Hierarchical Point Cloud Encoding and Decoding with Lightweight Self-Attention based Model

En Yen Puang¹, Hao Zhang², Hongyuan Zhu¹ Member, IEEE, and Wei Jing¹,³

Abstract—In this paper we present SA-CNN, a hierarchical and lightweight self-attention based encoding and decoding architecture for representation learning of point cloud data. The proposed SA-CNN introduces convolution and transposed convolution stacks to capture and generate contextual information among unordered 3D points. Following conventional hierarchical pipeline, the encoding process extracts feature in local-to-global manner, while the decoding process generates feature and point cloud in coarse-to-fine, multi-resolution stages. We demonstrate that SA-CNN is capable of a wide range of applications, namely classification, part segmentation, reconstruction, shape retrieval, and unsupervised classification. While achieving the state-of-the-art or comparable performance in the benchmarks, SA-CNN maintains its model complexity several order of magnitude lower than the others. In term of qualitative results, we visualize the multi-stage point cloud reconstructions and latent walks on rigid objects as well as deformable non-rigid human and robot models.

Index Terms—Visual Learning, Deep Learning for Visual Perception, Recognition

I. INTRODUCTION

WITH recent advancement on 3D sensory devices and the surging needs on 3D perception in applications such as the robotics and autonomous vehicles [1], understanding point cloud data from 3D sensors becomes increasingly important. Unlike 2D pixel-based images, 3D point cloud data is unordered and unstructured. Moreover, point clouds are usually in large scale and thus effective representation learning is often desirable for many downstream tasks [2].

Many deep learning based approaches [3] have been applied to 3D point cloud perception. Early work rely on handcrafted 3D features which require intensive feature engineering and are difficult to generalize. Then methods like [4], [5] convert 3D point cloud into voxel representation and extend 2D convolution to 3D, but bear high computational cost and low voxel resolution. Some work [6], [7] explore multi-view projection methods that apply 2D CNN to extract the view-wise features circumvent the heavy computational complexity, but usually requires heuristics in choosing informative viewpoints. Later, [8], [9] propose an efficient and effective end-to-end architecture that directly learns the representations from the unstructured point cloud data.

In term of dimensionality reduction, auto-encoder or encoder-decoder architecture has been widely adopted in compressing and/or distilling input into lower dimensional representation in multiple domains. In the context of point cloud, earlier methods [8], [10], [2] use mostly MLP in feature extraction and generation. Hierarchical pipeline was later introduced [9], [11], [12] to encode/extract feature in local-to-global manner, and to decode/generate point cloud in coarse-to-fine, multi-resolution stages.

To this end, we propose to apply a lightweight hierarchical self-attention mechanism [13], [9] to handle the unstructured, interrelated contextual information of point cloud data. Self-attention models have been widely adopted in natural language processing, vision and point cloud [14], [15] due to their efficiency in handling dependencies in sequential/unordered data. Leveraging on the efficiency and simplicity of the self-attention mechanism, we propose a lightweight point cloud convolution and transposed convolution operator for hierarchical feature extraction and generation on point cloud data. We demonstrate that our models are capable for a wide range of applications and achieve the state-of-the-art or comparable performance while keeping the model complexity low. Hence, the main contributions of this work are summarized as follows:

- we propose a hierarchical network architecture named SA-CNN for point cloud encoding and decoding using...
an lightweight self-attention based model;
- we demonstrate the applications of SA-CNN in classification, part segmentation, reconstruction, shape retrieval and unsupervised classification;
- we show that SA-CNN achieves the state-of-the-art or comparable performances in multiple benchmarks while keeping the model complexity orders of magnitude lower.

II. RELATED WORK

A. Point-based Methods for Point Cloud Analysis

Point-based methods became popular due to its efficiency, flexibility and scalability. Different network architectures e.g., Multi-Layer Perceptron (MLP), point convolution, graph-based, and attention-based network could be used to deal with the point-wise feature extraction. [8], [9] propose PointNet(++) as pioneer work for point-based methods on 3D point cloud data, which directly learn feature representation from the point cloud data. Convolution-based methods have also been applied to point cloud data [16], [17] by directly performing convolution-like operation on 3D points. More recently, graph-based neural networks [18], [19] and attention-based methods [20], [21], [22] also demonstrate good performance for point cloud feature learning. However, these related works only learn features in encoding manner. Our work is a point-based method that uses self-attention mechanism to build hierarchical encoding and decoding models for efficient representation learning of point cloud.

B. Self-Attention Mechanism

Self-attention mechanism has become an integral part of various neural network-based models for handling the sequence inputs, allowing modeling of dependencies without regard to their distance in the sequences. Specifically, the Transformers [13] have revolutionized natural language processing tasks due to their tremendous ability on representation learning. Recently, there are several work [14], [15] adopt the attention mechanism to address the 3D point cloud problems. [23], [24], [25] proposed additional mechanism together with self-attention to improve point sampling and feature aggregation. These self-attention based methods have achieved promising performance on various point cloud tasks, but their models are still less efficient with a large amount of parameters and high complexity. Our methods adopt the same self-attention mechanism in the design of a lightweight architecture to achieve the state-of-the-art or comparable performance with orders of magnitude lesser model parameter and computational complexity.

C. Point Cloud Auto-Encoding

Point cloud generation are generally divided into 2 categories: Generative Adversarial Network (GAN) based and non-GAN based methods. GAN-based generative model [2], [26], [12], [27] sample point cloud based on the data distribution learned during training. These methods often output realistic point cloud with great details, but generally require more complex training procedures.

Non-GAN based methods are mostly in encoder-decoder, unsupervised architecture [28]. To improve reconstruction performance, [29] requires the point cloud to be a 1D ordered list structure. [11], [10] parameterize the decoder with 2D grid patches as part of the latent representation, while [27] parameterizes point cloud with spectral frequencies. Our autoencoder is a non-GAN, self-supervised based architecture that reconstructs input point cloud based only on a latent vector and without any other parameterization.

III. PRELIMINARY

A. Self-Attention Layer

The self-attention layer $\mathcal{A}(\cdot)$ we adopt is a scaled dot-product attention mechanism [13] $\mathcal{A} : Q, K, V \rightarrow \mathbb{R}^{N \times d}$ which takes as input Query $Q \in \mathbb{R}^{N \times d}$, Key $K \in \mathbb{R}^{N \times d}$ and Value $V \in \mathbb{R}^{N' \times d}$, and aggregates information from $V$ for each token in $Q$ based on the alignment with $K$:

$$\mathcal{A}(Q, K, V) = \sigma \left( \frac{Q K^T}{\sqrt{d}} \right) V$$

where $\sigma$ is a softmax operator. Multi-head attention simply concatenates the output of multiple self-attention layer. The output is of $\mathbb{R}^{N \times h d}$ where $h$ is the number of head.

Point cloud data is here defined as an unordered set of $N$ points $P = \{p_i\}_{i=1}^N \in \mathbb{R}^3$ in 3D Euclidean space without additional features. When applied on point cloud processing, self-attention layer acts analogously to a convolution kernel. Query points are analogous to the kernel’s center. Whereas the neighbouring points, in which the Key and Value are derived from, are analogous to the rest of the kernel inputs. Multihead attention is then analogous to a convolution filter. Self-attention model is well suited in handling point cloud [14], [15], [30], [31] because:

- Self-attention is invariant to input permutation and hence able to handle input point cloud data in the form of unordered set.
- Instead of one-to-many relationship rendered by convolution kernel, self-attention improves points interactions by having many-to-many relationship in the input set in an efficient manner.
- With adequate normalization in the input set, self-attention is robust under rigid transformation such as translation and rotation.

B. Processing Pipeline

Unlike CNN’s grid structure, point cloud requires several pre-processing layers in a hierarchical pipeline [9]:

- **Sampling.** Given a set of points, a subset is selected and formed into an unstructured probing grid.
- **Grouping.** Given a set of points and its probing grid, the nearest neighbours are gathered into local neighbourhoods for each probing points.
- **Normalizing.** Given a set of local neighbourhoods, points are normalized by subtracting its probing point and scaling by a constant.
- **Interpolating.** Given neighbourhoods from 2 adjacent levels, features are propagated from the smaller neighbourhood with distance based interpolation.
IV. MODEL ARCHITECTURE

The proposed SA-CNN consists of 2 basic operators for processing data in the form of unordered set. We will first discuss these operators, then we describe how to build with these operators for different applications.

A. SA-CNN Modules

SAConv is a set convolution stack that aggregates features
\[ \text{SAConv} : \mathbb{R}^{n \times d_l} \rightarrow \mathbb{R}^{n \times d_{l+1}} \]
from an unsorted set of size \( n \) and outputs a single feature vector. For a self-attention layer to work analogously to a convolution kernel on point cloud, Sampling-Grouping-Normalizing (SGN) pre-processing layer is deployed to form the input \( \mathcal{X} \in \mathbb{R}^{(k+1) \times d} \) where \( d \) is the feature dimension and \( k \) is the number of nearest neighbour of a probing point. Query, Key and Value in Eq. 1 are from

\[
\begin{align*}
Q &= \mathcal{X}W_q \\
K &= \mathcal{X}W_k \\
V &= \mathcal{X}W_v
\end{align*}
\]

the linear layers where \( W_{q,k} \in \mathbb{R}^{d \times d_q} \) and \( W_v \in \mathbb{R}^{d \times d_v} \) are the weights. The output of the multi-head attention then goes through a sequence of common layers as depicted in Fig. 2 (left). A pooling layer is placed at the end as the symmetric function that aggregates the final features in the set.

SAConvT is a transposed convolution stack proposed to generate an unordered set \( \text{SAConv}^T : \mathbb{R}^{d_l} \rightarrow \mathbb{R}^{n \times d_{l+1}} \) based on a single input feature vector \( \mathcal{X} \). It consists of a random sampling layer that draw \( N' \in \mathbb{R}^{n \times d} \) samples from unit Gaussian distribution and

\[
\mathcal{X}' = \text{MLP} \left( \text{concat}_d(\mathcal{X}, \mathcal{X}W_r + N') \right)
\]

where \( W_r \in \mathbb{R}^{d \times d'} \) is the weight of the linear layer before the broadcasted element-wise multiplication \( * \) with \( N' \), and \( \text{concat}_d \) is concatenation along feature dimension. Multi-head attention takes it as input \( \mathcal{A}(\mathcal{X}') \) and then followed by a sequence of common layers as depicted in Fig. 2 (right).

Positional Encoding were proposed [13] to provide sequential information for each tokens. Although unsorted, points in point cloud are well represented in euclidean space. Therefore, we use the 3D \( xyz \) coordinate of point as its positional encoding. Combining Normalizing pre-processing layer, the positional encoding is its normalized coordinate with respect to the probing point of the neighbourhood. In the hierarchical pipeline, \( d_l = 3 + d_{l-1} \) given \( d_0 = 0 \) indicates the input dimension of level \( l \). Concretely, in each hierarchical levels the input to SAConv is the concatenation of the normalized 3D coordinate and the features computed from the previous level given the input dimension is 3.

B. Applications of SA-CNN

1) Classification and Part Segmentation: We apply SA-CNN modules as a drop-in replacement to solve the point cloud classification and part segmentation tasks using the hierarchical feature extraction pipeline [9] as depicted in Fig. 3. Classification network is composed of several levels, each consists of a SGN-SAConv pair. Each level abstracts a subset of probing points, while the last level groups (GN, without sampling) all the remaining for the final classification output. The part segmentation network adopts U-Net [32] structure with skip connections. ISGN combines interpolating layer with SGN and is used to propagate features to the next level.

2) Auto-encoding: We apply SA-CNN modules to solve the point cloud reconstruction task with a hierarchical auto-encoder architecture depicted in Fig. 4. Similar to the pipeline of the classification task, the encoder \( f : \mathcal{P} \rightarrow z \) takes in as input a set of points \( \mathcal{P} \in \mathbb{R}^{N \times 3} \) and processes it with several SGN-SAConv pairs. A MLP then turns the features into a latent representation vector \( z \in \mathbb{R}^L \) where \( L \ll N \).

The hierarchical decoder \( g : z \rightarrow \{ \hat{\mathcal{P}}_l \}_{l=1}^H \) takes as input the latent vector, and expands it through \( H \) levels of SGN-SAConvT pairs. Each hierarchical level expands every points in the level and produces a set of output points \( \mathcal{P} \) given

\[
|\mathcal{P}_a| < |\mathcal{P}_b| \quad \text{if} \quad a < b \quad \text{and} \quad |\mathcal{P}_H| = N.
\]
The multi-head attention in the SAConvT stack of the decoder takes as input a point feature \( X_i \) as well as the feature of its nearest neighbours in the previous level \( \bar{P}_{i-1} \). After Eq (3) the Query, Key and Value:

\[
Q = X_i^T W_q \quad (4a)
\]

\[
K = \text{concat}_n (X_i^T W_k, X_{i-1}^T W_{k'}) \quad (4b)
\]

\[
V = \text{concat}_n (X_i^T W_v, X_{i-1}^T W_{v'}) \quad (4c)
\]

where \( W_q, k, v, k', v' \) are the weights of the linear layers, and \( \text{concat}_n \) is concatenation along token dimension. A MLP then turns the features of the level into points in euclidean space by scaling and translating it to the input point’s coordinate frame. Fig [5] depicts the decoding process.

The objective of the auto-encoder is to reconstruct the input point cloud by minimizing chamfer distance \([2]\):

\[
\text{chamfer}(\mathcal{P}_1, \mathcal{P}_2) = \frac{1}{|\mathcal{P}_1|} \sum_{p \in \mathcal{P}_1} \min_{q \in \mathcal{P}_2} ||p - q||_2 + \frac{1}{|\mathcal{P}_2|} \sum_{q \in \mathcal{P}_2} \min_{p \in \mathcal{P}_1} ||p - q||_2 \quad (5)
\]

between the input points \( \mathcal{P} \) and the output point \( \bar{\mathcal{P}}_l \)

\[
L_{rCs} = \sum_{l=1}^{H} \alpha_l \text{chamfer}(\mathcal{P}_l, \bar{\mathcal{P}}_l) \quad (6)
\]

in each levels where \( \alpha_l \propto |\bar{\mathcal{P}}_l| \) and \( \alpha_H = 1 \). Hence, the reconstruction loss \( L_{rCs} \) is the weighted summation of the chamfer distances from all levels. This improves gradient propagation during the training of the auto-encoder.

3) Data Retrieval: For the data retrieval task, [33] proposed to use the outputs of the penultimate fully-connected layer in the classification network as low dimensional representation of the input point cloud. In contrast, latent representation of point cloud trained with an auto-encoder can be used for the data retrieval task. Given an object point cloud, its latent representation should be closer to those from the same class in term of cosine distance. We proposed to use this latent representation for the point cloud shape retrieval task. To utilize labeled data, we include a triplet loss \( L_{tri} \) to better model the latent space during training:

\[
L_{tri} = \max (\cos(z, z_{n}) - \cos(z, z_p) + \alpha, 0) \quad (7)
\]

where \( \cos(\cdot) \) is the cosine similarity score ranging from 0 to 1, \( z_p \) and \( z_n \) are positive and negative inputs from the same and different class respectively.

C. Implementation Details

There are 4 hierarchical levels in the classification network. The pooling function used in SAConv stacks are [Max, Max, Avg]. For segmentation network, there are \( 4 \times 2 \) levels in the U-Net structure. Max Pooling is only used for the first 3 levels and Average Pooling for the rest. Category one-hot vector is concatenated to the feature before the last level. For auto-encoding network, there are \( 3 \times 2 \) levels in the U-Net structure. For simplicity, \( d_q = d_k = d_v \) sizes for Query, Key and Value are the same in a self-attention layer.

We use Farthest Point Sampling (FPS) and K-Nearest Neighbour (KNN) in Sampling and Grouping layers respectively. In Normalizing layer we subtract probing point’s coordinates from the neighbourhood and multiply different constants in each levels. In Interpolating layer we use \( \exp(-3d) \) where \( \beta = 16 \) for weighted average of features based on point’s relative distance \( d \). MLP consists of multiple set of [Dropout, Linear, BatchNorm, GELU] sequence.

V. EXPERIMENT

In this section, we evaluate the proposed SA-CNN models in a series of comprehensive experiments. Concretely, we first evaluate our model in classification and part segmentation benchmarks. Then, demonstrate the ability in shape retrieval and performance in unsupervised classification of our auto-encoder. Lastly, we visualize the results of point cloud reconstruction and illustrate the continuity of the latent space through latent walk in multiple object domains.

A. Shape Classification

We evaluate SA-CNN on 3D point cloud classification task using ModelNet40 [34] which consists of 9843 training models and 2468 test models in 40 classes. We uniformly sample 1024 points, zero the bounding box’s center and normalize to unit sphere. During training we augment data with per-point Gaussian noise with std 0.001, random rotation between \([-25, 25]\) degree in the vertical axis and \([-5, 5]\) for the other 2
TABLE I
MODELNET40 CLASSIFICATION BENCHMARK. INPUT CONSISTS OF NUMBER OF POINTS, 'P' (POINT) AND/OR 'N' (NORMAL).

| Methods            | Input | #param(M) | Accuracy(%) |
|--------------------|-------|-----------|-------------|
| PointNet [8]       | 1k p  | 3.5       | 89.2        |
| PointNet++ [9]     | 1k p  | 1.5       | 90.7        |
| PointNet++ [9]     | 5k p,n | 1.5     | 91.9        |
| Ki-Net [35]        | 32k p | 2.0       | 91.8        |
| PointCNN [16]      | 1k p  | 0.3       | 92.5        |
| DGCNN [18]         | 1k p  | 1.8       | 92.9        |
| PCNN [36]          | 1k p  | 1.4       | 92.3        |
| DensePoint [33]    | 1k p  | 0.7       | 93.2        |
| RS-CNN [37]        | 1k p  | 1.4       | 93.6        |
| KPConv [17]        | 7k p  | 14.3      | 92.9        |
| Pnt Timer [24]     | 1k p,n | 13.50    | 92.8        |
| Pnt Timer [20]     | 1k p  | 9.6       | 93.7        |
| PCT [21]           | 1k p  | 2.9       | 93.2        |
| PointASNL [25]     | 1k p  | -         | 92.9        |
| AttPNet [22]       | 1k p  | -         | 93.6        |
| PAT [23]           | 1k p  | -         | 91.7        |
| SA-CNN (Ours)      | 1k p  | 0.04      | 92.3        |

TABLE II
COMPUTATIONAL COMPLEXITY FOR MODELNET40 CLASSIFICATION. ALL METHODS TAKE AS INPUT 1024 POINTS ON A GEFORCE RTX 3090.

| Methods            | FLOPs (samples) | Params size(Mb) | Training size(Mb) |
|--------------------|-----------------|-----------------|-------------------|
| PointNet [8]       | 878M            | 14              | 60                |
| PointNet++ [9]     | 1.69G           | 6               | 138               |
| DGCNN [18]         | 4.78G           | 7               | 185               |
| KPConv [17]        | 200M            | -               | -                 |
| PointCNN [16]      | 210M            | 1               | 22                |
| Pnt Timer [24]     | 92G             | 88              | 5736              |
| Pnt Timer [20]     | 3.6G            | 38              | 6334              |
| PCT [21]           | 70G             | 12              | 2400              |
| DensePoint [33]    | 651M            | -               | -                 |
| RS-CNN [37]        | 295M            | -               | -                 |
| SA-CNN (Ours)      | 8M              | 0.2             | 2                 |

axes, scaling between [0.6667, 1.5] and translation between [-0.2, 0.2] in all 3 axes. During testing, we adopt voting strategy where 10 tests are performed with the same data augmentations except rotations and noise.

Table I shows the comparison of classification results in terms of overall accuracy and the number of parameters on ModelNet40. Statistic in terms of floating point operations (FLOPs) per sample and memory for parameter and training required on GPU are depicted in Table II. SA-CNN classifier has 4/5/6/7 as number of head and attention size, 8/7/6/1 as the neighborhood size, with sampling ratio 0.3N/0.075N/0.015N, feature size 16/32/64/96 in the 4 levels, and a p = 0.6 dropout before the output layers.

Due to the simple and efficient hierarchical architecture SA-CNN achieves the lowest number of parameter, FLOPs and memory by several order of magnitude, while having comparable accuracy to the other self-attention and non-attention methods. This advantage makes SA-CNN suitable for low computation/memory point cloud applications. Although generic, the accuracy gap (1.4%) to the state-of-the-art suggests that SA-CNN could work with additional mechanism [25], [24], [23] to boost classification performance.

B. Part Segmentation

We evaluate SA-CNN on fine-grained object recognition using ShapeNet part [39] benchmark in point cloud parts segmentation task which consists of 12137 models for training and 2874 for testing. Models are classified into 16 categories, and each category contains between 2 to 6 parts. For each model we zero the bounding box’s center and normalize to a unit sphere. During training we augment data with random rotation between [-10, 10] degree, scaling between [0.8, 1.25] and translating between [-0.1, 0.1] in all 3 axes. The performance metric used for this task is intersection-over-union (IoU). There are 3 IoUs reported: IoU for each category, mean IoU (mIoU) over all categories, and mIoU over all instances.
TABLE III
PART SEGMENTATION RESULTS (IOU %) ON SHAPENET PART BENCHMARK. INPUTS ARE EITHER 3D POINTS, OR POINTS WITH NORMAL (N).

| method          | input | cat. mIoU | int. mIoU | air | bag | cap | car | chair | car | phone | guitar | knife | lamp | laptop | motor | bike | mug | pistol | rocket | skate | board | table |
|-----------------|-------|-----------|-----------|-----|-----|-----|-----|-------|-----|-------|--------|-------|------|--------|-------|------|-----|--------|--------|-------|-------|-------|
| SCN [38]        | 1k    | 81.8      | 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 [13]     | 4k    | 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 [8]    | 2k    | 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   |
| PointNet++ [9]  | 2k,n  | 81.9      | 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   |
| PCNN [56]       | 2k    | 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   |
| DGCNN [18]      | 2k    | 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-CNN [37]     | 2k    | 84.0      | 86.2      | 83.5| 84.8| 88.8| 79.6| 91.2  | 81.1| 91.6  | 88.4   | 86.0  | 96.0 | 73.7   | 94.1  | 83.4 | 60.5 | 77.7   | 83.6   |
| DensePnt [33]   | 2k    | 84.2      | 86.4      | 84.0| 85.4| 90.0| 79.2| 91.1  | 81.6| 91.5  | 87.5   | 84.7  | 95.9 | 74.3   | 94.6  | 82.9 | 64.6 | 76.8   | 83.7   |
| Point Tfmr [20] | -     | 83.7      | 86.6      | -   | -   | -   | -   | -     | -   | -     | -      | -     | -    | -      | -     | -    | -    | -      | -      |
| AttPNet [22]    | 2k    | 82.8      | 85.2      | -   | -   | -   | -   | -     | -   | -     | -      | -     | -    | -      | -     | -    | -    | -      | -      |
| GAPNet [14]     | 2k    | 82.0      | 84.7      | 84.2| 84.1| 88.8| 78.1| 90.7  | 70.1| 91.0  | 87.3   | 83.1  | 96.2 | 65.9   | 95.0  | 81.7 | 60.7 | 74.9   | 80.8   |
| LAE-Conv [15]   | 2k    | 84.1      | 85.9      | 83.3| 86.1| 85.7| 80.3| 90.5  | 82.7| 91.5  | 88.1   | 85.5  | 95.9 | 77.9   | 95.1  | 84.0 | 64.3 | 77.6   | 82.8   |
| SA-CNN (Ours)   | 2k    | 84.0      | 86.7      | 85.1| 85.3| 85.4| 78.6| 91.7  | 79.4| 91.4  | 87.6   | 86.8  | 95.5 | 72.2   | 94.6  | 81.3 | 62.3 | 81.5   | 84.8   |

Fig. 8. Latent walk in robots and human models.

Quantitative comparison among the other methods are summarised in Table III. SA-CNN achieves state-of-the-art on instance mIoU and several category IoUs. These results demonstrate SA-CNN’s capability and robustness in recognising a diverse range of fine-grained shapes. Fig 10 depicts some examples of the ShapeNet segmentation results.

C. Reconstruction with Auto-Encoder

In this section, we demonstrate the performance of SA-CNN in auto-encoding. We train the auto-encoder using models from two different domains. For rigid object reconstruction, we use models from ModelNet40 [34] and ShapeNetCore [40] with 40 and 55 classes respectively. For non-rigid object reconstruction, we use SMPL [41] human, allegro hand and aliengo quadruped robot model. All non-rigid dataset consists of 4096 synthetically generated training models of different poses. We uniformly sample 2048 points as input as well as output for the training of auto-encoder, and visualize the hierarchical decoding and reconstructions with latent size of 128 in Fig 6 and Fig 7.

Next, we evaluate SA-CNN qualitatively on auto-encoding by visualizing the latent walks of both rigid and non-rigid objects. Linear interpolations of latent code between 2 models are fed to the decoder to get the output reconstructions. Fig 8 and Fig 9 depict some examples of latent walks.

TABLE IV
THE SHAPE RETRIEVAL RESULTS ON THE MODELNET40.

| Methods       | Latent Size | mAP (%) |
|---------------|-------------|---------|
| PointNet [8]  | -           | 70.5    |
| PointCNN [16] | -           | 83.8    |
| DGCNN [18]    | -           | 85.3    |
| DensePnt [33] | 256         | 88.5    |
| AE            | 32          | 84.0    |
| AE            | 64          | 85.4    |
| AE            | 128         | 86.7    |
| AE            | 256         | 87.1    |
| AE + Triplet  | 32          | 88.9    |
| AE + Triplet  | 64          | 89.2    |
| AE + Triplet  | 128         | 90.1    |
| AE + Triplet  | 256         | 89.7    |

Fig. 9. Latent walk among ShapeNetCore objects.
D. Shape Retrieval

We evaluate the performance of SA-CNN auto-encoding with shape retrieval task. We train the auto-encoder using ModelNet40. For this task, given a query shape in test set, we first extract its latent vector using the encoder. Then we search for the top 10 nearest latent vectors seen during training by cosine distance. We report the mean average precision (mAP) based on the retrieved nearest neighbours.

Table V summarizes the shape retrieval performance trained with various latent sizes. With pure unsupervised training, our auto-encoder is able to match with methods that train with classification model. With just 32 latent size our method that train with reconstruction and triplet loss surpasses all other methods. Fig 11 depicts some examples of queries and the corresponding nearest neighbours. It is observed that some instances from classes such as cup and pot, and chair and stool, are highly similar, which leads to lower performance in classification by retrieval.

E. Unsupervised Classification

In this section, we evaluate the quality of SA-CNN representation learning by comparing classification accuracy using only the latent code as the input feature of a linear Support Vector Machine (SVM). We pre-train the auto-encoder using ShapeNetCore, and then fit and test the SVM using the latent code of ModelNet40 train/test split. Table V illustrates the latent size and results of the task. Our auto-encoder achieves the state-of-the-art with only 256 latent size show that SA-CNN is capable of compact and efficient unsupervised representation learning for point cloud data.

F. Model Complexity

Model complexity is an important aspect while using self-attention mechanism. In addition to Table II, we further summaries model complexity in term of number of parameters and FLOPs per sample of SA-CNN’s part segmentation and auto-encoding models in Table VI.

SA-CNN part segmentation network has 4/5/6/7/8/8/8/7/6 number of head and attention size, 0.4N, 0.12N, 0.024N sampling ratio, 14/12/10/8 neighbourhood size, 16/32/64/96/128/128/128/64 feature dimension in the levels, and uses Max pooling for the first 3 and Avg pooling for the rest of the SAConv. Auto-encoding networks have similar settings but increase proportionally to the latent size.

VI. Conclusion

In this work, we present SA-CNN, a lightweight self-attention based encoding and decoding architecture for 3D point cloud representation learning. The proposed SA-CNN is a drop-in replacement for multiple hierarchical point cloud analysis tasks. We demonstrate that SA-CNN achieves the state-of-the-art or comparable performance to other methods on multiple benchmarks with significantly lower model complexity. In addition, we visualize the multi-stages point cloud reconstruction and their latent walks on rigid objects as well as non-rigid human and robot models.
