differently from existing methods, we try to enhance the high-level representation of point cloud by cap-
Figure 2: Our network architecture for point cloud classification. In the beginning, the Naive Mesh Descriptor (NMD) can explicitly expand low-level 3D coordinates to a 14-degree geometric feature. Then a series of modules comprehensively learn the representation of the point cloud in high-level embedding spaces. The Attentional Feedback Module for EdgeConv (AFME) can aggregate prominent features of a local area, while the Fine-grained Edge Feature Extractor (FEEF) is able to provide more local details as complement. Further, the Channel-wise Affinity Attention (CAA) module is applied to avoid redundant information and refine the feature map along channels. Finally, we use the concatenated max pooling and average pooling results to form a global vector, by which the subsequent fully connected layers can regress the scores for possible classes.

3. Approach

As stated in Section 1, there are some unsolved problems of the existing methods. To tackle the challenges, we start from classical MLP mapping [26] and edge convolution [43]. We apply the basic MLP operation [26], i.e.
a 1×1 convolution on the point cloud feature map, usually followed by a batch normalization layer and an activation function:

\[
M(\cdot) := \tau(BN(c_{1 \times 1}(\cdot)))
\]

(1)

where \(M\) is the MLP, \(\tau\) is the activation function, \(BN\) is batch normalization, \(c\) is convolution and its subscript presents the filter size. Moreover, edge features \(f_\psi\) defined in [43] is given by:

\[
f_\psi(x_i) = (x_i, x_j - x_i), \quad x_i \in \mathbb{R}^d; \quad \forall x_j \in \text{Neighbors}(x_i)
\]

(2)

where \(x_i\) is the centroid of a local area in \(d\)-dimensional feature space, and the entire feature map can be formed as \(X_{N \times d} = [x_1^T, x_2^T, \ldots, x_N^T]\). The \(x_j\)’s are the neighbors found by \(k\)-nearest neighbors (knn) algorithm.

The aim of EdgeConv, \(E\), is to construct radial local graphs (i.e. edge features \(f_\psi\)) consisting of edges pointing from the neighbors to the centroids, and then mapped by a shared MLP in feature space:

\[
E(\cdot) := M(f_\psi(\cdot))
\]

(3)

Differently from these approaches, we attempt to incorporate more geometric features in different levels. As illustrated in Figure 2, our network has a series of modules with each consisting of two main parts: one extracting prominent features and the other learning fine-grained features. Our mesh descriptor explicitly expands geometric information in low-level space to meet the demand of comprehensive learning. In this section, we introduce the critical modules in detail and mathematically formulate the operations.

3.1. Attentional Feedback Module for EdgeConv

The typical feedback mechanism aims to accurately control a process by monitoring its actual output and feeding the error signal back to force the system to generate the desired output. To be more specific, the output is passed through a feedback path as a feedback signal, and then the forward path can use such feedback signals to both adjust and control the system. By forming such a closed loop, the system can reduce the error, improve stability, and enhance robustness. Inspired by this, we propose the Attentional Feedback Module for EdgeConv (AFME) illustrated in Figure 3 for edge-based prominent features.

Forward path. Here we employ EdgeConv as the forward path denoted as \(\Phi\) of our AFME since this operation can explicitly capture both global shape structure and local neighborhood information. With Equation 2 and 3, the forward path as can be formulated as:

\[
\Phi(\phi_i, x_i) = M(f_\psi(x_i)) = M(x_i, x_j - x_i)
\]

(4)
In general, we have three paths named the forward path (in green), the feedback path (in red), and the correction path (in blue). The error signal is defined as the difference between the feedback signal and the original input of AFME. By summing the inputs of forward and correction paths, we apply max pooling to extract local maximums and a Channel-wise Affinity Attention (CAA) module to refine them further. (*DashLine: the edge features of the forward path will be additionally used as the input of corresponding Fine-grained Edge Feature Extractor (FEFE))

And the output of forward path is:

\[ f_{\Phi_x} = \Phi(\phi_i, x_i); \quad f_{\Phi_x} \in \mathbb{R}^{d \times k} \]  

(5)

**Feedback and error signals.** Following our forward path \( \Phi \), an ideal output feature map should fully encode global and local details. On the other hand, if the output \( f_{\Phi_x} \) is indeed informative, we may restore the original input \( x_i \) from it. Suppose our feedback path (\( \Upsilon \)) aims to restore the input, then \( \Upsilon \)'s output, \( x_i' \), can be termed as a Feedback Signal:

\[
x_i' = \Upsilon(v_i, f_{\Phi_x}) = \Upsilon(v_i, \Phi(\phi_i, x_i)); \quad x_i' \in \mathbb{R}^d
\]

(6)

Further, we take the difference between the original and restored inputs, \( \Delta x_i \), to formulate corresponding Error Signal:

\[
\Delta x_i = x_i' - x_i
\]

(7)

**Feedback path.** Since we attempt to restore the input from \( f_{\Phi_x} \in \mathbb{R}^{d \times k} \), the feedback path needs to simulate a reverse process of the forward path (i.e., EdgeConv). In our proposal, we apply a shared Local Fully Connected (LFC) layer as the feedback path \( \Upsilon \) of our module:

\[
\mathcal{L}(\cdot) := \tau\left( BN(c_1 \times k(\cdot)) \right)
\]

(8)

where \( \mathcal{L} \) is the LFC. Mathematically, LFC is a special case of a shared MLP with kernel size \([1, k]\), by which the \( k \) neighbors in feature space can be fully connected. In contrast to EdgeConv expanding a center point to local neighbors in the new embedding space, the shared LFC layer can pull the neighbors back to the previous space \((d' \rightarrow d)\) and aggregate at a center point \((k \rightarrow 1)\) via learnable weights. In general, the feedback path follows:

\[
\Upsilon(v_i, f_{\Phi_x}) = \mathcal{L}(f_{\Phi_x})
\]

(9)

Based on Equations 5, 6 and 7, we can re-write the output of feedback path, i.e. Feedback Signal as:

\[
x_i' = \mathcal{L}(\Phi(\phi_i, x_i)); \quad x_i' \in \mathbb{R}^d
\]

(10)

and Error Signal as:

\[
\Delta x_i = \mathcal{L}(\Phi(\phi_i, x_i)) - x_i; \quad \Delta x_i \in \mathbb{R}^d
\]

(11)

**Correction path.** Finally, the module passes the Error Signal through a correction path (\( \Gamma \)), which has the same structure with forward path:

\[
\Gamma(\gamma_i, \Delta x_i) = \mathcal{E}(\Delta x_i) = \mathcal{M}(f_\psi(\Delta x_i))
\]

(12)

Therefore, the output features of the correction path can be formed as:

\[
f_{\Gamma_i} = \Gamma(\gamma_i, \Delta x_i); \quad f_{\Gamma_i} \in \mathbb{R}^{d \times k}
\]

(13)

In order to form the feedback loop, here we take the output of the correction path as the correction term for our original output of forward path. After that, we apply max-pooling over the local area to obtain a compact feature map. Moreover, the Channel-wise Affinity Attention (CAA, see Section 3.2 for details) module can further refine the final feature representation.

Finally, the operation of Attentional Feedback Module for EdgeConv (AFME) can be summarized as:

\[
f_i = AFME(x_i) = CAA(\max_{(k)}(f_{\Phi_x} + f_{\Gamma_i})); \quad f_i \in \mathbb{R}^d
\]

(14)

### 3.2. Channel-wise Affinity Attention Module

As mentioned in Section 2, most attention designs regarding point clouds operate in point-space, but the effects are not apparent. Instead, we prefer distributing attention weights along channels. Inspired by the spacetime non-local block [42], we can calculate the long-range dependencies without being concerned by point cloud data’s unorderedness. However, the corresponding calculations also have a high computational cost. Therefore we ought to find an appropriate method to avoid redundancy and refine the information in an abstract embedding space effectively and efficiently.
We propose our Channel-wise Affinity Attention (CAA) module targeting the channels of high-level point cloud feature maps. As Figure 4 shows, the main structure of the CAA module includes a Compact Channel-wise Comparator (CCC) block, a Channel Affinity Estimator (CAE) block, and a residual connection.

**Compact Channel-wise Comparator block.** Since the CAA module mainly focuses on channels, it is necessary to reduce the computing cost caused by the complexity in point-space. As we claimed above, it is hard to select the key points in such abstract high dimensional space. In the case of a given $d$-dimensional feature map $F_{N \times d}$, the Compact Channel-wise Comparator (CCC) block can simplify context in each channel by an shared MLP operating on channel vector $c_i$ (where $c_i \in \mathbb{R}^N$; and $F_{N \times d} = \{c_1, c_2, \ldots, c_d\}$) to implicitly replace $N$ original points with a smaller number $N' = N/ratio; \text{ ratio} > 1$. In contrast to explicitly selecting some points in abstract embedding space, CCC aims to efficiently reduce the size but sufficiently retain the information of each channel:

$$q_i = \mathcal{M}_q(c_i); \quad q_i \in \mathbb{R}^{N'}$$

$$k_i = \mathcal{M}_k(c_i); \quad k_i \in \mathbb{R}^{N'}$$

Specifically, $\mathcal{M}_q(\cdot)$ and $\mathcal{M}_k(\cdot)$ are two MLPs operating for Query Matrix and Key Matrix [36]:

$$Q_{N' \times d} = [q_1, q_2, \ldots, q_d]$$

$$K_{N' \times d} = [k_1, k_2, \ldots, k_d]$$

and we apply the product of transposed Query Matrix and Key Matrix to estimate corresponding channel-wise Similarity Matrix:

$$S_{d \times d} = Q^T K$$

where $S_{i,j}$ approximates the similarity between the $i^{th}$ channel and the $j^{th}$ channel of the given feature map $F_{N \times d}$.

**Channel Affinity Estimator block.** Typical self-attention structures used to calculate the long-range dependencies in spatial data based on inner-products, since the values can somehow represent the similarities between the items. In contrast, we define the non-similarities between channels and term it Channel Affinity. In our approach, the Channel Affinity Matrix of the feature map $F_{N \times d}$, can be modeled:

$$A_{d \times d} = \text{softmax} \left( \text{expand} \left( \text{max}(S) \right) - S_{d \times d} \right)$$ (15)

Particularly, we select the maximum similarities along the columns of $S$, and then expand them into the same size of $S$. By subtracting the original $S$ from the expanded matrix, the channels with higher similarities have lower affinities (illustrated in Figure 5(b)). Besides, $\text{softmax}$ is added to normalize the values, since $A_{d \times d}$ is used as the weight matrix for refinement. In this way, channels can put higher weights on other distinct channels, thereby avoid aggregating similar/redundant information.

According to the weight matrix, we can refine each point’s features by taking the weighted sum of all channels. We apply another MLP, $\mathcal{M}_v(\cdot)$, to get the Value Matrix as shown below:

$$V_{N \times d} = [v_1, v_2, \ldots, v_d]$$

$$v_i = \mathcal{M}_v(c_i); \quad v_i \in \mathbb{R}^N$$

This process can be easily achieved by the multiplication between $V_{N \times d}$ and the Channel Affinity Matrix. Additionally, we use a residual connection and learn a weight $\alpha$ to ease block training. The refined feature map by CAA is given below:

$$F'_{N \times d} = CAA(F) = F + \alpha \cdot V A$$ (16)
3.3. Geometric Features

For regular scattered point clouds, the given information about 3D coordinates is minimal. In our work, we attempt to enrich the geometric features of the point cloud from two aspects: (1) we describe the low-level relations explicitly, and (2) extract the high-level information implicitly.

Explicit geometric features. Here we define the explicit geometric features in low-level space to estimate features with explicit purposes. In geometry, a mesh is a type of well-constructed 3D data format, including faces, edges, as well as vertices. Similarly, we incorporate the estimated faces and edges to expand the low-level features representation of the 3D point clouds. Hence, we propose a Naive Mesh Descriptor (NMD) to enrich the original input data (i.e. 3D coordinates) with estimated face features.

Since most of the mesh data is constructed in triangle faces, we also adapt the point cloud for triangle mesh format. To be specific, firstly we search the two nearest neighbors, i.e. $k$nn with $k = 2$, in 3D space for point $p_i \in \mathbb{R}^3$, and then we form the triangle face corresponding to $p_i$ with the two neighbors: $p_{j1}, p_{j2} \in \mathbb{R}^3$. To explicitly describe the estimated triangle face, totally six items with exact geometric purposes are involved in:

$$p_i = (p_i, normal, edge_1, edge_2, length_1, length_2)$$  \hspace{1cm} (17)

Concretely:

$$\bar{p}_i = (x, y, z) \quad p_i \in \mathbb{R}^3$$
$$\text{normal} = edge_1 \times edge_2 \quad \text{normal} \in \mathbb{R}^3$$
$$edge_1 = p_{j1} - p_i \quad edge_1 \in \mathbb{R}^3$$
$$edge_2 = p_{j2} - p_i \quad edge_2 \in \mathbb{R}^3$$
$$length_1 = |edge_1| \quad length_1 \in \mathbb{R}^1$$
$$length_2 = |edge_2| \quad length_2 \in \mathbb{R}^1$$

Implicit geometric features. In contrast to explicit geometry in low-level space, we also expect to capture more implicit information in high-level space. As explained in Section 3.1, the Attentional Feedback Module for EdgeConv ($AFME$) can extract local prominent features via a max-pooling function in a high-level space. Although the prominent features can encode much geometric information for simple point clouds, more details are needed. Especially for some challenging cases e.g., real objects, complex scenes, or similar shapes etc., more fine-grained features are required for comprehensive feature representation.

Specifically, the edge features $f_\psi(x_i)$ from a $AFME$ are employed as the input of corresponding Fine-grained Edge Feature Extractor ($FEFE$, Figure 6). Instead of max-pooling for prominent features, the $LFC$ layer can aggregate more details from all neighbors. Besides, $CAA$ helps to refine the features for compact outputs. Therefore, the extracted fine-grained features are formulated:

$$\bar{f}_i = FEFE(f_\psi(x_i)) = CAA(L(f_\psi(x_i)))$$  \hspace{1cm} (18)

4. Experiments

In this section, we first provide the implementation and training details followed by the datasets we utilize for evaluation. We then analyze our network to establish the effects of different modules. Furthermore, we visualize the outputs, discuss the complexity of our model, and conclude this with the performance against state-of-the-art methods on synthetic and real-world point clouds.

Implementation details. Our proposed network starts with a Naive Mesh Descriptor, which expands the input 3D coordinates into a 14-degree geometric vector. Next, the geometric features are passed through four modules to learn high-level features in different embedding spaces i.e. 64, 64, 128, and 256. Moreover, each module has an Attentional Feedback Module for EdgeConv ($AFME$, the number of neighbors $k$ is 20) for extracting prominent features and a Fine-grained Edge Feature Extractor ($FEFE$) for local details.

To incorporate the information from different scales, we concatenate the output feature maps of the mentioned modules together, and a shared $MLP$ with $CAA$ module can further integrate them into a 1024 dimensional feature map.

---

**Table 1: Classification results (%) on ModelNet40 benchmark.**

| method                | input type | #points | avg class acc. | overall acc. |
|-----------------------|------------|---------|----------------|--------------|
| ECC [31]              | coords     | 1k      | 83.2           | 87.4         |
| PointNet [26]         | coords     | 1k      | 86.0           | 89.2         |
| SCN [46]              | coords     | 1k      | 87.6           | 90.0         |
| KD-Net [19]           | coords     | 1k      | -              | 90.6         |
| PointCNN [15]         | coords     | 1k      | 88.1           | 92.2         |
| PCNN [2]              | coords     | 1k      | -              | 92.3         |
| DensePoint [17]        | coords     | 1k      | -              | 92.8         |
| RS-CNN [18]           | coords     | 1k      | -              | 92.9         |
| DGCNN [43]            | coords     | 1k      | 90.2           | 92.9         |
| KP-Conv [34]          | coords     | 1k      | -              | 92.9         |
| Ours                  | coords     | 1k      | 91.0           | 93.8         |

**Note:**
- $coords$ and $norm$ are used to represent coordinates and norms.
- $voting$ is for multi-votes evaluation strategy.
- $k$nn is for the number of nearest neighbors.
- $\times$ denotes concatenation operation.
- $\cdot$ denotes element-wise product.
- $\ast$ denotes element-wise product.
- $\oplus$ denotes element-wise sum.
- $\ominus$ denotes element-wise subtraction.
- $\times$ denotes element-wise product.
- $\cdot$ denotes element-wise product.
- $\ast$ denotes element-wise product.
- $\ominus$ denotes element-wise subtraction.
- $\times$ denotes element-wise product.
- $\cdot$ denotes element-wise product.
- $\ast$ denotes element-wise product.
- $\ominus$ denotes element-wise subtraction.
We apply Stochastic Gradient Descent (SGD) with the momentum of 0.9 as the optimizer for training, and its initial learning rate of 0.1 decreases to 0.001 by cosine annealing [19]. The batch size is set to 32, and the number of training epochs is 300. Besides, we augment the training data with random scaling and translation as in [43], while there is no pre or post-processing performed during testing.

Datasets. We show the performance of the proposed network benefits from implicit and explicit geometric features is simple and intuitive, by comparing models 7 and 8 in Table 3, we can find a 0.4% improvement. To illustrate further, we present another ablation study concerning different modules of our network. It can be observed that the feedback module achieves well fitting (model 5&6). Further, the model benefits once we add the low-level geometric details of the estimated triangle face, including the vertex, face normal, and edges etc.

Visualization and complexity. From Figure 7 we can visualize corresponding learned features by AFME and FEFE modules in different layers of our network on ModelNet40. Particularly, all examples show the property of CNN: the shallow layers have higher impact on simpler features e.g., edges, corners, etc., while deep layers connect those simpler features for more semantically specific parts. As we stated before, AFME mainly extracts prominent features while FEFE asists to capture missing details. From the figure we can observe that FEFE complements AFME as expected.

Although we have similar operations to the competing methods e.g., FC layers, kmm algorithm, etc., we manage to simplify the complexity by sharing weights, reducing di-

| model | Naive Mesh Descriptor | length | overall acc. |
|-------|-----------------------|--------|--------------|
| 1     | \( \tilde{p}_1 = (p_i) \) | 3      | 93.4         |
| 2     | \( \tilde{p}_2 = (p_i, n, l_1, l_2) \) | 8      | 93.5         |
| 3     | \( \tilde{p}_3 = (p_i, e_1, e_2, l_1, l_2) \) | 11     | 93.5         |
| 4     | \( \tilde{p}_4 = (p_i, n, e_1, e_2) \) | 12     | 93.7         |
| 5     | \( \tilde{p}_5 = (p_i, n, e_1, e_2, l_1, l_2) \) | 14     | 93.8         |

Then we apply max-pooling and average-pooling in parallel over all channels for a global vector, by which an additional three fully connected layers (having 512, 256, c output) can regress the confidence scores for all possible categories. In the end, we employ cross-entropy between predictions and ground-truth labels as our loss function.

Training. We apply Stochastic Gradient Descent (SGD) with the momentum of 0.9 as the optimizer for training, and its initial learning rate of 0.1 decreases to 0.001 by cosine annealing [19]. The batch size is set to 32, and the number of training epochs is 300. Besides, we augment the training data with random scaling and translation as in [43], while there is no pre or post-processing performed during testing.

Datasets. We show the performance of the proposed network benefits from implicit and explicit geometric features is simple and intuitive, by comparing models 7 and 8 in Table 3, we can find a 0.4% improvement. To illustrate further, we present another ablation study concerning different modules of our network. It can be observed that the feedback module achieves well fitting (model 5&6). Further, the model benefits once we add the low-level geometric details of the estimated triangle face, including the vertex, face normal, and edges etc.

Visualization and complexity. From Figure 7 we can visualize corresponding learned features by AFME and FEFE modules in different layers of our network on ModelNet40. Particularly, all examples show the property of CNN: the shallow layers have higher impact on simpler features e.g., edges, corners, etc., while deep layers connect those simpler features for more semantically specific parts. As we stated before, AFME mainly extracts prominent features while FEFE asists to capture missing details. From the figure we can observe that FEFE complements AFME as expected.

Although we have similar operations to the competing methods e.g., FC layers, kmm algorithm, etc., we manage to simplify the complexity by sharing weights, reducing di-

To verify the functions and effectiveness of different parts in our network, here we conduct two ablation studies about the proposed modules and the contents of Naive Mesh Descriptor, respectively. We investigate the same proposed network on the ModelNet40 dataset.

Effects of different modules. Table 3 shows the results of ablation study concerning different modules of our network. It can be observed that the feedback module achieves well with EdgeConv, and the performance of model 5 shows a further enhancement with the CAA module applied (i.e., AFME). Besides, the results of model 3/4/5 prove that the network benefits from implicit and explicit geometric features. However, it is worth noting that increasing low-level geometric information (NMD) alone may not improve performance due to the redundant features that may cause overfitting (model 5&6). Further, the model benefits once we augment the geometrics in both low and high-levels.

Naive Mesh Descriptor. Although the idea of adding explicit geometric features is simple and intuitive, by comparing models 7 and 8 in Table 3, we can find a 0.4% improvement. To illustrate further, we present another ablation study to investigate the best representation of the Naive Mesh Descriptor. Table 2 shows the results of various possible forms of the NMD. According to the experiments, we conclude that the formation of the NMD for model 5 works better since these terms can comprehensively represent the low-level geometric details of the estimated triangle face, including the vertex, face normal, and edges etc.

Visualization and complexity. From Figure 7 we can visualize corresponding learned features by AFME and FEFE modules in different layers of our network on ModelNet40. Particularly, all examples show the property of CNN: the shallow layers have higher impact on simpler features e.g., edges, corners, etc., while deep layers connect those simpler features for more semantically specific parts. As we stated before, AFME mainly extracts prominent features while FEFE asists to capture missing details. From the figure we can observe that FEFE complements AFME as expected.

Although we have similar operations to the competing methods e.g., FC layers, kmm algorithm, etc., we manage to simplify the complexity by sharing weights, reducing di-

| model | FME | CAA | FEFE | NMD | overall acc. |
|-------|-----|-----|------|-----|--------------|
| 1     | ✓   | ✓   | ✓    | ✓   | 92.6         |
| 2     | ✓   | ✓   | ✓    | ✓   | 92.9         |
| 3     | ✓   | ✓   | ✓    | ✓   | 93.0         |
| 4     | ✓   | ✓   | ✓    | ✓   | 93.2         |
| 5     | ✓   | ✓   | ✓    | ✓   | 93.4         |
| 6     | ✓   | ✓   | ✓    | ✓   | 93.3         |
| 7     | ✓   | ✓   | ✓    | ✓   | 93.1         |
| 8     | ✓   | ✓   | ✓    | ✓   | 93.4         |
Table 4: Classification results (%) on ScanObjectNN benchmark.

| # shapes | overall acc. | avg class acc. | bag | bin | box | cabinet | chair | desk | display | door | shelf | table | bed | pillow | sink | sofa | toilet |
|----------|--------------|---------------|-----|-----|-----|---------|-------|------|---------|------|-------|-------|-----|--------|------|------|--------|
| 5k inputs | 83.5 | 58.1 | 39.8 | 62.8 | 15.0 | 65.1 | 84.4 | 36.0 | 62.3 | 85.2 | 60.6 | 66.7 | 51.8 | 61.9 | 46.7 | 72.4 | 61.2 |
| Ours | 90.8 | 90.8 | 90.8 | 90.8 | 90.8 | 90.8 | 90.8 | 90.8 | 90.8 | 90.8 | 90.8 | 90.8 | 90.8 | 90.8 | 90.8 | 90.8 | 90.8 |

Figure 7: Examples of the features learned by AFME and FEFE modules in shallow and deep layers of our network on ModelNet40. Shallow layers focus on edges/corners while deep layers cover more semantically meaningful parts. Besides, FEFE and AFME can capture complementary features, which are crucial for comprehensive point clouds representations.

4.2. Classification Performance

Results on synthetic point clouds. Table 1 shows the quantitative results on the synthetic ModelNet40 classification benchmark. The result of our network (overall acc: 93.8% and average class acc: 91.0%) exceeds state-of-the-art methods under the same test conditions\(^1\), our approach has a relatively good compromise between accuracy and model complexity. And we expect to further optimize the network for real-time applications.

5. Conclusion

In this paper, we propose a new CNN based module called the Attentional Feedback Module targeting some remaining problems of point cloud analysis: the feedback-like modules for edge features can automatically assist learning a better point cloud representation together with the Channel-wise Affinity Attention module that focuses on distinct channels. Besides, we involve more explicit geometries using Naive Mesh Descriptor and implicit geometries by Fine-grained Edge Feature Extractor. To compare our method with other state-of-the-art networks, we conduct experiments on both synthetic and real-world datasets. The results show the effectiveness and robustness of our approach.

\(^1\)Please refer to our supplementary material for more experimental results.

\(^2\)PB.T50.RS, the hardest case of ScanObjectNN dataset
References

[1] Saeed Anwar and Nick Barnes. Real image denoising with feature attention. In The IEEE International Conference on Computer Vision (ICCV), October 2019. 2

[2] Matan Atzmon, Hagai Maron, and Yaron Lipman. Point convolutional neural networks by extension operators. arXiv preprint arXiv:1803.1009J, 2018. 6

[3] Yizhak Ben-Shabat, Michael Lindenbaum, and Anath Fischer. 3dmfv: Three-dimensional point cloud classification in real-time using convolutional neural networks. IEEE Robotics and Automation Letters, 3(4):3145–3152, 2018. 8

[4] François Blais et al. Review of 20 years of range sensor development. Journal of electronic imaging, 13(1):231–243, 2004. 1

[5] Chunshui Cao, Xianming Liu, Yi Yang, Yinan Yu, Jiang Wang, Zilei Wang, Yongzhen Huang, Liang Wang, Chang Huang, Wei Xu, et al. Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks. In Proceedings of the IEEE International Conference on Computer Vision, pages 2956–2964, 2015. 1

[6] Can Chen, Luca Zanotti Fragonara, and Antonios Tsiourdos. Gapnet: Graph attention based point neural network for exploiting local feature of point cloud. arXiv preprint arXiv:1905.08705, 2019. 2

[7] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3146–3154, 2019. 2

[8] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 7132–7141, 2018. 2

[9] Michel Jaboyedoff, Thierry Oppikofer, Antonio Abellán, Marc-Henri Derron, Alex Løy, Richard Metzger, and Andrea Pedrazzini. Use of lidar in landslide investigations: a geometric optimization perspective. In Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, pages 22–31. ACM, 2004. 1

[10] Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In Proceedings of the IEEE International Conference on Computer Vision, pages 863–872, 2017. 6

[11] Marcel Körtgen, Gil-Joo Park, Marcin Novotni, and Reinhard Klein. 3d shape matching with 3d shape contexts. In The 7th central European seminar on computer graphics, volume 3, pages 5–17. Budmerice, 2003. 2

[12] Jiaxin Li, Ben M Chen, and Gim Hee Lee. Pointcn: Convolution on x-transformed point clouds. In Proceedings of the IEEE International Conference on Computer Vision, pages 922–928. IEEE, 2015. 1

[13] Yongcheng Liu, Bin Fan, Gaofeng Meng, Jiwen Lu, Shiming Xiang, and Chunhong Pan. Denspoint: Learning densely contextual representation for efficient point cloud processing. In The IEEE International Conference on Computer Vision (ICCV), October 2019. 6

[14] Wei Li, Xiatian Zhu, and Shaogang Gong. Harmonious attention network for person re-identification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2285–2294, 2018. 2

[15] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcn: Convolution on x-transformed points. In Advances in Neural Information Processing Systems, pages 820–830, 2018. 2, 6, 8

[16] Xinhai Liu, Zhizhong Han, Yu-Shen Liu, and Matthias Zwicker. Point2sequence: Learning the shape representation of 3d point clouds with an attention-based sequence to sequence network. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 8787–8785, 2019. 2

[17] Yongcheng Liu, Bin Fan, Gaofeng Meng, Jiwen Lu, Shiming Xiang, and Chunhong Pan. Denspoint: Learning densely contextual representation for efficient point cloud processing. In The IEEE International Conference on Computer Vision (ICCV), October 2019. 6

[18] Niilo J Mitra, Natasha Gelfand, Helmut Pottmann, and Leonidas Guibas. Registration of point cloud data from a geometric optimization perspective. In Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, pages 22–31. ACM, 2004. 1

[19] Niilo J Mitra and An Nguyen. Estimating surface normals in noisy point cloud data. In Proceedings of the nineteenth annual symposium on Computational geometry, pages 322–328. ACM, 2003. 2

[20] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. Training a feedback loop for hand pose estimation. In Proceedings of the IEEE International Conference on Computer Vision, pages 3316–3324, 2015. 1

[21] Anshul Paigwar, Ozgur Erkent, Christian Wolf, and Christian Laugier. Attentional pointnet for 3d-object detection in point clouds. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 652–660, 2017. 1, 2, 3, 6, 8

[22] 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 0–0, 2019. 2

[23] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In Advances in neural information processing systems, pages 5099–5108, 2017. 2, 6, 8
[28] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhifeng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 2

[29] 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. 1, 2

[30] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient ransac for point-cloud shape detection. In *Computer graphics forum*, volume 26, pages 214–226. Wiley Online Library, 2007. 1

[31] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3693–3702, 2017. 6

[32] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015. 1

[33] Yongbin Sun, Yue Wang, Ziwei Liu, Sanjay E Sarma, and Sanjey J Saorma. Pointgrow: Autoregressively learned point cloud generation with self-attention. *arXiv preprint arXiv:1810.05591*, 2018. 2

[34] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, Francois Goulette, and Leonidas J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 6

[35] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 7, 8

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 2, 5

[37] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017. 2

[38] George Vosselman, Sander Dijkman, et al. 3d building model reconstruction from point clouds and ground plans. *International archives of photogrammetry remote sensing and spatial information sciences*, 34(3/W4):37–44, 2001. 1, 2

[39] Chu Wang, Babak Samari, and Kaleem Siddiqi. Local spectral graph convolution for point set feature learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–66, 2018. 6

[40] Lei Wang, Yuchun Huang, Yaolin Hou, Shenman Zhang, and Jie Shan. Graph attention convolution for point cloud semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10296–10305, 2019. 2

[41] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions on Graphics (TOG)*, 36(4):72, 2017. 2

[42] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7794–7803, 2018. 2, 4

[43] 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 (TOG)*, 38(5):146, 2019. 2, 3, 6, 7, 8

[44] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cham: Convolutional block attention module. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018. 2

[45] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Lingguang Zhang, Xiaou 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. 7

[46] Saining Xie, Sainain Liu, Zeyu Chen, and Zhuowen Tu. Attentional shapecontextnet for point cloud recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4606–4615, 2018. 2, 6

[47] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 87–102, 2018. 2, 6, 8

[48] Wenhao Zhang and Chunxia Xiao. Pcan: 3d attention map learning using contextual information for point cloud based retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12436–12445, 2019. 2

[49] Kang Zhiheng and Li Ning. Pyramid: Point cloud pyramid attention network and graph embedding module for classification and segmentation. *arXiv preprint arXiv:1906.03299*, 2019. 2