Semantic Feature Mining for 3D Object Classification and Segmentation

Weihao Lu[1], Dezong Zhao[1], Cristiano Premebida[2], Wen-Hua Chen[3], Daxin Tian[4]

***Abstract*— Deep learning on 3D point clouds has drawn much attention, due to its large variety of applications in intelligent perception for automated and robotic systems. Unlike structured 2D images, it is challenging to extract features and implement convolutional networks over these unordered points. Although a number of previous works achieved high accuracies for point cloud recognition, they tend to process local point information in such a way that semantic information is not fully encoded. In this paper, we propose a deep neural network for 3D point cloud processing that utilizes effective feature aggregation methods emphasizing both generalizability and relevance. In particular, our method uses fixed-radius grouping for pooling layers and spherical kernel convolution for semantics mining. To address the issue of gradient degradation and memory consumption of a deep network, a parallel feature feed-forward mechanism and bottleneck layers are implemented to reduce the number of parameters. Experiments show that our algorithm achieves state-of-the-art results and competitive accuracy in both classification and part segmentation while maintaining an efficient architecture.**

I. INTRODUCTION

3D point cloud processing has been a popular research topic, since the use of multi-channel laser scanners became more spread out. It has a close relation to a wide range of applications such as augmented reality, autonomous driving and robotics [1], [2], [3], [4]. Accurate measurements of the surrounding provided by point clouds facilitate the interactions between intelligent systems and their environments. Although this 3D data format provides extra range information compared to conventional cameras, the subsequent analysis is limited by unordered sets of points, see Figure 1. It is challenging to give accurate inferences of object shapes based on abstract 3D shape patterns. To solve this issue, sufficient contextual information needs to be captured thoroughly in multiple scale regions.

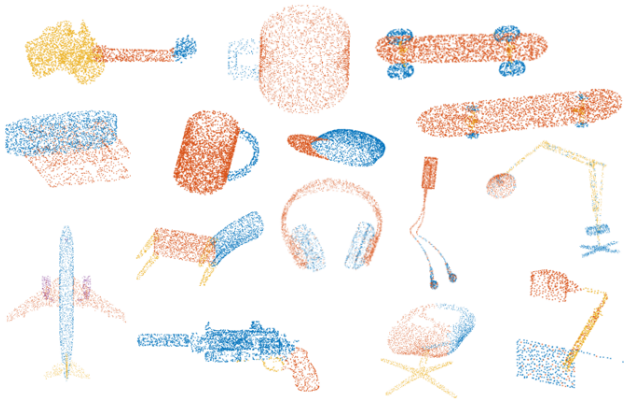

Fig. 1. Visualization of ShapeNet dataset [13]. Points of object parts, which are useful for robotic grasping, are indicated by different colors. It can still get ambiguous to infer the correct categories of the parts, especially when there are no sharp edges of transition.

With the rise of convolutional neural networks (CNN) in areas of computer vision and robotic perception, remarkable performances have been seen in extracting dense semantic information from images and videos [5], [6], [7]. Unlike 3D point clouds, images are defined in regular grids, i.e., pixels across different scales. This allows a direct implementation of convolution operations. Accordingly, much effort has been devoted to transferring the success of CNN from image analysis to irregular point cloud processing. These methods project point clouds onto regular data structures [8], [9], [10], [11], [12], i.e., multi-view images or volumes, in order to apply well-developed CNN. This approach has a high demand of memory, as well as the risk of losing 3D geometric relations among the points.

Another effective solution was pioneered by PointNet [1], where point features are directly learned from irregular points. PointNet [1] shows impressive results on 3D object recognition and segmentation by applying symmetric functions, eg. max-pooling, after encoding point coordinates with fully-connected layers. However, information at local scales might not be perceived properly, as a global pooling operator is used. To describe the local information, subsequent research utilizes hierarchical data structures [14], [15]. SO-Net [16] extracts local features by creating a self-organizing map. PointNet++ [15] applies PointNet to local subsets of points. Although capturing semantics at multi-scale, the captured information might not be optimally processed. This is because deep neural network tends to fail at retaining shallow features.

In addition to acquiring multi-scale information, local feature aggregation also plays an important role in shape

*This work was sponsored in part by the Engineering and Physical Sciences Research Council of the UK under the EPSRC Innovation Fellowship (EP/S001956/1), in part by the Newton Advanced Fellowship (UK-China International Cooperation and Exchange Project) jointly supported by the UK Royal Society (NAF\R1\201213)) and the National Natural Science Foundation of China (62061130221), and in part by the State Key Laboratory of Automotive Safety and Energy under Project No. KF2009.

[1]Weihao Lu and Dezong Zhao are with the School of Engineering, University of Glasgow, University Avenue, Glasgow G12 8QQ, UK w.lu.1@research.gla.ac.uk, dezong.zhao@glasgow.ac.uk

[2]Cristiano Premebida is with the Institute of Systems and Robotics, University of Coimbra, 3030-290 Coimbra Portugal cpremebida@deec.uc.pt

[3] Wen-Hua Chen is with the Department of Aeronautical and Automotive Engineering, Loughborough University, Epinal Way, Loughborough LE11 3TU, UK w.chen@lboro.ac.uk

[4]Daxin Tian is with the School of Transportation Science and Engineering, Beihang University, Haidian District, Beijing 100191, China dtian@buaa.edu.cn

recognition. A typical method is to use multi-layer perceptrons (MLP) [15], [17], [18]. MLP based methods encode the semantics of a query point by concatenating point features and relative positions of surrounding points, followed by convolution and pooling operations. Kernel based methods use pre-defined kernel points for convolution [19], [20]. Though many convolution operators have been designed for feature aggregation, most research only adopts one kind of operator in shallow networks.

In short, there are two problems to tackle when analyzing 3D point clouds with CNN: 1) by using traditional architectures the shallow features are not well-preserved by the end of a deep network, resulting in insufficient semantics; 2) local convolution operators that efficiently run at different scales in deep networks are required.

To fill these gaps, we propose a new general architecture to process point clouds directly with a relatively deep network, which can achieve the state-of-the-art results through efficient training. To alleviate the vanishing-gradient problem, our network is inspired by the residual network structure [6] and dense connections [7]. It strengthens shallow features and emphasizes feature propagation by leveraging shortcuts across the network layers. Since features are fully exploited by the end of the network, the number of parameters are dramatically reduced. Motivated by 2D CNNs on grid-like data, the convolution operator is designed to search for k-nearest neighbors around sample points. The neighbors are then sorted by the relative distance, thus allowing convolution operations on ordered points to efficiently excavate semantic information at a local scale. On the pooling layers, an MLP based operator is used with a fixed-radius search for generalizability.

Our major contributions are as follows:

- A multi-scale local feature aggregation method is proposed. It can effectively encode semantics information of points ensuring both generalizability and high relevance, as well as maintaining permutation invariance for irregular points;
- A general deep learning architecture consisting of the proposed operator is constructed. This deep network architecture exploits point clouds for rich contextual information with the help of the feature feed-forward mechanism and shortcut connections;
- Comprehensive experiments are carried out, which show that our proposed network can achieve the state-of-the-art performance on shape recognition and segmentation.

II. RELATED WORK

While early methods use hand-crafted descriptors [21] to analyze 3D data, the recent growth in the availability of 3D datasets [22], [23], [24] has boosted the development of deep learning algorithms on shape recognition and segmentation. In this section, we briefly review the existing research in deep learning with 3D point clouds, particularly on the feature learning for object classification and part segmentation.

A. Projection-based methods

By transforming the irregular point sets into a grid-like structure, a conventional CNN can be applied to handle 3D data for further analysis. One of the methods is to project the 3D points onto 2D planes from multiple perspectives [25], [9], [8], [26], [27]. A conventional CNN is then applied on a collection of 2D images. However, it cannot encode 3D information properly due to occlusions, thus raising difficulties when propagating features for 3D segmentation.

Alternatively, a point cloud is registered in unit volume grids [10], [28] as the input. Therefore a 3D CNN can perform natively over the voxelized data. These methods are limited by the trade-off between the memory usage and voxel resolution. Subsequent works utilize efficient data structures, eg. Octrees to reduce the memory consumption [29], [30]. However, the results have reflected the insufficient resolution for accurate voxel representation.

B. Point-based methods

Besides converting point clouds into regular formats, many works focus on consuming raw point clouds for learning. PointNet [1] is a network that pioneers in this direction by encoding point-wise feature with MLPs, followed by a symmetric function to maintain the order invariance of points. Huang et al. [31] improves PointNet by employing recurrent neural networks. However, all features are captured at the global scale, making it insensitive to local shapes. To alleviate this problem, researchers choose to learn local geometric information by dividing the 3D points into subsets [32], [14], [33]. Qi et al. [15] proposed PointNet++, which adopts a hierarchy to capture local geometric information from neighboring points, and uses PointNet [1] for feature aggregation.

C. Convolution-based methods

Recent research shows great interests in designing convolution operators for capturing semantic information at multiple scales. RS-CNN [34] takes the relative positions of a set of points as its input, and learns the weights with MLPs. ConvPoint [35] uses convolution in both euclidean and feature spaces, where the mapping is also learned with a simple MLP. Octree guided CNN [36] performs convolution with a spherical kernel by assigning surrounding points to the partitions of spherical neighborhood. Thomas at el. [19] proposed KPConv, where input points are convolved through a set of rigid or deformable kernel points with filter weights on each point. By leveraging the $\mathcal{X}$-Conv transformation that is learned through MLP, permutation invariance is achieved by PointCNN [37]. PointCNN learns transformation matrix separately, causing slow converge during training. A-CNN captures local neighborhood geometry by annular convolutions, whose kernel size can be arbitrarily defined. DGCNN [17] introduces a graph convolution, EdgeConv, over the weights between a point and its neighbors in feature space, which are dynamically optimized.

However, these networks encode point features with only a single type of convolution operator. These operators might

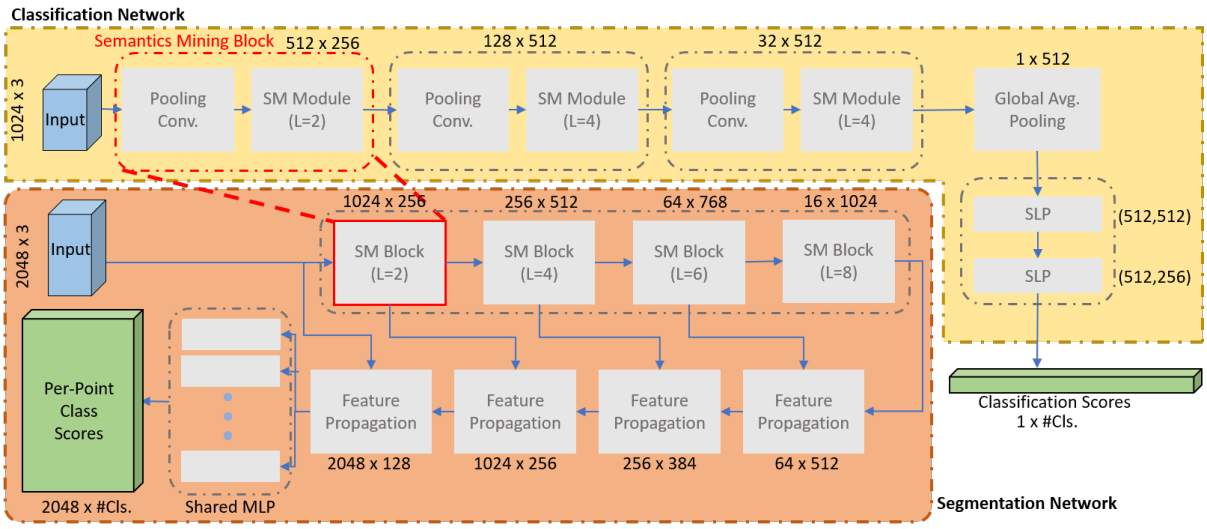

Fig. 2. Overview of the proposed architecture. The upper part shows the classification branch of the network, which consists of three semantics mining blocks, followed by a global average pooling layer to summarize the encoded feature. Two single-layer perceptrons (SLP) are used to output class scores. The lower part shows the network configuration for segmentation. Four semantics mining blocks are used paired with four feature propagation layers for point up-sampling. Rich semantic information is propagated to all input points, followed by a shared MLP for per-point classification. The numbers indicate the output feature-map size of each module in the format of $N \times C$, where N is the number of points and C is the width of feature channels. L refers to the number of SM layers within the module.

not be optimized or specialized for various receptive fields. Also the input might be 'washed-out' through these operators in a deep neural network. Instead, our proposed method constructs multi-level hierarchical network, utilizing two different operators for multi-scale local feature aggregation.

III. SEMANTICS MINING ARCHITECTURE

To construct an efficient deep learning network for 3D point clouds, we first need a convolution operator that is able to process points directly. Thus a network architecture that can preserve the gradient details through multiple layers needs to be built. To define our problem, we take a set of points as the input, which can be denoted as $\mathbf{P} = \{p_i \,|\, i = 1, 2, 3, ..., N\}$, where N is the number of points and p_i is a 1×3 vector of Cartesian coordinates (x, y, z) in the metric space. These points are defined on the object surface. Inputs, like RGB colors and normal vectors, can be added as input features in a format of vector (r, g, b) and (n_x, n_y, n_z). For simplicity, we only use point coordinates for the inference task. In short, we have an $N \times 3$ matrix as the input representing the point cloud.

For object classification, the proposed network outputs a $1 \times k$ score vector for k categories. For segmentation task, our network outputs individual labels for each point and thus an $N \times k$ score matrix for each of the k classes. The network layout is illustrated in Figure 2.

A. Pooling Layer

Point sets usually contain different numbers of points, causing an issue with convolution that is defined with a fixed kernel shape. Most of the networks, such as [15], address this problem by down-sampling and pooling layers. The purpose of these operations is to obtain a fixed-size input as well as decreasing the spatial resolution to enlarge the receptive field. This general idea in [15] is followed by pooling in our case. To analyze point clouds directly, conventional convolutions cannot be used. Here, we perform transformation and aggregation on a sampled point and its neighbors. The neighborhood of the sampled point q is defined by Ω_q. For a point $\mathbf{p_i}$ that lies within the neighborhood Ω_q, the vector (x, y, z) is transformed to a local coordinate system for convolutions by a transformation function τ. An MLP is then applied, followed by a non-linear activation function. The n-th channel of feature corresponding to a single sampled point q and neighbors p_i is defined by:

$$F(q)^{(n)} = \sigma(\zeta(\tau(p_i))) = \sigma(\mathbf{w_i^{(n)}} \cdot \tau(p_i)) \;\forall p_i, \qquad (1)$$

where ζ denotes the convolution, σ is the non-linearity and $\mathbf{w^{(n)}}$ is the shared convolution weights. This operation achieves permutation invariance by leveraging a symmetrical pooling function ρ. A pooling function that summarizes the feature regardless of the point order makes CNNs feasible on irregular points.

Specifically, the Pooling Convolution (PC) module picks a subset of points q through a down-sampling function. The sampling function, s, is defined as farthest point sampling (FPS) in this case. Compared to random sampling, iterative FPS is better at coping with non-uniform data density and thus a better coverage over the metric space. The receptive field also has a more balanced distribution over the entire point cloud.

The neighborhood, Ω_q, is formed through a grouping function g, such that the centroid points can be encoded with more generalizable features regarding to the local region. This issue is addressed by utilizing a fixed radius neighbor search, which finds points within a given radius

Algorithm 1 Pooling Convolution (PC) Module

Input: P, $\mathbf{f_{in}}$, Z: the matrix operation of ζ
Output: q, $\mathbf{F_q}$

1: Down-sample from point set: $q \leftarrow s(P)$
2: Form neighborhood around q: $\Omega_q \leftarrow g(q, P)$
3: Transform to local coordinates:
$p'_i = \tau(p_i) = p_i - q, \forall p_i \in \Omega_q$
4: Convolve over transformed feature:
$\mathbf{p'} = \{p'_i \,|\, i = 1, ..., K\}$,
$\mathbf{f(q)} = \sigma(Z(\text{concat}(\mathbf{f_{in}}, \mathbf{p'})))$
5: Aggregate with symmetric function:
$\mathbf{F(q)} = \rho(\{\mathbf{f_q}\})$

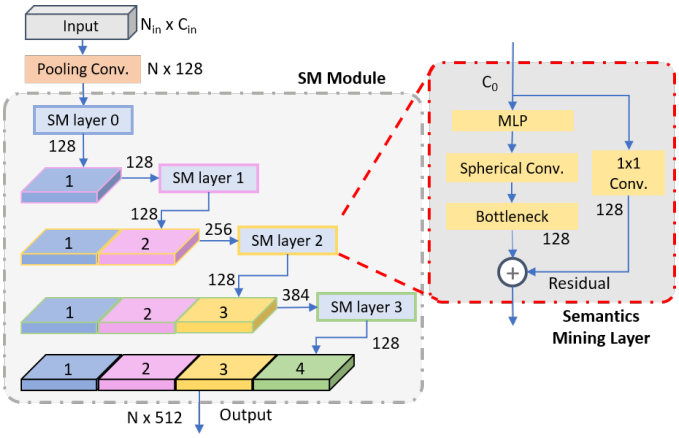

Fig. 3. Illustration of a four-layer semantics mining module. Operational layers are labeled with description and the features are the boxes with number (1, 2, 3 and 4). For any point cloud input, the pooling convolution elevates the dimentionality as well as down-sampling the points before feeding data to the SM module. Inspired by DenseNet [7] and DensePoint [33], each SM layer convolves over the concatenated features. This relation is shown by the colors of the boxes (eg. SM layer 2 convolves over feature 1 and 2 to give output feature 3, where the operation and input feature boxes are in yellow frames and the output feature box is filled with yellow). Inspired by [6], the detail of each SM layer is also illustrated on the right with bottleneck layer and residual shortcut.

corresponding to the query points. Yet it lacks the ability to adapt to a change in the scale of the entire point cloud. An adaptive radius mechanism may solve the problem. Each group is given a limit, K, on the number of neighbors to ensure the memory consumption during training, as well as guaranteeing a fixed scale region for a more generalizable receptive field.

Particularly, the process is summarized in Algorithm 1. A sampling function and a grouping function are denoted as s and g. This operator outputs a collection of sampled points $\mathbf{q} = \{q_i \,|\, i = 1, ..., N'\}$ of size $N' \times 3$ (where $N' \leq N$ is number of sampled points), as well as feature $\mathbf{F(q)}$ of size $N' \times K \times C$, with C being the number of feature channels. With the Pooling Convolution, the number of points is reduced with an increase in feature dimensionalities.

B. Semantics Mining Layer

To exploit the underlying information, we propose a Semantics Mining (SM) layer that is able to abstract deep semantics efficiently from points. Compared to the pooling layer, although the same input is given, the feature mining layer emphasizes the depth of convolution. While the pooling layer keeps the generalizability, this layer focuses on local regions for distinctive local geometric patterns. Besides, it also requires to maintain the permutation invariance and retain the low-level relations (eg. relative positions) between points. Inspired by spherical convolutions from several existing works [20], [38], [39], [40], we construct a spherical convolution operator embedded in the SM layer.

For simplicity, the spherical convolution operator is built upon PC modules. Compared to PC modules, SM layers gather neighboring points with a k-nearest neighbor (kNN) search instead of the fixed-radius approach. This guarantees the relevance of the gathered points by only picking the top k-nearest points. The kNN search also has a natural order from the closest to the farthest, which remedies the irregularity.

After the low-level features are lifted by an MLP, the neighborhood is divided into partitions. Computation complexity will be significantly reduced by picking up the representative features from each of the partitions. Practically, to achieve this operation we choose a pooling function over a stride convolution, since the latter is more costly. The shortened feature space parameterized by $d = 1, 2, ..., D$ is represented by:

$$f(d)^{(n)} = \rho(\{f(p)^{(n)}\}),\ d \in \Omega_q,\ p \in \Omega_q. \tag{2}$$

By reducing the spatial resolution, we are able to apply a more efficient convolution for feature aggregation. The general formulation is:

$$F(q)^{(n)} = \sigma(\zeta(f(d)^{(n)})) = \sigma(\mathbf{w_i^{(n)}} \cdot f(d)). \tag{3}$$

C. Residual and Dense Connection

A conventional CNN architecture tends to fail on retaining the input features and gradients when it becomes deeper [7]. Although there are more layers, a layer's weights are determined purely by its previous layer. This results in an inefficient learning process, as the final feature width can only be increased by the last layer. By leveraging the idea of dense connections [7], a feed-forward mechanism is able to stack the learned feature from all of the previous layers and thus no extra parameters are required to increase the last layer feature width. Different from DensePoint [33], reference to the layer inputs are added with the aid of residual shortcuts. Additionally, the weights can be learned at multiple scales. By stacking multiple Semantic Mining layers, we introduce the Semantic Mining (SM) module that utilizes the dense connection structure [7]. To work with the entire network, a SM module is paired with a PC modules forming an independent block to progress into the next scale region, see Figure 3.

D. Bottleneck Layer

Besides, the design of a shortcut connection [6] is found effective for our model. The shortcut connection is adopted

TABLE I

PART SEGMENTATION RESULTS ON SHAPENET DATASET [13].

Method	class mIoU	instance mIoU	air plane	bag	cap	car	chair	ear phone	guitar	knife	lamp	laptop	motor bike	mug	pistol	rocket	skate board	table
Kd-Net [41]	77.4	82.3	80.1	74.6	74.3	70.3	88.6	73.5	90.2	87.2	81.0	94.9	57.4	86.7	78.1	51.8	69.9	80.3
PointNet [1]	80.4	83.7	83.4	78.7	82.5	74.9	89.6	73.0	91.5	85.9	80.8	95.3	65.2	93.0	81.2	57.9	72.8	80.6
KCNet [14]	82.2	84.7	82.8	81.5	86.4	77.6	90.3	76.8	91.0	87.2	84.5	95.5	69.2	94.4	81.6	60.1	75.2	81.3
DGCNN [17]	82.3	85.1	84.2	83.7	84.4	77.1	90.9	78.5	91.5	87.3	82.9	96.0	67.8	93.3	82.6	59.7	75.5	82.0
RS-Net [31]	81.4	84.9	82.7	86.4	84.1	78.2	90.4	69.3	91.4	87.0	83.5	95.4	66.0	92.6	81.8	45.1	75.8	82.2
PCNN [11]	81.8	85.1	82.4	80.1	85.5	79.5	90.8	73.2	91.3	86.0	85.0	95.7	73.2	94.8	83.3	51.0	75.0	81.8
SO-Net [16]	80.8	84.6	81.9	83.5	84.8	78.1	90.8	72.2	90.1	83.6	82.3	95.2	69.3	94.2	80.0	51.6	72.1	82.6
PointNet++ [15]	81.9	85.1	82.4	79.0	87.7	77.3	90.8	71.8	91.0	85.9	83.7	95.6	71.6	94.1	81.3	58.7	76.4	82.6
Ours	82.1	**85.4**	83.1	77.5	86.1	78.2	90.5	77.7	91.3	89.0	83.8	95.4	67.0	93.1	81.2	62.3	75.5	83.5

across each SM layer in our network. Noticeably, it becomes computationally expensive even if the output feature-map size of each layer is reasonably small. It has been mentioned in [6], [7] that a bottleneck layer can be added to the network to address this issue. A bottleneck layer is introduced to the proposed network through a 1×1 convolution to reduce the feature-map size to C/r, where r is the bottleneck ratio. See Figure 3 for the layer arrangement.

IV. EXPERIMENTS

In this section, the performance of our network with classification and part segmentation is evaluated. We consider the ModelNet40 dataset [22] for object classification and ShapeNet [13] for part segmentation. The ModelNet40 dataset contains 9843 3D objects for training and 2468 for testing of 40 categories. The ShapeNet dataset consists of 16881 3D objects of 16 types, with 50 distinctive part classes, see Figure 1 for examples. Each type of object is partitioned into 2 to 6 parts. For both datasets, each object is represented by points generated from the surface of 3D CAD models.

A. Classification

1) Implementation Details: For classification, the network is configured as shown in Figure 2. The semantics mining back-end is composed with three blocks, with the number of SM layers being $L = (2, 4, 4)$ (i.e., each block has 2, 4 and 4 Semantics Mining layers respectively). The kernel numbers of each block are 512, 128 and 32. Listed below is the summary of the key parameters.

- $\rho =$ max-pooling
- $\sigma =$ ReLU
- Number of partitions, $D = 2$
- Bottleneck ratio, $r = 2$
- PC module feature-map size, $N \times 128$
- SM layer feature-map size, $N \times 128$

Following a global average pooling layer, a 2-layer MLP of size (512,512) and (512,256) is used for classification with Batchnorm and a 0.5 dropout ratio for each layer. At the end a soft-max classifier outputs a $B \times 40$ prediction matrix, where B is the batch size. The network takes 1024 points as its input. It is trained with the ADAM optimizer and a batch size of 72. The learning rate is set to 0.001 with a decay rate of 0.7 every 21 epochs.

TABLE II

OBJECT CLASSIFICATION RESULTS ON MODELNET40 DATASET [22].

Method	Input	# Points	Acc.(%)
PointNet [1]	xyz	1k	89.2
Kd-Net (depth=15) [41]	xyz	32k	91.8
PointNet++ [15]	xyz	1k	90.7
KCNet [14]	xyz	1k	91.0
A-CNN [18]	xyz	1k	92.6
DGCNN [17]	xyz	1k	92.9
KP-Conv rigid [19]	xyz	6.8k	92.9
PointCNN [37]	xyz	1k	92.2
ShellNet [20]	xyz	1k	93.1
SO-Net [16]	xyz	2k	90.9
PointNet++ [15]	xyz, normal	5k	91.9
SO-Net [16]	xyz, normal	5k	93.4
Ours ($L = (2, 4)$)	xyz	1k	93.1
Ours ($L = (2, 4, 4)$)	xyz	1k	**93.2**
Ours ($L = (4, 4, 4)$)	xyz	1k	92.9
Ours ($L = (2, 6, 6)$)	xyz	1k	92.8

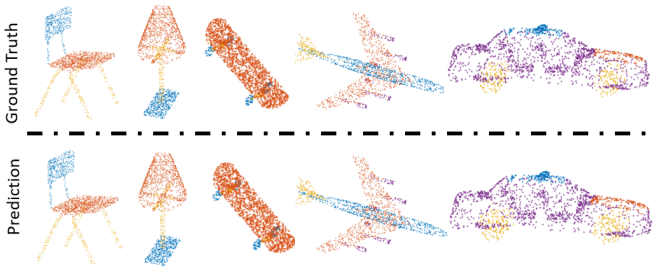

Fig. 4. Visualization of part segmentation results on ShapeNet. The predicted results are compared with the ground truth labels.

2) Results: The results of the state-of-the-art methods and our network on the ModelNet40 dataset are listed in Table IV-A.2. Note that xyz denotes the point coordinates and normal denotes the surface normal vector at that point. We also compare the performance of a 2-block model with $L = (2, 4)$. It can be seen that the 2-block network achieves a competitive result with an input of only 1024 points. With an extra block, our baseline network achieves an accuracy of 93.2%. This is because the extra block captures deeper semantics by increasing the size of the receptive field by lowering the spatial resolution. However, an increase in the number of SM layer does not benefit the network.

TABLE III
NUMBER OF TRAINABLE PARAMETERS

Method	# params.	Acc. (%)
PointNet [1]	3.5M	89.2
PointNet++ [15]	1.48M	91.9
3DmFV [42]	45.77M	91.6
KPConv [19]	14.3M	92.9
DGCNN [17]	1.81M	92.9
PointCNN [37]	0.6M	92.2
3D-GCN [43]	0.89M	92.1
Ours	1.7M	**93.2**

TABLE IV
COMPARISON OF NETWORK CONFIGURATIONS

Config.	DC	Res.	B/r	Neighbor	Acc. (%)
A	-	-	-	kNN	91.6
B	X	-	-	kNN	92.3
C	X	X	-	kNN	92.8
D	X	X	8	kNN	92.3
E	X	X	4	kNN	92.4
F	X	X	2	fixed-radius	92.5
Base	X	X	2	kNN	**93.2**

B. Segmentation

1) Implementation Details: For segmentation, the network is configured as shown in Figure 2. Four blocks are used for the back-end part, with the number of SM layers being $L = (2, 4, 8, 8)$. The kernel numbers of each block are 1024, 256, 64 and 16. Most of the key parameters are kept the same to the classification model. Besides, four feature propagation (FP) layers are used. Each FP layer propagates the semantics to neighboring points and concatenates with the features encoded by the back-end network. A shared MLP of size (128,128) is used for segmentation with dropout rates of 0.5 and 0.2. The final layer, which also concatenates the one-hot encoding 16 object labels, outputs per point class prediction for 50 parts of size $B \times N \times 50$. The segmentation network takes 2048 points as the input and is trained with a batch size of 20. The learning rate is set to 0.001 with a decay rate of 0.7 every 12 epochs.

2) Results: Table I shows the results of the performance of our network on part segmentation on ShapeNet dataset [13], in which the comparisons with some other methods are also given. Performances are measured by mean Intersection-over-Union (IoU). The instance mIoU indicates the average IoU across all individual instances, while per-class mIoU is calculated by averaging the IoU of all instances within a category. Class mIoU is the average over all categories. The qualitative results are shown in Figure 4.

C. Network Efficiency

The efficiency of our network is evaluated by counting the trainable parameters in the network. The number of parameters of a network is an important factor in evaluating the network complexity and efficiency. Table III shows a comparison of number of parameters and inference accuracy of some recent methods. It can be noted that our model achieves a competitive result while keeping a reasonably simple network structure.

D. Ablation Study

Using the proposed network as a baseline, we perform an ablation study on the network components. In Table IV, the effectiveness of each component is summarized, namely Dense Connection (DC), Residual shortcut (Res), Bottleneck ratio (B/r) and neighbor searching method for the SM layer. '-' indicates such component is not implemented, 'X' indicates the opposite. Seven network configurations are compared. Network A uses the spherical convolution with a classic CNN structure. Network B embeds the SM modules to a DenseNet [7] inspired structure. Network C adds the residual shortcut to network B. The bottleneck layer is introduced to Network D, E and F, with a bottleneck ratio of 8 , 4 and 2 respectively. Network F finds neighboring points through a fixed radius search for the SM layers.

It is noticeable that the dense connection structure gives a 0.7% boost to the accuracy compared to Network A. The residual shortcut across each SM layer contributes extra 0.5%. The effectiveness of the bottleneck layer largely depends on the bottleneck ratio. Although narrowing the feature map width, a bottleneck layer with bottleneck ratio 2 results in a lighter network as well as an increase of 0.4% in accuracy (Network C: 92.8%; Baseline: 93.2%). It should be noted that shortening the feature-map is not always beneficial. Information might be insufficient if the bottleneck ratio is too high (network D and E). As mentioned in Section III-B, a fixed-radius neighbor search might not be specific enough for semantics mining. Evidence can be seen from the 0.7% drop of network F compared to the baseline network. More importantly, kNN generates sorted neighbors, which makes the subsequent convolution more effective.

V. CONCLUSIONS

In this paper, a deep learning architecture for direct point cloud analysis is introduced. Our network deepens the current shallow networks by adopting the dense connection and residual structure, alleviating the vanish gradient problem and facilitating the feature propagation. Additionally, the pooling layers utilize a light-weight MLP for encoding low-level relations, while the semantics mining layers efficiently aggregate high-dimensional features through convolutions on the ordered spherical partitions of the neighborhood to further reduce the number of parameters. This architecture shows its effectiveness on the ModelNet40 and ShapeNet datasets by achieving promising results on object recognition and part segmentation.

To extend this work, one direction is to impose a more general encoder for the inference on real-world data. This may involve dealing with occlusions and translation invariance of objects. Another direction is to apply this work to the subsequent tasks such as 3D point cloud retrieval and object detection in the autonomous driving context.

REFERENCES

[1] C. R. Qi, H. Su, K. Mo, and L. 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, pp. 652–660, 2017.

[2] V. A. Sindagi, Y. Zhou, and O. Tuzel, "Mvx-net: Multimodal voxelnet for 3d object detection," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 7276–7282, IEEE, 2019.

[3] J. Razlaw, J. Quenzel, and S. Behnke, "Detection and tracking of small objects in sparse 3d laser range data," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 2967–2973, IEEE, 2019.

[4] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4758–4765, 2018.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[7] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," *arXiv preprint arXiv:1608.06993*, vol. 1608, 2018.

[8] Y. Feng, Z. Zhang, X. Zhao, R. Ji, and Y. Gao, "Gvcnn: Group-view convolutional neural networks for 3d shape recognition," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 264–272, 2018.

[9] H. Guo, J. Wang, Y. Gao, J. Li, and H. Lu, "Multi-view 3d object retrieval with deep embedding network," *IEEE Transactions on Image Processing*, vol. 25, pp. 5526–5537, 2016.

[10] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view cnns for object classification on 3d data," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5648–5656, 2016.

[11] M. Atzmon, H. Maron, and Y. Lipman, "Point convolutional neural networks by extension operators," *arXiv preprint arXiv:1803.10091*, 2018.

[12] S. M. Ahmed, P. Liang, and C. M. Chew, "Epn: Edge-aware pointnet for object recognition from multi-view 2.5d point clouds," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3445–3450, 2019.

[13] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[14] Y. Shen, C. Feng, Y. Yang, and D. Tian, "Mining point cloud local structures by kernel correlation and graph pooling," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4548–4557, 2018.

[15] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in neural information processing systems*, pp. 5099–5108, 2017.

[16] J. Li, B. M. Chen, and G. Hee Lee, "So-net: Self-organizing network for point cloud analysis," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9397–9406, 2018.

[17] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *Acm Transactions On Graphics (tog)*, vol. 38, no. 5, pp. 1–12, 2019.

[18] A. Komarichev, Z. Zhong, and J. Hua, "A-cnn: Annularly convolutional neural networks on point clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7421–7430, 2019.

[19] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "Kpconv: Flexible and deformable convolution for point clouds," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6411–6420, 2019.

[20] Z. Zhang, B.-S. Hua, and S.-K. Yeung, "Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1607–1616, 2019.

[21] M. Jiang, Y. Wu, T. Zhao, Z. Zhao, and C. Lu, "Pointsift: A sift-like network module for 3d point cloud semantic segmentation," *arXiv preprint arXiv:1807.00652*, 2018.

[22] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920, 2015.

[23] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese, "Joint 2d-3d-semantic data for indoor scene understanding," *arXiv preprint arXiv:1702.01105*, 2017.

[24] T. Hackel, N. Savinov, L. Ladicky, J. D. Wegner, K. Schindler, and M. Pollefeys, "Semantic3d. net: A new large-scale point cloud classification benchmark," vol. IV-1-W1, pp. 91–98, 2017.

[25] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proc. ICCV*, 2015.

[26] Z. Han, M. Shang, Z. Liu, C. Vong, Y. Liu, M. Zwicker, J. Han, and C. L. P. Chen, "Seqviews2seqlabels: Learning 3d global features via aggregating sequential views by rnn with attention," *IEEE Transactions on Image Processing*, vol. 28, no. 2, pp. 658–672, 2019.

[27] H. Huang, E. Kalogerakis, S. Chaudhuri, D. Ceylan, V. G. Kim, and E. Yumer, "Learning local shape descriptors from part correspondences with multiview convolutional networks," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 1, pp. 1–14, 2017.

[28] J. Xie, G. Dai, F. Zhu, E. K. Wong, and Y. Fang, "Deepshape: Deep-learned shape descriptor for 3d shape retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 7, pp. 1335–1345, 2017.

[29] G. Riegler, A. Osman Ulusoy, and A. Geiger, "Octnet: Learning deep 3d representations at high resolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3577–3586, 2017.

[30] M. Tatarchenko, A. Dosovitskiy, and T. Brox, "Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2088–2096, 2017.

[31] Q. Huang, W. Wang, and U. Neumann, "Recurrent slice networks for 3d segmentation of point clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2626–2635, 2018.

[32] B.-S. Hua, M.-K. Tran, and S.-K. Yeung, "Pointwise convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 984–993, 2018.

[33] Y. Liu, B. Fan, G. Meng, J. Lu, S. Xiang, and C. Pan, "Densepoint: Learning densely contextual representation for efficient point cloud processing," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5239–5248, 2019.

[34] Y. Liu, B. Fan, S. Xiang, and C. Pan, "Relation-shape convolutional neural network for point cloud analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8895–8904, 2019.

[35] A. Boulch, "Convpoint: Continuous convolutions for point cloud processing," *Computers & Graphics*, 2020.

[36] H. Lei, N. Akhtar, and A. Mian, "Octree guided cnn with spherical kernels for 3d point clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9631–9640, 2019.

[37] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "Pointcnn: Convolution on x-transformed points," in *Advances in Neural Information Processing Systems 31*, pp. 820–830, 2018.

[38] H. Lei, N. Akhtar, and A. Mian, "Spherical convolutional neural network for 3d point clouds," *arXiv preprint arXiv:1805.07872*, 2018.

[39] H. Lei, N. Akhtar, and A. Mian, "Spherical kernel for efficient graph convolution on 3d point clouds," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[40] C. Esteves, C. Allen-Blanchette, A. Makadia, and K. Daniilidis, "Learning so (3) equivariant representations with spherical cnns," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 52–68, 2018.

[41] R. Klokov and V. 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*, pp. 863–872, 2017.

[42] Y. Ben-Shabat, M. Lindenbaum, and A. Fischer, "3dmfv: Three-dimensional point cloud classification in real-time using convolutional neural networks," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3145–3152, 2018.

[43] Z. H. Lin, S. Y. Huang, and Y. C. F. Wang, "Convolution in the cloud: Learning deformable kernels in 3d graph convolution networks for point cloud analysis," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1797–1806, 2020.