Point2Sequence: Learning the Shape Representation of 3D Point Clouds with an Attention-Based Sequence to Sequence Network

Xinhai Liu,[1] **Zhizhong Han,**[1,2] **Yu-Shen Liu,**[1*] **Matthias Zwicker**[2]
[1]School of Software, Tsinghua University, Beijing 100084, China
Beijing National Research Center for Information Science and Technology (BNRist)
[2]Department of Computer Science, University of Maryland, College Park, USA
lxh17@mails.tsinghua.edu.cn, h312h@umd.edu, liuyushen@tsinghua.edu.cn, zwicker@cs.umd.edu

Abstract

Exploring contextual information in the local region is important for shape understanding and analysis. Existing studies often employ hand-crafted or explicit ways to encode contextual information of local regions. However, it is hard to capture fine-grained contextual information in hand-crafted or explicit manners, such as the correlation between different areas in a local region, which limits the discriminative ability of learned features. To resolve this issue, we propose a novel deep learning model for 3D point clouds, named Point2Sequence, to learn 3D shape features by capturing fine-grained contextual information in a novel implicit way. Point2Sequence employs a novel sequence learning model for point clouds to capture the correlations by aggregating multi-scale areas of each local region with attention. Specifically, Point2Sequence first learns the feature of each area scale in a local region. Then, it captures the correlation between area scales in the process of aggregating all area scales using a recurrent neural network (RNN) based encoder-decoder structure, where an attention mechanism is proposed to highlight the importance of different area scales. Experimental results show that Point2Sequence achieves state-of-the-art performance in shape classification and segmentation tasks.

Introduction

3D point clouds, also called point sets, are considered as one of the simplest 3D shape representations, since they are composed of only raw coordinates in 3D space. A point cloud can be acquired expediently by popular sensors such as LiDAR, conventional cameras, or RGB-D cameras. Furthermore, this kind of 3D data is widely used in 3D modeling (Golovinskiy, Kim, and Funkhouser 2009), autonomous driving (Qi et al. 2017a), indoor navigation (Zhu et al. 2017) and robotics (Wang and Posner 2015). However, learning features or shape representations based on point clouds by deep learning models remains a challenging problem due to the irregular nature of point clouds (Qi et al. 2017b).

As a pioneering approach, PointNet (Qi et al. 2017b) resolves this challenge by directly applying deep learning on point sets. PointNet individually computes a feature of each point and then aggregates all the point features into a global feature by a pooling layer. This leads to PointNet being limited by capturing contextual information of local regions. Attempting to address this issue, several researches take the aggregation of local regions into consideration. KC-Net (Shen et al. 2018) employs a kernel correlation layer and a graph-based pooling layer to capture the local information of point clouds. SO-Net (Li, Chen, and Lee 2018) and DGCNN (Wang et al. 2018) further explore the local structures by building k-nearest neighbors (kNN) graphs and integrating neighbors of a given point using learnable edge attributes in the graph. PointNet++ (Qi et al. 2017c) first extracts features for multi-scale local regions individually and aggregates these features by concatenation, where the two steps are repeated to complete the hierarchical feature extraction. Similarly, ShapeContextNet (Xie et al. 2018) segments the local region of a given point into small bins, then extracts the feature for each bin individually, and finally concatenates the features of all bins as the updated feature for the point. However, most of these previous methods employ hand-crafted or explicit ways for encoding contextual information in local regions, which makes it hard to fully capture fine-grained contextual information, such as the correlation between different areas in the feature space. However, the correlation between different areas in a local region is an important contextual information. Fully exploiting this information might enhance the discriminability of learned features and improve the performance in shape analysis tasks.

We address this issue by proposing a novel deep learning model for 3D point clouds, called Point2Sequence, to encode fine-grained contextual information in local regions in a novel implicit way. Point2Sequence employs a novel RNN-based sequence model for local regions in point clouds to capture the correlation by aggregating multi-scale areas with attention. Specifically, each local region is first separated into multi-scale areas. Then, the feature of each area scale is extracted by a shared Multi-Layer-Perceptron (MLP) layer. Finally, our novel encoder-decoder based sequence model aggregates the features of all area scales, where an attention mechanism is involved to highlight the importance of different scale areas. Experimental results show that Point2Sequence is able to learn more discriminative features from point clouds than existing methods in shape classification and part segmentation tasks.

Our contributions are summarized as follows.

*Corresponding author: Yu-Shen Liu

- We propose Point2Sequence to learn features from point clouds by capturing correlations between different areas in a local region, which takes full advantage of encoding fine-grained contextual information in local regions.
- We introduce an attention mechanism to highlight the importance of different scale areas, and the feature extraction of local regions is enhanced by leveraging the correlation between different area scales.
- To the best of our knowledge, Point2Sequence is the first RNN-based model for capturing correlations between different areas in local regions of point clouds, and our outperforming results verify the feasibility of RNNs to effectively understand point clouds.

Related Work

Learning from point clouds by rasterization. As an irregular type of 3D data, it is intuitive to rasterize the point clouds into uniform sparse 3D grids and then apply volumetric convolutional neural networks. Some approaches (Wu et al. 2015; Maturana and Scherer 2015) represent each voxel with a binary representation which indicates whether it is occupied in the space. Nevertheless, the performance is largely limited by the time and memory consuming due to the data sparsity of 3D shapes. Several improvements (Li et al. 2016; Wang and Posner 2015) have been proposed to relief the sparsity problem of the volumetric representation. However, the sparsity of 3D shapes is an inherent drawback, which still makes it difficult to process very large point clouds. Some recent methods (Su et al. 2015; Wang, Pelillo, and Siddiqi 2017) have tried to project the 3D point clouds or 3D shapes into 2D views and then apply 2D CNNs to recognize them. Influenced by the great success of 2D CNNs for images, such methods have achieved dominant results in 3D shape classification and retrieval tasks (Kanezaki, Matsushita, and Nishida 2016). Due to the lack of depth information, it is nontrivial to extend view-based methods to per-point processing tasks such as point classification and shape classification.

Compared with uniform 3D grids, some latest studies utilize more scalable indexing techniques such as kd-tree and octree to generate regular structures which can facilitate the use of deep learning functions. To enable 3D convolutional networks, OctNet (Riegler, Ulusoy, and Geiger 2017) build a hierarchically partition of the space by generating a set of unbalanced octrees in the regular grids, where each leaf node stores a pooled feature representation. Kd-Net (Klokov and Lempitsky 2017) performs multiplicative transformations according to the subdivisions of point clouds based on the kd-trees. However, in order to obtain rotation invariant of shapes, these methods usually require extra operations such as pre-alignment or excessive data argumentations.

Different from the above-mentioned methods, our method directly learns from point clouds without pre-alignment and voxelization.

Learning from point clouds directly. As a pioneer, PointNet (Qi et al. 2017b) achieves satisfactory performance by directly applying deep learning methods on point sets.

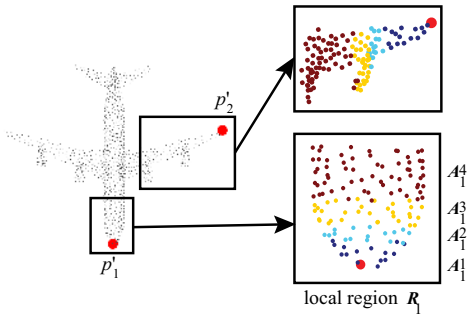

Figure 1: The *left* is an airplane point cloud with two sampled centroids p'_1 and p'_2 in the red color. The *right* is the corresponding local regions of p'_1 (below) and p'_2 (above), where different colors represent different area scales within the local region. For example, there are four different scale areas $\{\boldsymbol{A}^1_1, \boldsymbol{A}^2_1, \boldsymbol{A}^3_1, \boldsymbol{A}^4_1\}$ in the local region $\boldsymbol{R}_1$ centered by p'_1.

PointNet individually computes the feature of each point and then aggregates all the point features into a global feature by pooling. This leads to PointNet limited by capturing contextual information in local regions. Some enhancements are proposed to address this problem by combining contextual information in local regions by hand-crafted or explicit ways. PointNet++ (Qi et al. 2017c) has been proposed to group points into several clusters in pyramid-like layers, where the feature of multi-scale local regions can be extracted hierarchically. PointCNN (Li et al. 2018) and SpiderCNN (Xu et al. 2018) investigate the convolution-like operations which aggregate the neighbors of a given point by edge attributes in the local region graph. However, with hand-crafted or explicit ways of encoding contextual information in local regions, it is hard for these methods to capture fine-grained contextual information. In particular, the correlation between different areas in feature space is an important contextual information, which limits the discriminative ability of learned features for point cloud understanding.

Correlation learning with RNN-based models. To aggregate sequential feature vectors, recurrent neural networks (Elman 1990) have shown preeminent performance in many popular tasks such as speech recognition (Li and Wu 2015) or handwriting recognition (Bertolami et al. 2009). Inspired by the sequence to sequence architecture (Cho et al. 2014), RNN-based seq2seq models can capture the fine-grained contextual information from the input sequence and effectively convert it to another sequence. Furthermore, Sutskever, Vinyals, and Le (2014) engages an attention mechanism that intuitively allows neural networks to focus on different parts of the input sequences, which is more in line with the actual situation. In addition, several kinds of attention mechanism (Bahdanau, Cho, and Bengio 2014; Luong, Pham, and Manning 2015) have been presented to enhance the performance of networks in machine translation (MT). To utilize the powerful ability of correla-

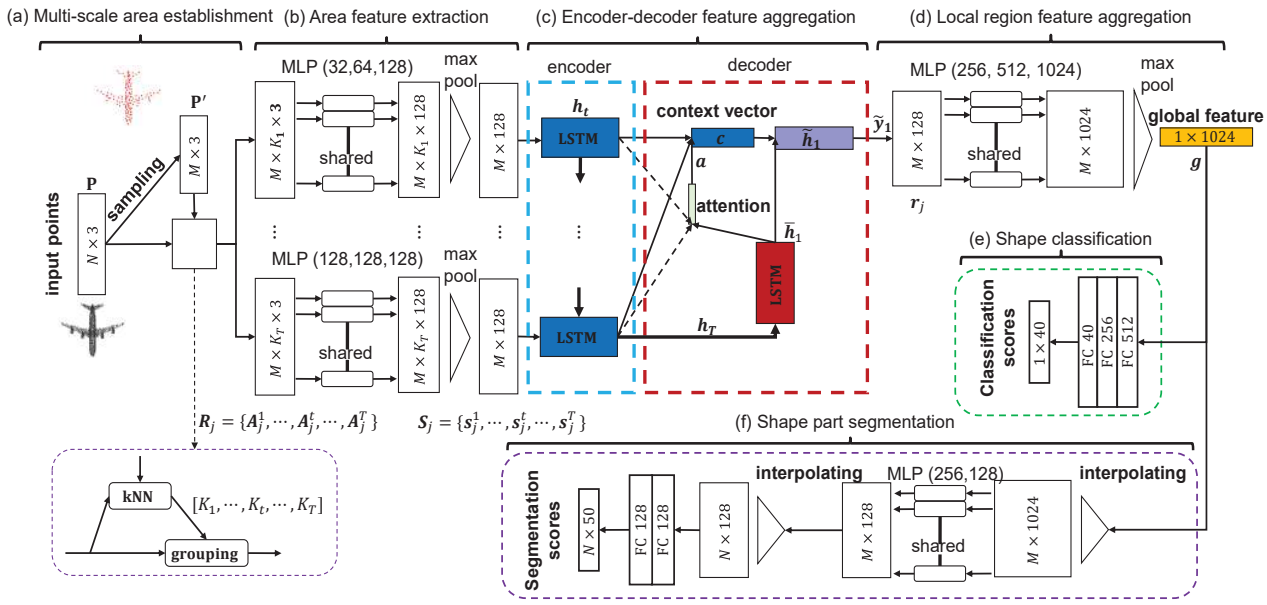

Figure 2: Our Point2Sequence architecture. Point2Sequence first samples local regions from an input point cloud and establishes multi-scale areas in each local region in (a). Then, MLP layer is employed to extract the feature of each multi-scale area in (b). Subsequently, the feature of each local region is extracted by attention-based seq2seq structure in (c). Finally, the global feature of the point cloud is obtained by aggregating the features of all sampled local regions in (d). The learned global feature can be used not only for shape classification shown in (e) but also for part segmentation with some extension network shown in (f).

tion learning from RNN-based sequence to sequence structure, Point2Sequence adopts a seq2seq encoder-decoder architecture to learn the correlation of different areas in each local region with attention.

The Point2Sequence Model

In this section, we first overview our Point2Sequence model, and then detail the model including multi-scale area establishment, multi-scale area feature extraction, attention-based sequence to sequence structure and model adjustments for segmentation.

Overview

Figure 2 illustrates our Point2Sequence architecture. Our model is formed by six parts: (a) Multi-scale area establishment, (b) Area feature extraction, (c) Encoder-decoder feature aggregation, (d) Local region feature aggregation, (e) Shape classification and (f) Shape part segmentation. The input of our network is a raw point set $\mathbf{P} = \{p_i \in \mathbb{R}^3, i = 1, 2, \cdots, N\}$. We first select M points $\mathbf{P'} = \{p'_j \in \mathbb{R}^3, j = 1, 2, \cdots, M\}$ from $\mathbf{P}$ to define the centroids of local regions $\{\boldsymbol{R}_1, \cdots, \boldsymbol{R}_j, \cdots, \boldsymbol{R}_M\}$. As illustrated in Figure 1, T different scale areas $\{\boldsymbol{A}_j^1, \cdots, \boldsymbol{A}_j^t, \cdots, \boldsymbol{A}_j^T\}$ in each local region $\boldsymbol{R}_j$ centered by p'_j are established in the multi-scale area establishment part, where T different scale areas contain $[K_1, \cdots, K_t, \cdots, K_T]$ points, respectively. We then extract a D-dimensional feature $\boldsymbol{s}_j^t$ for each scale area $\boldsymbol{A}_j^t$ through the scale feature extraction part. In each local region $\boldsymbol{R}_j$, a feature sequence $\boldsymbol{S}_j = \{\boldsymbol{s}_j^1, \cdots, \boldsymbol{s}_j^t, \cdots, \boldsymbol{s}_j^T\}$ of multi-scale areas is aggregated into a D-dimensional feature vector $\boldsymbol{r}_j$ by the encoder-decoder feature aggregation part. Finally, a 1024-dimensional global feature $\boldsymbol{g}$ is aggregated from the features $\boldsymbol{r}_j$ of all local regions by the local region feature aggregation part. Our model can be trained for shape classification or shape part segmentation by minimizing the error according to the ground truth class labels or per point part labels.

Multi-scale Area Establishment

Similar to PointNet++ (Qi et al. 2017c) and ShapeContextNet (Xie et al. 2018), we build the structure of multi-scale areas in each local region on the point cloud. This part is formed by three layers: sampling layer, searching layer and grouping layer. The sampling layer selects M points from the input point cloud as the centroids of local regions. In each local region, searching layer searches the K_t points in the input points and return the corresponding point indexes. According to the point indexes, grouping layer groups the K_t points from $\mathbf{P}$ to form each multi-scale area.

As shown in Figure 2, a certain amount of points are first selected as the centroids of local regions by the sampling layer. We adopt the farthest point sampling (FPS) to iteratively select M ($M < N$) points. The new added point p'_j is always the farthest point from points $\{p'_1, p'_2, \cdots, p'_{j-1}\}$ in the 3D space. Although random sampling is an optional choice, the FPS can achieve a more uniform coverage of the entire point cloud in the case of the same centroids.

Then the searching layer respectively finds the top

$[K_1, \cdots, K_t, \cdots, K_T]$ nearest neighbors for each local region $\boldsymbol{R}_j$ from the input points and returns the corresponding indexes of these points. In the searching layer, we adopt kNN to find the neighbors of centroids according to the sorted Euclidean distance in the 3D space. Another optional search method is ball search (Qi et al. 2017c) which selects all points within a radius around the centroid. Compared with ball search, kNN search guarantees the size of local regions and makes it insensitive to the sampling density of point clouds.

Finally, by the grouping layer, these indexes of points are used to extract the points in each area of local region $\boldsymbol{R}_j$, where we obtain the points with the size of $M \times K_t \times 3$ in the scale area $\boldsymbol{A}_j^t$ of all local regions.

Multi-scale Area Feature Extraction

To extract the feature for each multi-scale area, a simple and effective MLP layer is employed in our network. The MLP layer first abstracts the points in each scale area $\boldsymbol{A}_j^t$ into the feature space and then the point features are aggregated into a D-dimensional feature $\boldsymbol{s}_j^t$ by max pooling. Therefore, in each local region $\boldsymbol{R}_j$, a $T \times D$ feature sequence $\boldsymbol{S}_j$ of the multi-scale areas is acquired.

In addition, similar to prior studies such as SO-Net (Li, Chen, and Lee 2018), the coordinates of points in each area $\boldsymbol{A}_j^t$ are converted to the relative coordinate system of the centroid p_j' by a subtraction operation: $p_l = p_l - p_j'$, where l is the index of point in the area. By using the relative coordinate, the learned features of point sets can be invariant to transformations such as rotation and translation, which is crucial for improving the learning ability of networks. Moreover, we also combine the area feature $\boldsymbol{s}_j^t$ with the coordinates of area centroid p_j' to enhance the association between them.

Attention-based Sequence to Sequence Structure

In this subsection, we propose an attention-based encoder-decoder network to capture the fine-grained contextual information of local regions. Through the multi-scale area feature extraction, each local region $\boldsymbol{R}_j$ is abstracted into a feature sequence $\boldsymbol{r}_j$. To learn from the feature sequence, we adopt a RNN-based model to aggregate the features $\boldsymbol{S}_j$ of size $T \times D$ into a D dimension feature vector $\boldsymbol{s}_j$ in each local region by an implicit way. To further promote the correlation learning between different areas in local regions, we employ an attention mechanism to focus on items of the feature sequence.

Multi-scale area encoder. To learn the correlation between different areas, a RNN is employed as encoder to integrate the features of multi-scale areas in a local region. The RNN encoder is consisted by a hidden state $\mathbf{h}$ and an optional output $\mathbf{y}$, which operates on the input feature sequence $\boldsymbol{S}_j = \{\boldsymbol{s}_j^1, \ldots, \boldsymbol{s}_j^t, \ldots, \boldsymbol{s}_j^T\}$ of multi-scale areas in each local region. Each item $\boldsymbol{s}_j^t$ of the sequence $\boldsymbol{S}_j$ is a D-dimensional feature vector and the length of $\boldsymbol{S}_j$ is T which is also the steps of the encoder. At each time step t, the hidden state $\mathbf{h}_t$ of the RNN encoder is updated by

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \boldsymbol{s}_j^t), \tag{1}$$

where f is a non-linear activation function which can be a long short-term memory (LSTM) unit (Hochreiter and Schmidhuber 1997) or a gated recurrent unit (Cho et al. 2014).

A RNN can learn the probability distribution over a sequence by being trained to predict the next item in the sequence. Similarly, at time t, the output $\mathbf{y}_t$ of the encoder can be represented as

$$\mathbf{y}_t = \mathbf{W}_a \mathbf{h}_t, \tag{2}$$

where $\mathbf{W}_a$ is a learnable weight matrix.

After forwarding the entire input feature sequence at step T, the last-step hidden state $\mathbf{h}_T$ of the encoder is acquired, which contains the context information of the entire input sequence.

Local region feature decoder. To obtain the feature $\boldsymbol{r}_j$ for each local region $\boldsymbol{R}_j$, we employ a RNN decoder to translate the contextual information from the multi-scale areas in $\boldsymbol{R}_j$. Different from the models in machine translation, there is no decoding target for the decoder in our case. To address this issue, we employ $\mathbf{h}_T$ as the decoding target which contains contextual information of the entire input feature sequence. Therefore, a one-step decoding process is adopt in our network to decode the feature $\boldsymbol{r}_j$ for each local region $\boldsymbol{R}_j$. Similar to the encoder, the decoder is also consisted by a hidden state $\bar{\mathbf{h}}$ and output $\bar{\mathbf{y}}$. We initialize the $\bar{\mathbf{h}}_0$ with a zero state $\mathbf{z}_0$ and the current hidden state of the decoder at step one can be updated by

$$\bar{\mathbf{h}}_1 = f(\mathbf{z}_0, \mathbf{h}_T), \tag{3}$$

where f is an activation function as shown in Eq. (1). Similarly, the output of decoder $\bar{\mathbf{y}}_1$ is computed by

$$\bar{\mathbf{y}}_1 = \mathbf{W}_b \bar{\mathbf{h}}_1, \tag{4}$$

where $\mathbf{W}_b$ is a learnable matrix in the training.

To further enhance the decoding of the contextual information in the input feature sequence $\boldsymbol{S}_j$, a context vector $\mathbf{c}$ is generated to help the predict of feature $\tilde{\mathbf{y}}_1$ of local region with attention. Therefore, a new hidden state $\tilde{\mathbf{h}}_\mathbf{1}$ and output $\tilde{\mathbf{y}}_1$ are computed in the decoder as introduced later, where we employ the output $\tilde{\mathbf{y}}_1$ to be the feature $\boldsymbol{r}_j$ for each local region.

Attention mechanism in decoder. Inspired by the thought of focusing on parts of the source sentence in the machine translation, we adopt an attention mechanism to highlight the importance of different areas in each local region. The goal is to utilize the context vector $\mathbf{c}$ which is generated by

$$\mathbf{c} = \sum\nolimits_{t=1}^{T} \boldsymbol{\alpha}(t) \mathbf{h}_t, \tag{5}$$

where $\boldsymbol{\alpha}$ is the attention vector and t is the time step.

The idea of our attention model is to consider all the hidden states of the encoder when generating the context vector $\mathbf{c}$. In this model, a fixed-length attention vector $\boldsymbol{\alpha}$, whose size is equal to the sequence length T of the input side, is

Table 1: The shape classification accuracy (%) comparison on ModelNet10 and ModelNet40.

Method	Input	ModelNet10		ModelNet40	
		Class	Instance	Class	Instance
PointNet (Qi et al. 2017b)	1024×3	-	-	86.2	89.2
PointNet++ (Qi et al. 2017c)	1024×3	-	-	-	90.7
ShapeContextNet (Xie et al. 2018)	1024×3	-	-	87.6	90.0
Kd-Net (Klokov and Lempitsky 2017)	$2^{15} \times 3$	93.5	94.0	88.5	91.8
KC-Net (Shen et al. 2018)	1024×3	-	94.4	-	91.0
PointCNN (Li et al. 2018)	1024×3	-	-	-	91.7
DGCNN (Wang et al. 2018)	1024×3	-	-	90.2	92.2
SO-Net (Li, Chen, and Lee 2018)	2048×3	93.9	94.1	87.3	90.9
Ours	1024×3	**95.1**	**95.3**	**90.4**	**92.6**

Table 2: The accuracies (%) of part segmentation on ShapeNet part segmentation dataset.

	mean	Intersection over Union (IoU)															
		air.	bag	cap	car	cha.	ear.	gui.	kni.	lam.	lap.	mot.	mug	pis.	roc.	ska.	tab.
# SHAPES		2690	76	55	898	3758	69	787	392	1547	451	202	184	283	66	152	5271
PointNet	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
PointNet++	85.1	82.4	79.0	87.7	77.3	90.8	71.8	91.0	85.9	83.7	95.3	**71.6**	94.1	81.3	58.7	**76.4**	82.6
ShapeContextNet	84.6	83.8	80.8	83.5	**79.3**	90.5	69.8	**91.7**	86.5	82.9	**96.0**	69.2	93.8	82.5	**62.9**	74.4	80.8
Kd-Net	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
KCNet	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	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
SO-Net	84.9	82.8	77.8	**88.0**	77.3	90.6	73.5	90.7	83.9	82.8	94.8	69.1	94.2	80.9	53.1	72.9	**83.0**
Ours	**85.2**	82.6	81.8	87.5	77.3	90.8	77.1	91.1	86.9	83.9	95.7	70.8	**94.6**	79.3	58.1	75.2	82.8

generated by comparing the current hidden state $\bar{\mathbf{h}}_\mathbf{1}$ with each source hidden state $\mathbf{h}_t$ as

$$\boldsymbol{\alpha}(t) = \frac{exp(score(\bar{\mathbf{h}}_1, \mathbf{h}_t))}{\sum_{t'=1}^{T} exp(score(\bar{\mathbf{h}}_1, \mathbf{h}_{t'}))}. \quad (6)$$

Here, $score$ is referred as

$$score(\bar{\mathbf{h}}_1, \mathbf{h}_t) = \bar{\mathbf{h}}_1^\top \mathbf{W}_c \mathbf{h}_t, \quad (7)$$

which is a content-based function to show the correlation between these two vectors. Here, $\mathbf{W}_c$ is also a learnable weight matrix in the training.

With the help of Eq. (5), we can acquire the new hidden state $\tilde{\mathbf{h}}_1$ based on the current hidden state $\bar{\mathbf{h}}_1$ in the decoder. Specifically, with the current hidden state $\bar{\mathbf{h}}_1$ and the context vector $\mathbf{c}$, a simple concatenation layer combines the contextual information of the two vectors to generate the attentional new hidden state as follows,

$$\tilde{\mathbf{h}}_1 = tanh(\mathbf{W}_d[\mathbf{c}; \bar{\mathbf{h}}_1]). \quad (8)$$

And the output $\tilde{\mathbf{y}}_1$ is similarly computed by

$$\tilde{\mathbf{y}}_1 = \mathbf{W}_s \tilde{\mathbf{h}}_1, \quad (9)$$

where $\mathbf{W}_d$ and $\mathbf{W}_s$ are variables to be learned.

Our attention-based sequence to sequence structure aggregates the sequential features $\boldsymbol{S}_j$ of multi-scale areas in each local region $\boldsymbol{R}_j$ by an implicit way. So far, the features $\boldsymbol{r}_j$ of all local regions with size of $M \times D$ are acquired. In the subsequent network, a 1024-dimensional global feature $\boldsymbol{g}$ of the input point cloud is extracted from the features of all local regions by the global feature extraction part. As depicted in Figure 2, we apply the global feature $\boldsymbol{g}$ to shape classification and part segmentation tasks.

Model Adjustments for Segmentation

The goal of part segmentation is to predict a semantic label for each point in the point cloud. In Point2Sequence, a global feature of the input point cloud is generated. Therefore, we need to acquire the per-point feature for each point in the point cloud from the global feature. There are two optional implementations, one is to duplicate the global feature with N times (Qi et al. 2017b; Wang et al. 2018), and the other is to perform upsampling by interpolation (Qi et al. 2017c; Li, Chen, and Lee 2018). In this paper, two interpolate layers are equipped in our networks as shown in Figure 2, which propagate the feature from shape level to point level by upsampling. Compared with shape classification, it is a challenge task to distinguish the parts in the object, which requires more fine-grained contextual information of local regions. We implement the feature ϕ propagation according to the Euclidean distance between points in 3D space. The feature is interpolated by inverse square Euclidean distance wighted average based on k nearest neighbors as

$$\phi(p) = \frac{\sum_{i=1}^{k} w(p_i)\phi(p_i)}{\sum_{i=1}^{k} w(p_i)}, \quad (10)$$

where $w(p_i) = \frac{1}{(p-p_i)^2}$ is the inverse square Euclidean distance between p and p_i.

To guide the interpolation process, the interpolated features are concatenated with the corresponding point features in the abstraction side and several MLP layers are equipped in our network to enhance the performance. Several shared fully connected layers and ReLU layers are also applied to promote the extraction of point features, like the branch in shape classification.

Table 3: The effect of the number of sampled points M on ModelNet40.

M	128	256	384	512
Acc (%)	91.86	92.34	**92.54**	91.86

Experiments

In this section, we first investigate how some key parameters affect the performance of Point2Sequence in the shape classification task. Then, we compare our Point2Sequence with several state-of-the-art methods in shape classification on ModelNet10 and ModelNet40 (Wu et al. 2015), respectively. Finally, the performance of our Point2Sequence is evaluated in the part segmentation task on ShapeNet part dataset (Savva et al. 2016).

Ablation Study

Network configuration. In Point2Sequence, we first sample $M = 384$ points as the centroids of local regions by the sampling layer. Then the searching layer and grouping layer select $T = 4$ scale of areas with $[16, 32, 64, 128]$ points in each area of a local region. The points in each area are abstracted into a $D = 128$ dimensional feature by a 3-layer MLP and then these abstracted features are aggregated by max pooling. And the feature sequence of different areas in each local region is aggregated by the RNN-based encoder-decoder structure with attention. Here, we initialize the RNN encoder and decoder with h=128 dimensional hidden state, where LSTM is used as the RNN cell. The rest setting of our network is the same as in Figure 2. In addition, ReLU is used after each fully connected layer with Batch-normalization, and Dropout is also applied with drop ratio 0.4 in the fully connected layers. In the experiment, we trained our network on a NVIDIA GTX 1080Ti GPU using ADAM optimizer with initial learning rate 0.001, batch size of 16 and batch normalization rate 0.5. The learning rate and batch normalization rate are decreased by 0.3 and 0.5 for every 20 epochs, respectively.

Parameters. All the experiments in the ablation study are evaluated on ModelNet40. ModelNet40 contains 12311 CAD models from 40 categories and is split into 9843 for training and 2468 for testing. For each model, we adopt 1024 points which are uniformly sampled from mesh faces and are normalized into a unit ball as input.

We first explore the number of sampled points M which influences the distribution of local regions in point clouds. In the experiment, we keep the settings of our network as depicted in the network configuration section and modifies the number of sampled points M from 128 to 512. The results are shown in Table3, where the instance accuracies on the benchmark of ModelNet40 have a tendency to rise first and then fall. The highest accuracy of 92.54% is reached at $M = 384$ sampled points. This comparison implies that Point2Sequence can extract the contextual information from local regions effectively and $M = 384$ is a optimum number of sampled points to coverage the whole point cloud.

Table 4: The effects of the type of RNN cell (CT) and hidden state dimension h on ModelNet40.

Metric	RT=LSTM	GRU	h=64	256
Acc (%)	**92.54**	92.18	92.46	92.18

Therefore, we employ the sampled points $M = 384$ as the setting of our network in the following experiments. Then, as shown in Table 4, we show the effect of the type of the RNN cell (RT) and the dimension of the RNN hidden state h, respectively. The accuracy degenerates to 92.18% when replacing the LSTM cell as the GRU cell. Based on the setting of $h = 128$, we set h to 64 and 256, which reduce the accuracy of $h = 128$ to 92.46% and 92.18%. The above results suggest that dimension of hidden state $h = 128$ and the type of hidden state RT=LSTM is more suitable for our network.

Table 5: The effects of the attention mechanism (Att) and decoder (Dec) on ModelNet40.

Metric	Att+ED	No Att	No Dec	Con	MP
Acc (%)	**92.54**	92.26	92.42	92.06	91.73

We also discuss the impact of the attention mechanism, the decoder and the encoder-decoder structure on our network. We evaluate the performance of our network without the attention mechanism (No Att), without decoder (No Dec) and without encoder-decoder structure (No ED). In No ED, we remove the encoder-decoder from our network and aggregate the features of multi-scale areas in each local region by concatenating (Con) or max pooling (MP). In Table 5, the result of the attention mechanism with encoder-decoder (Att+ED) is better than the results of removing parts of the attention mechanism and encoder-decoder. This comparison shows that the decoder works better with the attention mechanism, and the decoder without attention will decrease the capability of the network. And our RNN-based sequence to sequence structure outperforms hand-crafted manners such as concatenate (Con) and max pooling (MP) in aggregating feature of multi-scale areas.

Table 6: The effect of the number of scales T on ModelNet40.

T	4	3	2	1
Acc (%)	92.54	92.46	**92.63**	91.94

Moreover, we reduce the scale of local regions T from 4 to 1 and remains the largest scale with 128 points. As depicted in Table 6, we obtain a even better result of 92.62% with $T = 2$ ($K_1 = 64, K_2 = 128$). The results with $T > 1$ are better than the result with $T = 1$, which shows that the strategy of multi-scale areas can be better in capturing contextual information from local regions. Therefore, the number of areas T affects the performance of Point2Sequence in extracting the information from local regions.

Finally, based on the setting of multi-scale areas $T = 2$,

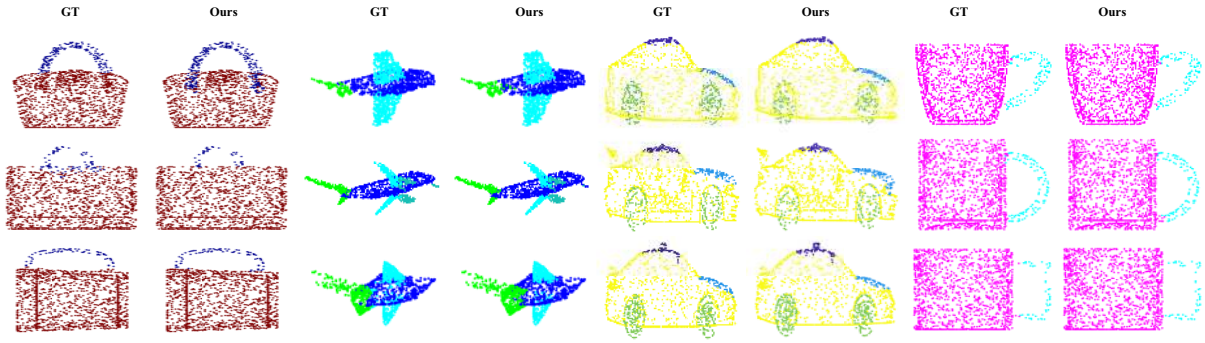

Figure 3: Visualization of part segmentation results. In each shape pair, first column is the ground truth (GT), and second column is our predicted result, where parts with the same color have a consistent meaning. From left to right: bag, airplane, car, and cup.

Table 7: The effect of learning rate on ModelNet40.

LR	0.0005	0.001	0.002
Acc (%)	91.94	**92.63**	91.86

we explore the effect of learning rate (LR) by setting it to 0.0005 and 0.002. As shown in Table 7, the highest accuracy is reached at LR = 0.001. Therefore, LR = 0.001 is the optimal choice of Point2Sequence.

Shape Classification

In this subsection, we evaluate the performance of Point2Sequence on ModelNet10 and ModelNet40 benchmarks, where ModelNet10 contains 4899 CAD models which is split into 3991 for training and 908 for testing. Table 1 compares Point2Sequence with the state-of-the-art methods in the shape classification task on ModelNet10 and ModelNet40. We compare our method with the results of eight recently ranked methods on each benchmark in terms of class average accuracy and instance average accuracy. For fair comparison, all the results in Table 1 are obtained under the same condition, which handles with raw point sets without the normal vector. By optimizing the cross entropy loss function in the training process, on both benchmarks, Point2Sequence outperforms other methods in class average accuracies and instance average accuracies. In ModelNet40 shape classification, our method achieves the instance accuracy of 92.6% which is 1.9% and 0.2% higher than PointNet++ (Qi et al. 2017c) and DGCNN (Wang et al. 2018), respectively. Experimental results show that Point2Sequence outperforms other methods by extracting the contextual information of local regions.

Part Segmentation

To further validate that our approach is qualified for point cloud analysis, we evaluate Point2Sequence on the semantic part segmentation task. The goal of part segmentation is to predict semantic part label for each point of the input point cloud. As dipected in Figure 2, we build the part segmentation branch to implement the per-point classification.

In part segmentation, ShapeNet part dataset is used as our benchmark for the part segmentation task, the dataset contains 16881 models from 16 categories and is spit into train set, validation set and test set following PointNet++. There are 2048 points sampled from each 3D shape, where each point in a point cloud object belongs to a certain one of 50 part classes and each point cloud contains 2 to 5 parts.

We employ the mean Intersection over Union (IoU) proposed in (Qi et al. 2017b) as the evaluation metric. For each shape, the IoU is computed between groundtruth and the prediction for each part type in the shape category. To calculate the mIoU for each shape category, we compute the average of all shape mIoUs in the shape category. Overall mIoU is also calculated as the average mIoUs over all test shapes. Similar to the shape classification task, we optimized the cross entropy loss in the training process. We compare our results with PointNet (Qi et al. 2017b), PointNet++ (Qi et al. 2017c), Kd-Net (Klokov and Lempitsky 2017), SO-Net (Li, Chen, and Lee 2018), KC-Net (Shen et al. 2018), ShapeContextNet (Xie et al. 2018) and DGCNN (Wang et al. 2018). In Table 2, we report the performance of Point2Sequence in each category and the mean IoU of all testing shapes. Compared with the stated-of-the-art methods, Point2Sequence acquires the best mean instance IoU of 85.2% and achieves the comparable performances on many categories. Figure 3 visualizes some examples of our predicted results, where our results are highly consistent with the ground truth.

Conclusions

In this paper, we propose a novel representation learning framework for point cloud processing in the shape classification and part segmentation tasks. An attention-based sequence to sequence model is proposed to utilize a sequence of multi-scale areas, which focuses on learning the correlation of different areas in a local region. To enhance the performance, an attention mechanism is adopted to highlight the importance of multi-scale areas in the local region. Experimental results show that our method achieves competitive performances with the state-of-the-art methods.

Acknowledgments

Yu-Shen Liu is the corresponding author. This work was supported by National Key R&D Program of China (2018YFB0505400), the National Natural Science Foundation of China (61472202), and Swiss National Science Foundation grant (169151). We thank all anonymous reviewers for their constructive comments.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473*.

Bertolami, R.; Bunke, H.; Fernandez, S.; Graves, A.; Liwicki, M.; and Schmidhuber, J. 2009. A novel connectionist system for improved unconstrained handwriting recognition. *TPAMI*.

Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv:1406.1078*.

Elman, J. L. 1990. Finding structure in time. *Cognitive Science* 14(2):179–211.

Golovinskiy, A.; Kim, V. G.; and Funkhouser, T. 2009. Shape-based recognition of 3D point clouds in urban environments. In *ICCV*, 2154–2161.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780.

Kanezaki, A.; Matsushita, Y.; and Nishida, Y. 2016. RotationNet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. *arXiv:1603.06208*.

Klokov, R., and Lempitsky, V. 2017. Escape from cells: Deep kd-networks for the recognition of 3D point cloud models. In *ICCV*, 863–872.

Li, X., and Wu, X. 2015. Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition. In *ICASSP*, 4520–4524.

Li, Y.; Pirk, S.; Su, H.; Qi, C. R.; and Guibas, L. J. 2016. FPNN: Field probing neural networks for 3D data. In *NIPS*, 307–315.

Li, Y.; Bu, R.; Sun, M.; and Chen, B. 2018. PointCNN. *arXiv:1801.07791*.

Li, J.; Chen, B. M.; and Lee, G. H. 2018. SO-Net: Self-organizing network for point cloud analysis. In *CVPR*, 9397–9406.

Luong, M.-T.; Pham, H.; and Manning, C. D. 2015. Effective approaches to attention-based neural machine translation. *arXiv:1508.04025*.

Maturana, D., and Scherer, S. 2015. Voxnet: A 3D convolutional neural network for real-time object recognition. In *IROS*, 922–928.

Qi, C. R.; Liu, W.; Wu, C.; Su, H.; and Guibas, L. J. 2017a. Frustum pointnets for 3D object detection from RGB-D data. *arXiv:1711.08488*.

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017b. PointNet: Deep learning on point sets for 3D classification and segmentation. In *CVPR*.

Qi, C. R.; Yi, L.; Su, H.; and Guibas, L. J. 2017c. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 5099–5108.

Riegler, G.; Ulusoy, A. O.; and Geiger, A. 2017. OctNet: Learning deep 3D representations at high resolutions. In *CVPR*.

Savva, M.; Yu, F.; Su, H.; Aono, M.; Chen, B.; Cohen-Or, D.; Deng, W.; Su, H.; Bai, S.; Bai, X.; et al. 2016. Shrec'16 track large-scale 3D shape retrieval from ShapeNet core55. In *Proceedings of the Eurographics Workshop on 3D Object Retrieval*.

Shen, Y.; Feng, C.; Yang, Y.; and Tian, D. 2018. Mining point cloud local structures by kernel correlation and graph pooling. In *CVPR*.

Su, H.; Maji, S.; Kalogerakis, E.; and Learned-Miller, E. 2015. Multi-view convolutional neural networks for 3D shape recognition. In *ICCV*, 945–953.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *NIPS*, 3104–3112.

Wang, D. Z., and Posner, I. 2015. Voting for voting in online point cloud object detection. In *Robotics: Science and Systems*.

Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S. E.; Bronstein, M. M.; and Solomon, J. M. 2018. Dynamic graph CNN for learning on point clouds. *arXiv:1801.07829*.

Wang, C.; Pelillo, M.; and Siddiqi, K. 2017. Dominant set clustering and pooling for multi-view 3D object recognition. In *BMVC*.

Wu, Z.; Song, S.; Khosla, A.; Yu, F.; Zhang, L.; Tang, X.; and Xiao, J. 2015. 3D ShapeNets: A deep representation for volumetric shapes. In *CVPR*, 1912–1920.

Xie, S.; Liu, S.; Chen, Z.; and Tu, Z. 2018. Attentional ShapeContextNet for point cloud recognition. In *CVPR*, 4606–4615.

Xu, Y.; Fan, T.; Xu, M.; Zeng, L.; and Qiao, Y. 2018. SpiderCNN: Deep Learning on Point Sets with Parameterized Convolutional Filters. *arXiv:1803.11527*.

Zhu, Y.; Mottaghi, R.; Kolve, E.; Lim, J. J.; Gupta, A.; Fei-Fei, L.; and Farhadi, A. 2017. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*, 3357–3364.