Multi-View 3D Shape Recognition via Correspondence-Aware Deep Learning

Yong Xu, Chaoda Zheng, Ruotao Xu, Yuhui Quan* and Haibin Ling

***Abstract*—In recent years, multi-view learning has emerged as a promising approach for 3D shape recognition, which identifies a 3D shape based on its 2D views taken from different viewpoints. Usually, the correspondences inside a view or across different views encode the spatial arrangement of object parts and the symmetry of the object, which provide useful geometric cues for recognition. However, such view correspondences have not been explicitly and fully exploited in existing work. In this paper, we propose a correspondence-aware representation (CAR) module, which explicitly finds potential intra-view correspondences and cross-view correspondences via kNN search in semantic space and then aggregates the shape features from the correspondences via learned transforms. Particularly, the spatial relations of correspondences in terms of their viewpoint positions and intra-view locations are taken into account for learning correspondence-aware features. Incorporating the CAR module into a ResNet-18 backbone, we propose an effective deep model called CAR-Net for 3D shape classification and retrieval. Extensive experiments have demonstrated the effectiveness of the CAR module as well as the excellent performance of the CAR-Net.**

***Index Terms*—3D Shape Analysis, Multi-View Learning, Correspondence Learning, Object Recognition**

I. Introduction

Understanding 3D geometric data (*e.g.* 3D shape/object models) has been a fundamental problem since the born of computer vision. It allows including features derived from 3D shapes into recognition pipelines, with broad application prospects such as autonomous driving, robotics, augmented reality, digital entertainment, civil infrastructure monitoring, medical imaging, among others; see *e.g.* [1], [2], [3], [4]. In the past, due to the limited availability of 3D geometric data, early work mainly focuses on the theoretical representation of 3D shapes (*e.g.* [5]). With recent development of 3D sensing technologies as well as the prevalence of computer graphics data, there is a growing interest in developing practical techniques for 3D shape recognition.

Classification and retrieval are two popular tasks in 3D shape recognition. Traditional approaches focus on the design of handcrafted shape features; see *e.g.* [5], [6]. Embracing the advances of deep learning in image recognition [7], many recent studies (*e.g.* [8], [9], [10], [11], [12]) address the problem via training a deep neural network (DNN) from a labeled dataset. Unlike images which are generally represented as matrices or tensors, 3D geometric data has various forms owing to the diversity of 3D sensors and 3D modeling software, such as 3D point cloud, volumetric grid, and multi-view images. These forms have their own properties and advantages. According to the form of DNN's input, existing deep learning approaches for 3D shape classification and retrieval can be divided into three kinds: point-based approaches, volumetric approaches, and multi-view approaches.

Point-based approaches (*e.g.* [13], [10], [14], [15], [16], [17]) consume point clouds, a popular format of 3D data, using a DNN. A point cloud is generally a set of point coordinates with irregular organization and orderless structure. Such differences from image data (indexed by row/column) hinder the direct use of traditional convolutional neural networks (CNNs) for processing point clouds. Thus, specifically-designed DNN architectures are needed in point-cloud-based approaches; see *e.g.* [13], [18]. Volumetric approaches (*e.g.* [9], [19], [20], [8], [21]) accept voxelized 3D data as input, where a 3D shape is represented as a volumetric binary occupancy grid. Unlike point clouds, a volumetric grid has a regular ordered structure in analogy to image's grid. Therefore, the CNNs that process images can be adopted to handle voxelized 3D data by simply extending 2D convolutional layers to the 3D ones. Since the computational efficiency of 3D convolutional layers does not scale well in accordance with the resolution of input voxelized data, one key in volumetric approaches is the acceleration of related processing in the DNN; see *e.g.* [22], [23].

Multi-view approaches (*e.g.* [11], [24], [25], [26], [27], [28], [29], [30], [31], [32]) capture multiple 2D views from a 3D object and then use the views to train and test a DNN. Most existing multi-view approaches adopt the following pipeline: extracting view-wise features and then aggregating them for classification. The advantages of multi-view approaches are plenty. Firstly, the DNNs pre-trained on images can be directly leveraged for processing 2D views. Secondly, multi-view approaches have a mechanism similar to human perception, *i.e.*, people look around an object to understand it by combining surfaces' information from multiple views. In fact, cognitive

Yong Xu is with School of Computer Science and Engineering at South China University of Technology, China, as well as with Peng Cheng Laboratory, Shenzhen, China. (email: yxu@scut.edu.cn)

Chaoda Zheng is with the School of Science and Engineering, Chinese University of Hong Kong, Shenzhen 518172, China, as well as with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China. (email: chaodazheng@link.cuhk.edu.cn)

Ruotao Xu is with School of Computer Science and Engineering at South China University of Technology, China. (email: xu.ruotao@mail.scut.edu.cn)

Yuhui Quan is with School of Computer Science and Engineering at South China University of Technology, China, as well as with Guangdong Provincial Key Laboratory of Computational Intelligence and Cyberspace Information, China. (email: csyhquan@scut.edu.cn)

Haibin Ling is with the Department of Computer Science, Stony Brook University, Strony Brook, USA. (email: hling@cs.stonybrook.edu)

*Corresponding author: Yuhui Quan.

This work was supported in part by National Nature Science Foundation of China under Grants 62072188 and 61872151, in part by Science and Technology Program of Guangdong Province under Grant 2019A050510010, and in part by CCF-Tencent Open Fund 2020.

supports have been given by many studies that human visual perception on 3D shapes relies on various 2D observations from different view points [33]. Thirdly, geometric representation artifacts (*e.g.* no interior, polygon soups, non-manifold geometry) may be avoided in multi-view approaches [34]. Fourthly, multi-view approaches are directly applicable to the 3D objects captured with backgrounds. Last but not least, view images can be generated from any other format of 3D objects.

With all aforementioned advantages, multi-view approaches have achieved very encouraging results; see *e.g.* [12], [24], [25], [34], [29]. This motivated us to develop a multi-view deep learning approach for 3D object classification and retrieval, with excellent performance provided.

A. Main Idea

A good model for learning 3D shape representations from multiple views should be capable of exploiting both intra-view patterns and cross-view relations to improve the effectiveness of learned features. Intra-view patterns can be characterized by the view's local semantic features and the correspondences of these semantic features. Such feature correspondences inside a view are referred to as *intra-view correspondences*. In comparison, cross-view relations are described by views' consistency, which says object parts in one view have their correspondences in other views and their semantic features are carried along the way. Such correspondences across different views are referred to as *cross-view correspondences*.

A 3D shape often has many similar parts due to symmetry of the object and repetition of object parts. How these similar parts distribute provides useful clues for identifying the object. For instance, an object with four identical wheels distributed on two rows is likely a car, while the one with only two wheels along a line is likely a bike. The similar parts can be presented in different regions of the same view, which are captured by the intra-view correspondences, and can also be presented in different views, which are captured the cross-view correspondences. Therefore, discovering intra/cross-view correspondences and analyzing their spatial relation can reveal useful knowledge, *e.g.*, how object parts relate to each other and how they vary along with viewpoints changes, and improve the discrimination and robustness of learned features.

The spatial relation of a pair of correspondences can be described by the correspondences' intra-view positions, as well as by the viewpoints from which the views are taken. While intra-view positions are undoubtedly useful, viewpoints are also important for recognition tasks as they encode the location information from the 3D camera space. Also, the viewpoints' relation is correlated to the semantics of object parts. For instance, similar views from adjacent viewpoints are likely to contain the correspondences associated with the same or close similar object parts, while non-overlapping views from two distant viewpoints are likely to contain the correspondences associated to similar parts of distinct spatial locations. Furthermore, different viewpoints may lead to different manners that intra-view correspondences are related by.

Inspired by the importance and benefits of intra/cross-view correspondences, we develop a Correspondence-Aware Representation (CAR) module for DNN-based 3D shape recognition, whose main idea is illustrated in Fig. 1. Considering every feature inside a view as a point in some semantic space, the CAR module finds its K nearest neighbors from the views as its potential correspondences. Each of the K correspondence pairs, represented by semantic features as well as 3D locations regarding viewpoints and intra-view positions, is encoded identically and independently into a description. Then the results are aggregated into a correspondence-aware representation as the module's output, which can be used in subsequent parts of an end-to-end DNN. In practice, the CAR module can be applied to different network layers, in order to exploit view correspondences at different scales using different levels of features provided by the DNN.

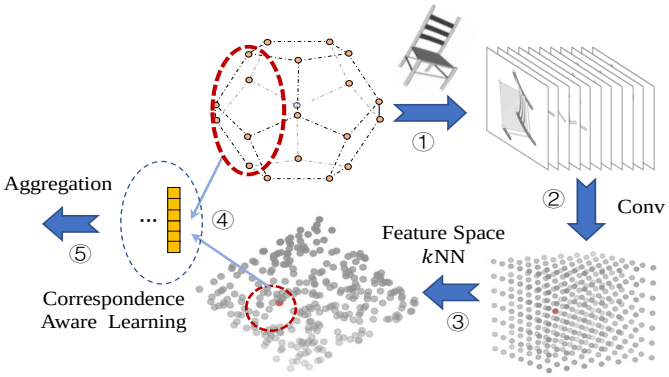


Fig. 1: Illustration of main idea of CAR module. The views generated from a 3D object with some viewpoints are represented in a semantic feature space. Viewing the local feature at each position of each view as a point in the semantic space, all the features are represented as a point set embedded in the semantic space. The kNN search is used to find the K potential correspondences for each point in the space. Our CAR module learns to output a correspondence-aware representation by exploiting the spatial relation of these potential correspondences in terms of viewpoints and intra-view positions.

B. Contributions

By combining the proposed CAR module with a ResNet-18 backbone, we develop an effective model called CAR-Net for 3D shape classification and retrieval. Benefiting from the explicit correspondence-aware learning, our CAR-Net can generate improved shape descriptors from both intra-view and cross-view correspondences. Evaluated on the ModelNet40, ModelNet10 and ShapeNetCore55 datasets, the CAR-Net shows excellent performance, with noticeable performance gain from correspondence-aware learning. In particular, even using fewer views, it still outperforms many existing ones.

Learning correspondence-aware representations is a challenging task, as the intra/cross-view correspondences of 3D objects have various patterns. For instance, the corresponding object parts may be far away from or close to each other, may vary in terms of poses, may deform due to projections, or even may not exist in other views. Most existing approaches only take cross-view similarities into account during feature

aggregation (*e.g.* [35], [24], [25], [36]), or exploit cross-view dependencies by using recurrent neural networks (RNNs) on sequentialized views (*e.g.* [37], [38], [28], [34], [29]). These approaches only exploit cross-view correspondences implicitly, and ignore intra-view correspondences. Few approaches exploit both intra-view and cross-view correspondences simultaneously. One available work [32] uses global features reweighted with a nonlocal fashion to encode the similarities between all intra-view/cross-view feature pairs. However, this method does not include 3D positional information of view features (*e.g.* viewpoints) which is crucial for shape description. The viewpoint relation is exploited in some existing work by treating viewpoints as learned latent variables [12] or as the vertexes of a graph DNN [39]; however, these approaches do not focus on learning correspondence-aware representations.

In comparison to the above approaches, the proposed one in this paper has the following advantages:

- capability of explicitly learning and inferring from both intra-view and cross-view correspondences, and
- awareness of the spatial distribution of viewpoints.

Such advantages bring performance improvement of the proposed approach over existing ones.

C. Organization

The rest this paper is structured as follows. Section II is devoted to literature review. Section III gives the detailed description of the proposed CAR-Net. Section IV presents the experimental evaluation. Section V concludes the paper.

II. Related Work

In this section, a literature review is given on the deep learning approaches for 3D shape classification and retrieval, including point-based approaches, volumetric approaches, multi-view approaches and other approaches. We will focus more on the multi-view approaches, as our approach falls into this type.

A. Point-Based Approaches

One seminal work of point-based approaches is PointNet [13], which learns individual features via a per-point multi-layer perceptron (MLP) and combines them into a global permutation-invariant descriptor via element-wise max pooling. PointNet cannot make full use of the local structure of a point for capturing fine-grained shape patterns, as the features are computed in a per-point manner. Some later studies introduced hierarchical feature extraction mechanisms for better characterizing local shapes. PointNet++ [40] applies PointNet recursively on the nested groups of the input point cloud. Klokov *et al.* [14] proposed to build a kd-tree on the input point set and run hierarchical feature extraction from the leaves to the root. The kd-tree conducts a non-overlap partition on the space of input data, and thus the receptive fields of the resulting DNN are non-overlapping. Li *et al.* [15] addressed this issue by replacing the kd-tree with a self-organizing map and performing kNN search from points to the map's nodes.

There are also studies on designing local point operations for encoding complex local shapes. Shen *et al.* [41] integrated the point-set kernel learning into the PointNet, which learns a set of 3D points that jointly respond to a set of neighboring data points according to their geometric affinity measured by correlation. Liu *et al.* [18] proposed a specific convolutional layer which can learn low-level geometric relations among points. Liu *et al.* [17] proposed an RNN-based sequence model to capture the contextual information inside the local regions of a point set, with an attention-based aggregation on multi-scale areas. In [10], [16], [42], the cloud point is modeled as a graph, on which graph convolutions/CNNs are applied.

B. Volumetric Approaches

The early studies of volumetric approaches, such as [9], [19], [20], [8], [21], employ CNNs equipped with 3D convolution layers to process voxelized data. Owing to the complexity of 3D convolutions, the computational time, memory cost and model size of these approaches increase very fast as the resolution of the voxel grid or the depth of the DNN model increases, resulting in both low efficiency and limited effectiveness. Recently, some voxel-based approaches have been proposed with reduced computational burden and improved compactness. Riegler *et al.* [22] proposed to exploit the sparsity of the voxel grid for acceleration in processing high-resolution voxel grids. Kumawat *et al.* [23] proposed an efficient alternative to the standard 3D convolutional layer.

C. Multi-View Approaches

One of the very first studies in multi-view approaches is the MVCNN proposed by Su *et al.* [11]. It extracts view-wise features with a shared CNN, which are then aggregated for classification via a view-level max pooling layer. Though simple, MVCNN provides a general pipeline adopted by later multi-view approaches. One key in such a pipeline is the view feature aggregation module, and some studies have been focused on improving the feature aggregation by exploiting similarities among view content. Wang *et al.* [35] proposed to run recurrent clustering on view features and aggregate within-cluster view features by weighted average pooling. Feng *et al.* [24] proposed a hierarchical grouping scheme for view-level pooling, where the pooling is done in each group and each level so that the intrinsic hierarchical correlations among views are encoded. Yu *et al.* [25] proposed the harmonized bilinear pooling which aggregates view features based on local patch similarity across views. Zheng *et al.* [36] proposed to learn weighted average for view-level pooling based on the affinity ranking of view features.

There are some approaches exploiting dependencies among views for learning better view features. Johns *et al.* [26] proposed to model and learn the relation of view features from paired views. Xu *et al.* [27] proposed to enhance the representation of each view via its adjacent views. A few studies [37], [38], [28], [34], [29] treat all the views as a view sequence ordered by some rule and learn the view dependencies from those sequences using RNNs. The main difference among these approaches lies in the RNN's architecture. Ma *et al.* [37] used the long short-term memory units (LSTM) with highway connections. Dai *et al.* [38] used the LSTM with

Siamese structure. Han *et al.* [28] used a two-way LSTM. Xu *et al.* [34] used a bi-directional LSTM. Jiang *et al.* [29] applied multiple LSTMs on multiple looping view sequences. Instead of using RNNs, Han *et al.* [30] proposed to process the view sequence with hierarchical attention. He *et al.* [31] proposed to process view sequences using the n-gram model in natural language processing, which is more efficient than RNNs. Yang *et al.* [32] proposed to learn region-to-region relationships among different views in a nonlocal fashion.

Compared with aforementioned approaches, the proposed one in this paper exploits the dependencies among views by explicit correspondence-aware learning. In addition, it takes the relation between viewpoints into account. The viewpoint relation is also considered in Kanezak *et al.* [12], which treats viewpoints as latent variables learned for pose alignment. In comparison, the proposed approach utilizes viewpoint relations by an MLP. A parallel work to ours is conducted by Wei *et al.* [39], which models the spatial relations among views with a graph and exploits such relations by a graph DNN.

It is worth mentioning that multi-view 3D shape recognition systems often suffer from high computational cost of projection rendering and feature matching. See [43] for the related acceleration techniques for multi-view approaches. It is also worth mentioning that, instead of using views generated by perspective projection under virtual pin-hole cameras, Sfikas *et al.* [44], [45] used panoramic view for the 2D representation of 3D objects.

D. Other Approaches

There is a group of approaches that take mesh surfaces as input by generalizing CNNs to non-Euclidean geometries (*e.g.* spectral CNNs [46] and anisotropic CNNs [47]) or by using handcrafted features of 3D objects as input (*e.g.* [48]). These approaches are devoted to shape matching, but not to classification and retrieval tasks. Also note that a few multi-model approaches use two or more formats of 3D data for further improvement, *e.g.* both voxels and views are used in [49]. Muralikrishnan *et al.* [50] proposed to learn a unified representation of 3D shapes across different formats of 3D shape data including point cloud, voxels and views. In addition, there are also some approaches built upon other machine learning techniques, *e.g.* multi-hyper-graph learning [51].

III. Main Body

A. Framework

The architecture of the proposed CAR-Net for 3D shape recognition is illustrated in Fig. 2. Given a 3D object, its multiple views in the form of 2D images $\boldsymbol{V}_1, \cdots, \boldsymbol{V}_M \in \mathbb{R}^{H_0 \times W_0 \times 3}$ are generated as input, and the CAR-Net, denoted by $\mathcal{F}$, outputs a label prediction vector $\boldsymbol{\ell} \in \mathbb{R}^N$ for N classes:

$$\mathcal{F} : \{\boldsymbol{V}_1, \cdots, \boldsymbol{V}_M\} \subset \mathbb{R}^{H_0 \times W_0 \times 3} \rightarrow \boldsymbol{\ell} \in \mathbb{R}^N. \quad (1)$$

The backbone of CAR-Net is a ResNet-18 [7]. Recall that the ResNet-18 contains five stages; see Fig. 2. The first stage, denoted by $CONV_1$, sequentially connects two convolutional layers and a pooling layer. The rest four stages are denoted by $RES_2, \cdots, RES_5$ respectively, each of which is composed of two residual blocks. We adopt the following modifications on ResNet-18: (a) The backbone is divided into three parts, and two proposed CAR modules are inserted into the middle part; (b) A module for view feature aggregation is added before the classifier. As a result, the CAR-Net is sequentially composed of four parts:

$$\mathcal{F} = \mathcal{F}_1 \circ \mathcal{F}_2 \circ \mathcal{F}_3 \circ \mathcal{F}_4, \quad (2)$$

which are summarized as follows:

- ($\mathcal{F}_1$) **View-wise feature representation.** The first part $\mathcal{F}_1$ maps each view to some feature space for semantic representation, which is done by separately passing each view to the first part of the backbone CNN.
- ($\mathcal{F}_2$) **Correspondence-aware representation learning.** Given the features of all views from $\mathcal{F}_1$, the second part $\mathcal{F}_2$ captures and exploits their view correspondences based on two sequential CAR modules. This step results in a correspondence-aware feature tensor for all views.
- ($\mathcal{F}_3$) **View-wise feature refinement.** The enhanced features from the CAR modules are further refined by applying the second part of the backbone view-wisely.
- ($\mathcal{F}_4$) **Feature aggregation.** The refined features of all views from $\mathcal{F}_3$ are aggregated into a single descriptor and then passed to a classifier.

The details of each part are given in the following subsections. In particular, the CAR module is our main focus.

For the convenience of presentation, we define the notations used in the remainder sections as follows. Light letters are used for scalars (*e.g.* m, M), bold lower letters for column vectors (*e.g.* $\boldsymbol{x} = [x_1, \cdots, x_M]$), bold upper letters for matrices or tensors (*e.g.* $\boldsymbol{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_M]$), hollow upper letters for sets (*e.g.* $\mathbb{R}, \mathbb{X}$), and calligraphic upper letters for operators (*e.g.* $\mathcal{F}$).

B. Multi-View Representation of 3D Object

Multi-view learning approaches for 3D shape recognition take multiple view images from a 3D object as input. Typically, 3D objects are stored as polygon meshes or point clouds. Following existing works [11], [12], [35], the multi-view images of a 3D object are generated by rendering the mesh or point cloud under the perspective projection of a virtual camera with varied viewpoints. Before rendering, the 3D object is uniformly scaled into a unit sphere.

The setup of camera viewpoints can be critical to the performance of a multi-view approach. A viewpoint (*i.e.* spatial position of virtual camera) can be represented as a unit vector in $\mathbb{R}^3$. There are two widely-adopted viewpoint setups in existing literature, as shown in Fig 3, which are

- **Circle-12.** There are 12 virtual cameras placed evenly around the object with elevation angle $\theta = 30^\circ$ from the ground plane, pointing towards the object's centroid.
- **Dodecahedron-20.** There are 20 virtual cameras placed at the vertices of a dodecahedron enclosing the object.

Both the setups will be used in the experiments. Following [12], all the views are of the size 224×224.

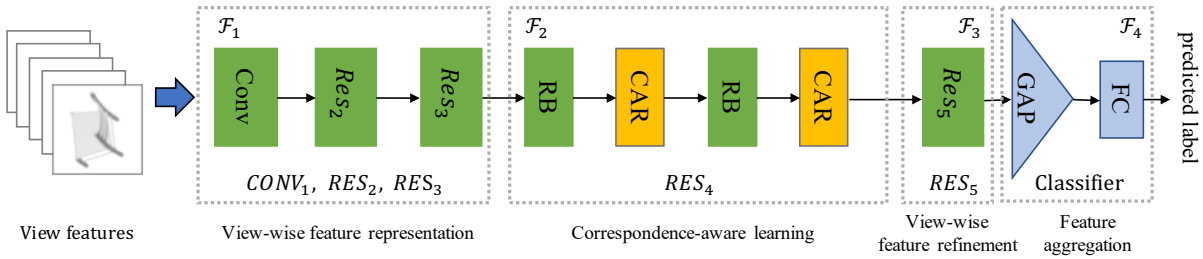
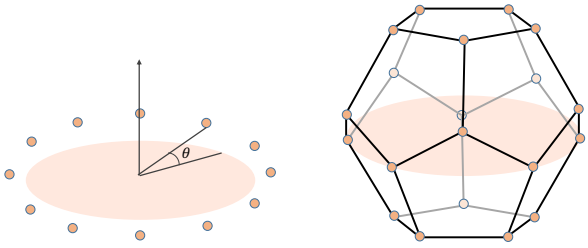


Fig. 2: Architecture of CAR-Net for 3D shape recognition. The GAP stands for Global Average Pooling.



(a) Circle-12 (b) Dodecahedron-20

Fig. 3: Illustration of two camera viewpoint setups.

C. View Feature Representation and Refinement

Recall that our CAR-Net contains two backbone parts used in $\mathcal{F}_1$ and $\mathcal{F}_3$ respectively. Given M views $\boldsymbol{V}_1, \cdots, \boldsymbol{V}_M \in \mathbb{R}^{H_0 \times W_0 \times 3}$ as input, the backbone part in $\mathcal{F}_1$ is applied to each view individually and identically:

$$\mathcal{F}_1 : \boldsymbol{V}_m \in \mathbb{R}^{H_0 \times W_0 \times 3} \rightarrow \boldsymbol{Z}_m^1 \in \mathbb{R}^{H_1 \times W_1 \times C_1}, \quad (3)$$

for $m = 1, \cdots, M$, where $\boldsymbol{Z}_m^1$ denotes the output C_1-channel feature tensor on $\boldsymbol{V}_m$. All $\boldsymbol{Z}_m^1$s are input to $\mathcal{F}_2$ and transformed to a set of correspondence-aware feature tensors:

$$\mathcal{F}_2 : \{\boldsymbol{Z}_m^1\}_{m=1}^M \subset \mathbb{R}^{H_1 \times W_1 \times C_1} \rightarrow \{\boldsymbol{Z}_m^2\}_{m=1}^M \subset \mathbb{R}^{H_2 \times W_2 \times C_2}. \quad (4)$$

Similar to $\mathcal{F}_1$, the backbone part in $\mathcal{F}_3$ operates on its input feature tensors view-wisely:

$$\mathcal{F}_3 : \boldsymbol{Z}_m^2 \in \mathbb{R}^{H_2 \times W_2 \times C_2} \rightarrow \boldsymbol{Z}_m^3 \in \mathbb{R}^{H_3 \times W_3 \times C_3}, \quad (5)$$

for all m, where $\boldsymbol{Z}_m^3$ corresponds to the feature tensor of the m^{th} view.

For simplicity, instead of using two ResNets individually, we use the single model of ResNet-18 [7] for the two backbone parts. The first three stages including $CONV_1, RES_2, RES_3$ are used for $\mathcal{F}_1$, and the last stage RES_5 is used for $\mathcal{F}_3$; see Fig. 2. The RES_4 is used for $\mathcal{F}_2$ in which a CAR module is inserted after each of the two residual blocks. The reason we select RES_4 for attaching the CAR modules is, the feature tensors in RES_4 are of moderate size for acceptable efficiency during correspondence-aware learning, and meanwhile they provide sufficient semantic/original information for finding high-quality potential correspondences.

D. Correspondence-Aware Representation Module

The key component of CAR-Net is the CAR module, which aims at enhancing the convolutional view-wise features from $\mathcal{F}_1$, via simultaneously exploiting intra-view and cross-view correspondences. The CAR module accepts the feature tensors $\boldsymbol{X}_1, \cdots, \boldsymbol{X}_M$ from the residual block as input, and it outputs the correspondence-aware features which are of the same size as the input:

$$\begin{aligned} \text{CAR Module: } \{\boldsymbol{X}_1, \cdots, \boldsymbol{X}_M\} &\subset \mathbb{R}^{H \times W \times C} \\ &\rightarrow \{\boldsymbol{Y}_1, \cdots, \boldsymbol{Y}_M\} \subset \mathbb{R}^{H \times W \times C}. \end{aligned} \quad (6)$$

The pipeline of the CAR module is illustrated in Fig 4. The module treats the feature tensors from all views as a set of MHW features and it contains two steps: (a) *correspondence searching* which finds top K potential correspondences for every feature, and (b) *correspondence encoding* which utilizes potential correspondences to generate correspondence-aware feature representations.

Correspondence searching. For the simplicity of presentation, we rearrange the input feature tensors $\boldsymbol{X}_1, \cdots, \boldsymbol{X}_M$ into a matrix $\boldsymbol{X}$ as follows:

$$\{\boldsymbol{X}_1, \cdots, \boldsymbol{X}_M\} \subset \mathbb{R}^{H \times W \times C} \rightarrow \boldsymbol{X} \in \mathbb{R}^{MHW \times C}. \quad (7)$$

Each row of $\boldsymbol{X}$ corresponds to a C-dimensional local feature of some view at a certain spatial location, or say, it is a feature of a local patch in some view image, which can be viewed as a point in a C-dimensional feature space. In the first step, the kNN grouping is called to find the K nearest neighbors for each local feature (*i.e.* each row of $\boldsymbol{X}$) in the C-dimensional feature space:

$$k\text{NN} : \boldsymbol{X} \in \mathbb{R}^{MHW \times C} \rightarrow \boldsymbol{I} \in \{1, 2, \cdots, MHW\}^{MHW \times K}, \quad (8)$$

where $\boldsymbol{I}$ is the matrix of KNN's indices such that $\boldsymbol{I}(j,:)$ stores the indices of the K nearest neighbors of $\boldsymbol{X}(j,:)$ in a decreasing order in terms of similarity. Concretely, the search is done by comparing $\boldsymbol{X}(j,:)$ with other rows in terms of ℓ_2 distance, for all j. We totally find $K = K_1 + K_2$ nearest neighbors including K_1 ($K_1 = 2$ by default) intra-view nearest neighbors and K_2 cross-view nearest neighbors. Then, the collected nearest neighbors of $\boldsymbol{X}(j,:)$ are selected as its potential inter/intra-view correspondences. Recall that the C-dimensional feature space is generated from the backbone, which encodes certain degree of visual semantics. As a result, the discovered potential correspondences are likely to correspond to the same or similar parts of an object; see Fig. 5 for an illustration of collected potential cross-view correspondences.

Correspondence encoding. The second step of our CAR module is to encode the key information of collected potential

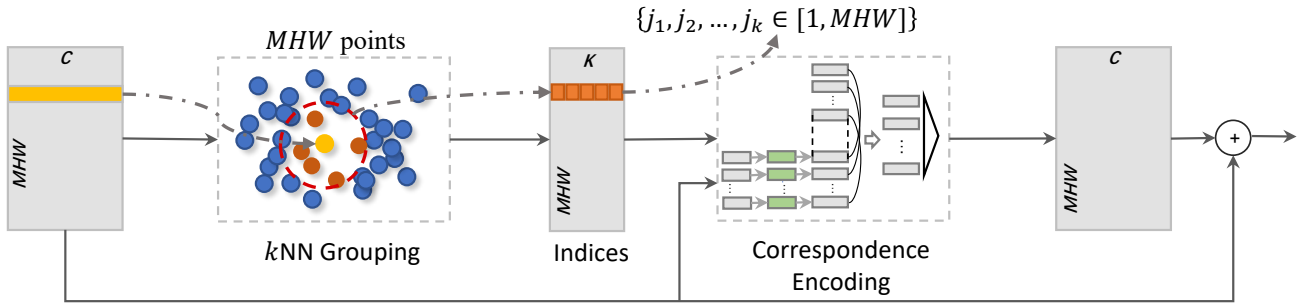


Fig. 4: Pipeline of CAR module. The structure of the Correspondence Encoding (CE) layer is illustrated in Fig 6.

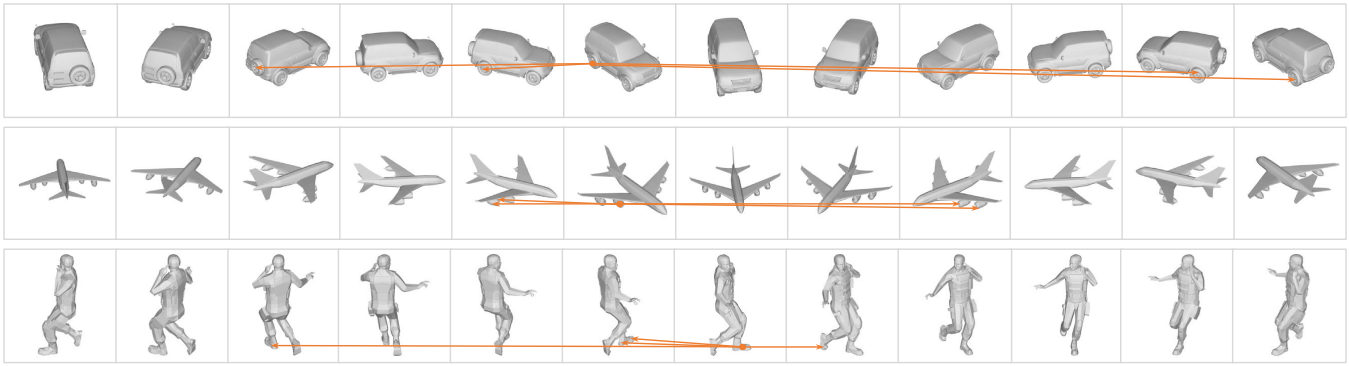


Fig. 5: Examples of potential correspondences found by our CAR module. Each row gives the 12 views of a 3D object. The starting point of arrows represents the source, and the ending points represent its four potential cross-view correspondences. The visualization results show that our model can find reasonable intra/cross-view correspondences to support its prediction.

correspondences into the feature representation. We first generate a correspondence-aware feature by applying a so-called correspondence encoding (CE) layer. The CE layer accepts the feature matrix $\boldsymbol{X}$ and the correspondence indices matrix $\boldsymbol{I}$ as input, and outputs a feature matrix $\boldsymbol{Y}$:

$$\text{CE} : (\boldsymbol{X}, \boldsymbol{I}) \rightarrow \boldsymbol{Y} \in \mathbb{R}^{MHW \times C}, \tag{9}$$

See Fig. 6 for the the workflow of the CE layer. Then $\boldsymbol{X}$ is combined with $\boldsymbol{Y}$ as the module's output:

$$\boldsymbol{X} \rightarrow \boldsymbol{X} + \boldsymbol{Y}. \tag{10}$$

Recall that $\boldsymbol{X}$ and $\boldsymbol{I}$ have the same number of rows, and their rows store the views' features and the associated feature correspondence indices respectively. The CE layer processes each view feature in $\boldsymbol{X}$ as follows. For convenience, we define

$$\boldsymbol{f}^j := (\boldsymbol{X}(j,:))^\top \in \mathbb{R}^C. \tag{11}$$

In other words, $\boldsymbol{f}^j$ is the j^{th} feature in $\boldsymbol{X}$, which is a local feature in some view. Let $\boldsymbol{v}^j \in \mathbb{R}^3$ denote the unit viewpoint vector of virtual camera associated with the view of $\boldsymbol{f}^j$, *i.e.*, it indicates the camera position in the 3D space. Further, let h^j, w^j denote the normalized spatial indices (*i.e.* normalized to $[0,1]$ from $[1,H]$ and $[1,W]$) of $\boldsymbol{f}^j$ in the view. Then, the entire information associated with $\boldsymbol{f}^j$ is described as a tuple:

$$\text{local feature:} \quad (\boldsymbol{f}^j, \boldsymbol{v}^j, h^j, w^j), \ \forall j. \tag{12}$$

By looking up the KNN's indices stored in $\boldsymbol{I}(j,:)$, we have the top K (potential) correspondences of $\boldsymbol{f}^j$, whose indices are denoted by $j_1, \cdots, j_K$. Each correspondence is also represented as a quadruplet

$$\text{correspondence:} \quad (\boldsymbol{f}^{j_k}, \boldsymbol{v}^{j_k}, h^{j_k}, w^{j_k}), \ k = 1, \cdots, K. \tag{13}$$

Given $\boldsymbol{f}^j$ as the source, a correspondence-aware feature is generated from each of its potential correspondences as follows. For each of its K correspondences, the following descriptor $\boldsymbol{d}_k^j$ is defined:

$$\boldsymbol{d}_k^j := (\boldsymbol{f}^j, \boldsymbol{v}^{j_k} - \boldsymbol{v}^j, \|\boldsymbol{v}^{j_k} - \boldsymbol{v}^j\|_2^2, h^{j_k} - h^j, w^{j_k} - w^j) \in \mathbb{R}^{C+6}. \tag{14}$$

It can be seen that such a descriptor contains the source semantic feature $\boldsymbol{f}^j$, as well as the difference between $\boldsymbol{f}^j$ and its k^{th} correspondence in terms of viewpoints and intra-view positions. Each description vector is passed to a shared MLP:

$$\text{MLP}: \ \boldsymbol{d}_k^j \in \mathbb{R}^{C+6} \rightarrow \boldsymbol{q}_k^j \in [0,1]^C. \tag{15}$$

The underlying idea behind is that the spatial configurations of intra/cross-view correspondences provide very useful clues for identifying 3D shapes. The spatial configurations can be characterized by the displacement of viewpoints and the difference of intra-view spatial indices between the feature and its correspondence. Thus, inputting $\boldsymbol{d}_k^j$ to the MLP can encode the spatial configurations of correspondences for feature enhancement. In our MLP, the output $\boldsymbol{q}_k^j$ has the same size as $\boldsymbol{f}^{j_k}$, and it is combined with $\boldsymbol{f}^{j_k}$ to generate an enhanced feature $\boldsymbol{y}^{j_k}$ with correspondence awareness for the k^{th} correspondence as follows:

$$\boldsymbol{y}^{j_k} = \boldsymbol{f}^{j_k} \odot \boldsymbol{q}_k^j, \tag{16}$$

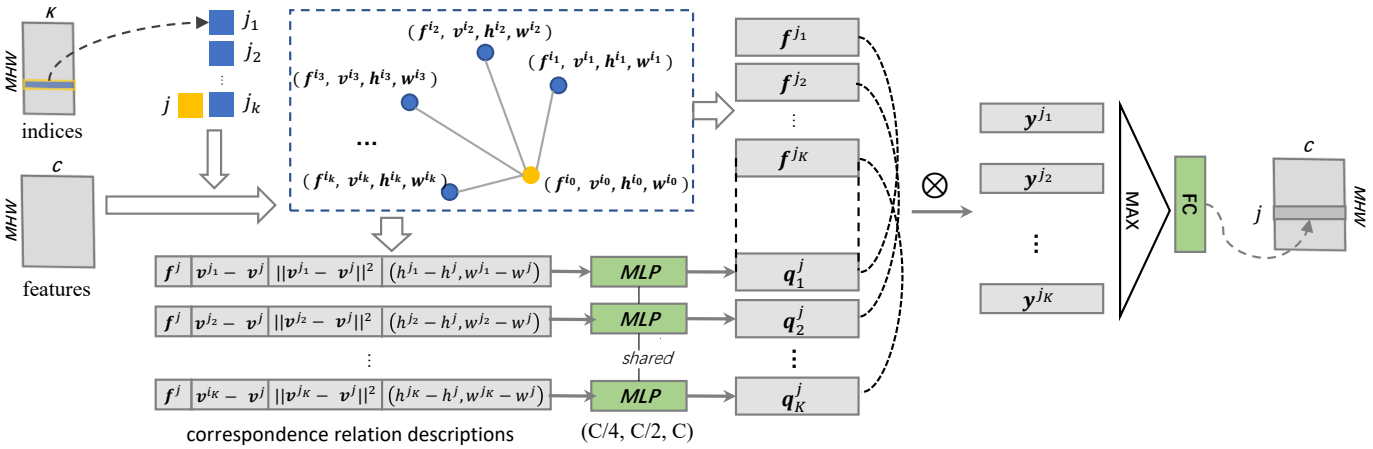


Fig. 6: Workflow of CE layer. Each MLP consists of three FC layers with RELU activation and BN at its end. The FC layer is equipped with ReLU activation and BN. The numbers under the MLPs denote the output size of each layer of the MLP.

for all k, where $\odot$ denotes element-wise product. It can be seen that $\boldsymbol{y}^{j_k}$ encodes both the local semantic features $\boldsymbol{f}^j, \boldsymbol{f}^{j_k}$ and the spatial distribution relation of this correspondence pair.

Afterwards, the enhanced features $\{\boldsymbol{y}^{j_k}\}_{k=1}^K$ are aggregated into a single correspondence-aware feature $\boldsymbol{y}^j$ as follows:

$$\text{aggregation}: \ \{\boldsymbol{y}^{j_k}\}_{k=1}^K \subset \mathbb{R}^C \rightarrow \boldsymbol{y}^j = \text{FC}(\max_k(\boldsymbol{y}_k^j)) \in \mathbb{R}^C, \tag{17}$$

for any j, where $\max(\cdot)$ denotes element-wise max pooling, and $\text{FC}(\cdot)$ denotes a fully-connected (FC) layer with ReLU activation followed by BN. The max pooling takes the most interesting information (strongest responses) over all correspondences for maximal utilization of the enhanced representations, and it also brings the invariance to correspondences' order. The FC layer acts as a transform aligning $\max_k(\boldsymbol{y}_k^j)$ to the semantic space of $\boldsymbol{f}^j$ (*i.e.* bridging the semantic gap between $\max_k(\boldsymbol{y}_k^j)$ and $\boldsymbol{f}^j$), as its result will be added to $\boldsymbol{f}^j$ in the last step of the CAR module defined in (10). All aggregated features $\boldsymbol{y}^j$, $j = 1, \cdots, MHW$, are stacked into a matrix $\boldsymbol{Y}$ where $\boldsymbol{Y}(j,:) = (\boldsymbol{y}^j)^\top$, which is transformed back to a feature tensor and then used as the output of the CE layer.

It is worth mentioning that, for the correspondence descriptor $\boldsymbol{d}_k^j$, we do not directly use $(\boldsymbol{v}^{j_k}, \boldsymbol{v}^j, h^{j_k}, h^j, w^{j_k}, w^j)$ but their paired difference instead. This enforces the MLP to explicitly explore the difference between a pair of correspondences for prediction. In addition, it helps to reduce the MLP's size. Moreover, we do not include the semantic feature $\boldsymbol{f}^{j_k}$ into $\boldsymbol{d}_k^j$ for the MLP's input, as $\boldsymbol{f}^{j_k}$ provides little additional information over $\boldsymbol{f}^j$ owing to the similarity between $\boldsymbol{f}^{j_k}$ and $\boldsymbol{f}^j$. This also helps to reduce the MLP's size. Empirically, we find that introducing $\boldsymbol{f}^{j_k}$ brings very minor improvement but an increase of model size. See Section IV-E4 for more analysis on the correspondence descriptor.

Remark 1. *(CAR module vs. Nonlocal module) The mechanism behind our CAR module has certain similarity to that of nonlocal modules (e.g. [52], [32]), and we compare them as follows. A nonlocal module generally takes the similarity between all feature pairs into account, while our CAR module explicitly selects top-K correspondences. On the one hand, nonlocal modules may exploit additional information ignored by our module. One the other hand, the explicit selection of top-K correspondences in our module avoids introducing/overfitting useless features from unrelated pairs that might be harmful for recognition/learning. In addition, our CAR module encodes the viewpoint relation in the representation, which is not utilized in existing nonlocal modules.*

E. View Feature Aggregation and Loss Function

In the final aggregation stage $\mathcal{F}_4$, the refined features of all views from $\mathcal{F}_3$ are aggregated by a global average pooling layer which averages the entries within each view along the height and the width dimensions. Afterwards, an FC layer with softmax activation is applied for predicting the class label.

Given a set of training data $\{(\mathbb{X}^q, \boldsymbol{l}^q)\}_{q=1}^Q$, where $\mathbb{X}^q$ is a 3D object associated with the one-hot class label $\boldsymbol{l}^q \in \{0,1\}^N$ for N classes. We first generate the view sequence $\boldsymbol{V}_1^q, \cdots, \boldsymbol{V}_M^q$ on each 3D object $\mathbb{X}^q$. Let $\boldsymbol{\ell}^q = \mathcal{F}(\{\boldsymbol{V}_1^q, \cdots, \boldsymbol{V}_M^q\})$ denote the soft label predicted by CAR-Net. Then the loss function for training CAR-Net is defined by the cross-entropy:

$$\mathcal{L} := -\sum_{q=1}^{Q}\sum_{n=1}^{N} \boldsymbol{l}^q(n)\log(\boldsymbol{\ell}^q(n)). \tag{18}$$

IV. Experiments

A. Implementation Details

Our CAR-Net is implemented with MXNET. In training, the backbone (*i.e.* ResNet-18) of CAR-Net is initialized using the model pre-trained on ImageNet. A dropout layer with dropping ratio 0.5 is inserted before the final FC layer to reduce overfitting. The model weights in the CAR modules are randomly initialized with the MSRA approach [55]. The scale parameter of the FC's BN in the CE layer is set to zero to ensure each CAR module is an identity mapping before training, so as to minimize the influence to the pre-trained backbone model at the beginning of training. The Momentum optimizer with a momentum 0.9 and a weight decay of 5e-4 is used for minimizing the training loss. The initial learning rate

TABLE I: Performance comparison on ModelNet datasets. The $n\times$ behind each multi-view method indicates the number of views. The methods are grouped according to the type of their input. Best results are boldfaced.

Input	Method	ModelNet40		ModelNet10	
		Classification Accuracy (%)	Retrieval mAP (%)	Classification Accuracy (%)	Retrieval mAP (%)
Voxels	VRN-Ensemble [20]	95.54	-	97.14	-
	VRN (w/o Ensemble) [20]	91.33	-	93.61	-
	LP-3DCNN [23]	92.10	-	94.40	-
Point cloud + Views	PVNet [53]	93.20	89.50	-	-
Point Cloud	RS-CNN [18]	93.60	-	-	-
	SO-Net [15]	93.40	-	95.70	-
	LDGCNN [16]	92.90	-	-	-
	Point2Sequence [17]	92.60	-	95.30	-
	PointNet++ [40]	91.90	-	-	-
	PointNet [13]	89.20	-	-	-
Views	RotationNet [12], 20×	97.37	-	98.46	-
	RotationNet [12], 12×	91.00	-	94.00	-
	MVCNN-New [54], 12×	95.00	-	-	-
	View N-gram [31], 12×	-	89.30	-	92.80
	Adjacent Views [27], 36×	-	87.05	-	-
	Relation Network [32], 12×	94.30	86.70	95.30	-
	MHBN [25], 6×	94.70	-	95.00	-
	MLVCNN [29], 36×	94.16	92.84	-	-
	Wang et al. [35], 12×	93.80	-	-	-
	3D2SeqViews [30], 12×	93.40	90.76	94.71	92.12
	GVCNN [24], 12×	93.10	85.70	-	-
	Ma et al. [37],12×	91.05	84.34	95.29	93.19
	MVCNN [11], 80×	90.10	79.50	-	-
	CAR-Net [Ours], 12×	95.22	91.27	95.82	91.53
	CAR-Net [Ours], 20×	**97.73**	**95.04**	**99.01**	**97.12**

is set to 0.01 for the backbone CNN parts and 0.1 for the CAR modules, which is divided by 10 every 20 epochs. By default, the CAR-Net is equipped with two CAR modules at RES_4 and $K = 8$. For the variants of CAR-Net used in subsequent experiments, if not specified, we train and test them using the same protocol as CAR-Net.

B. Evaluation on ModelNet Datasets

The ModelNet40 and ModelNet10 benchmark datasets are adopted for our experiments. These two datasets are widely used for the performance evaluation of 3D shape recognition, which are two subsets of the ModelNet [8], a large repository of clean 3D CAD models. The ModelNet40 dataset contains 12311 objects from 40 categories, where the training/test split is $9843/2468$. The ModelNet10 dataset contains 4899 objects from 10 categories, where the training/test split is $3991/908$.

To demonstrate the effectiveness of the proposed CAR-Net, we compare it with 18 representative 3D shape recognition methods on the classification and retrieval tasks. The methods for comparison consist of nine multi-view methods [12], [54], [25], [29], [35], [30], [24], [37], [11], two voxel-based methods [20], [23], six point-based methods [18], [15], [16], [17], [13], [40] and one multi-modal method [53]. For classification, the performance is measured in terms of classification accuracy measured by instance accuracy (*i.e.* number of correctly-classified samples over total number of samples). For retrieval, following existing work, we extract shape descriptors from the input of the last layer of the model. Then taking each test sample as a query, we compute a ranking list on the remainder test samples, with the descend order of similarity measured by the ℓ_2 distance between two shape descriptors. The ranking lists of all the test samples are then used to calculate the mean average precision (mAP).

Both the Circle-12 and Dodecahedron-20 camera setups mentioned in Section III-B are used for the test. See Table I for the results. The results of the compared methods are all quoted from their original papers, if possible. We use '-' to denote the unavailability of results.

Classification results. As shown in Table I, with 20 views under the Dodecahedron-20 setup, our model outperforms all the other compared methods in the classification task, mostly by a large margin. Even when fed with only 12 views under the Circle-12 setup, our model achieves competitive results on both datasets, surpassing all the competitors except VRN-Ensemble and RotationNet with 20 views. As discussed in its original work [20], VRN-Ensemble mainly benefits from the ensemble of predictions from different models. The ensemble scheme is also applicable to other methods including ours. When VRN is not combined with the ensemble scheme, its performance is not as good as ours with 12 views. As for RotationNet, we can see that its performance is more sensitive than ours to the camera setup. When the number of views is reduced from 20 to 12, the accuracy of RotationNet has a decrease of $6.37\%/4.46\%$ on ModelNet40/10. In comparison, the performance decrease of our CAR-Net is much less. Also note that, the Relation Network [32] is a nonlocal DNN, and its performance is inferior to our CAR-Net.

It is worth mentioning that MHBN achieves comparable results to ours with only 6 views. However, using more views (*e.g.* 12 views) in MHBN yields worse performance [25]. This

TABLE II: Performance comparison on ShapeNetCore55 dataset in terms of four metrics: precision, recall, F-Score, mAP (mean average prediction), and NDCG (normalized discounted cumulative gain). Two schemes are used respectively to obtain average results: Micro (weighted average regarding category size) and Macro (unweighted average). Best results are boldfaced.

Method	Micro (Weighted by category sizes)					Macro (unweighted)				
	Precision	Recall	F1-Score	mAP	NDCG	Precision	Recall	F1-Score	mAP	NDCG
Kanezaki_RotationNet [56]	0.810	0.801	0.798	0.772	0.865	0.602	0.639	0.590	0.583	0.656
Zhou_Improved_GIFT [56]	0.786	0.773	0.767	0.722	0.827	0.592	0.654	0.581	0.575	0.657
Tatsuma_ReVGG [56]	0.765	0.803	0.772	0.749	0.828	0.518	0.601	0.519	0.496	0.559
Furuya_DLAN [56]	**0.818**	0.689	0.712	0.663	0.762	0.618	0.533	0.505	0.477	0.563
Thermos_MVFusionNet [56]	0.743	0.677	0.692	0.622	0.732	0.523	0.494	0.484	0.418	0.502
Deng_CM-VGG5-6DB [56]	0.418	0.717	0.479	0.540	0.654	0.122	0.667	0.166	0.339	0.404
Li_ZFDR [56]	0.535	0.256	0.282	0.199	0.330	0.219	0.409	0.197	0.255	0.377
Mk_DeepVoxNet [56]	0.793	0.211	0.253	0.192	0.277	0.598	0.283	0.258	0.232	0.337
SHREC16-Su_MVCNN [56]	0.770	0.770	0.764	0.735	0.815	0.571	0.625	0.575	0.566	0.640
SHREC16-Bai_GIFT [56]	0.706	0.695	0.689	0.640	0.765	0.444	0.531	0.454	0.447	0.548
Ma *et al.* [37]	0.526	**0.899**	0.602	0.810	0.868	0.151	0.812	0.206	0.604	0.632
3D2SeqViews [30]	0.613	0.804	0.616	**0.852**	**0.909**	0.199	**0.857**	0.252	**0.725**	**0.862**
CAR-Net [Ours]	0.815	0.805	**0.803**	0.772	0.865	**0.627**	0.691	**0.632**	0.624	0.694

suggests that increasing the number of views does not always bring performance improvement. It is also worth mentioning that, while the multi-view methods are overall superior to the voxel-based and points-based methods in the classification task, different types of methods may provide complementary discrimination. In practice, different types of methods could be fused for further improvement, as PV-Net [53] showed.

Retrieval results. In the retrieval task, our model with 20 views as input achieves exciting results, where it consistently outperforms all other competitors by a large margin. When only comparing the methods consuming 12 views, we can see that CAR-Net surpasses other methods on ModelNet40. On ModelNet10, it does not perform as well as 3D2SeqViews [30] and Ma *et al.* [37]. It can also be seen that, the MLVCNN [29] using 36 input views only performs better than our model with 12 views. However, our model using 20 views performs much better than MLVCNN [29].

C. Evaluation on ShapeNetCore55 Dataset

The ShapeNetCore55 dataset is another popular benchmark dataset for 3D object recognition. It contains 51162 objects classified into 55 categories and 203 subcategories. Compared to ModelNet40/10, the ShapeNetCore55 dataset covers a wider range of objects with more complex shapes, and it is the official dataset of the retrieval competition SHREC2017 [56].

Following the standard protocol, the retrieval task is conducted on ShapeNetCore55. We adopt the same network setup as the previous subsection and use 20 input views under the Dodecahedron-20 setup to generate the results. The model is trained on the 55 categories and 203 subcategories respectively. During retrieval, the model trained on 55 categories is used to find out the samples with the same predicted label as the query object. Then the output feature of the model trained on 203 subcategories is used to re-rank these samples in terms of ℓ_2 distance. Four metrics including precision, recall, F-Score, mAP and NDCG (normalized discounted cumulative gain) are used for measuring the retrieval performance. The first three are computed based on binary in-category relevance versus out-of-category relevance, while the last metric uses a graded relevance that additionally considers the matching of the sub-categories. For each of the four metrics, its values over categories are averaged using two schemes respectively: (a) MICRO scheme that uses weighted average regarding category size; and (b) MACRO scheme that uses unweighted average. See [56] for more details on the metrics.

In Table II, our method is compared with Ma *et al.* [37] and 3D2SeqViews [30] as well as all the participants in [56]. The results of the former two are from their original papers [37], [30]. For the participants in [56], we use their official competition results from the website of [56]. In this challenging dataset, our CAR-Net also outperforms all the competitors, which once again demonstrates the effectiveness of our method. Surprisingly, CAR-Net shows noticeable superiority over RotationNet, the top performer in SHREC2017. When compared to Ma *et al.* [37] and 3D2SeqViews [30], our method also yields better results overall. These two compared methods show unbalanced performances under different metrics (*e.g.* high macro recalls and extremely low macro precision). In contrast, our results are more stable across different metrics, with relatively high values achieved. In Fig. 7 we show the precision-recall (PR) curves of the compared methods with available results. It can be seen that our method achieves the best PR curve in both the micro and macro cases.

We would like to note that, following the SHREC'17 Track [56], a varied number of returned entries (at most 1000) is allowed per ranked list. As a result, the choice of the number of returned entries for calculating the performance metrics is up to the competitor (method). In other words, the results in Table II actually may correspond to different numbers of returned entries. To balance the precision and recall, for each query object, we only construct the ranked list on objects with the same predicted class as the query, which is the same as RotationNet [12] (see the official report in [56]). In the results, Ma *et al.* [37] and 3D2SeqViews [30] achieved quite high recall but very low precision (especially on Macro metrics). This is probably because they choose to return more entries for each query.

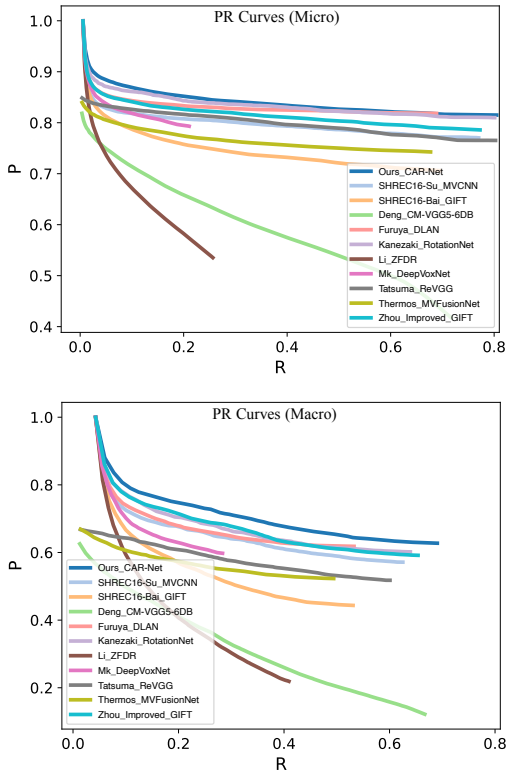


Fig. 7: Precision-recall curves on ShapeNetCore55 dataset.

D. Ablation Studies

The following ablation studies are conducted. (a) To show the benefits of the CAR module, we removed all CAR modules in CAR-Net, and then retrain and test the resulting model on ModelNet40/10 using the same protocol as before. (b) To verify the benefit from using viewpoint relation information in the CAR modules, we keep the CAR modules but remove $\boldsymbol{v}^{j_k} - \boldsymbol{v}^j, \|\boldsymbol{v}^{j_k} - \boldsymbol{v}^j\|_2^2$ from the correspondence descriptor $\boldsymbol{d}_k^j$ defined in (14) for all k, j. The size of each MLP in the CAR module is adapted accordingly. (c) Similarly, we remove $h^{j_k} - j^j, w^{j_k} - h^j$ from $\boldsymbol{d}_k^j$ to evaluate the performance of CAR-Net without using the spatial relative indices for the correspondence descriptor. (d) We replace the max pooling in the CAR module with mean pooling and sum pooling respectively, where the sum pooling acts as a weighted sum using the $\boldsymbol{q}_k^j$ as element-wise weight. (e) To compare the effectiveness of CAR module with nonlocal module in exploiting view correspondence, we replace each CAR module in our CAR-Net by the nonlocal module proposed in [52]. The nonlocal module [52] for video processing is adopted to multi-view processing by considering each view as a video frame. Same as the CAR module, it accepts multi-view features as input and outputs enhanced multi-view features, but without explicitly finding correspondences and utilizing viewpoint relations.

Table III lists the results of the ablation studies on the ModelNet datasets, from which we have the following observations. (a) Without using CAR modules for exploiting view correspondences, noticeable performance decrease is observed. This demonstrates that view correspondences are useful for 3D shape recognition and our CAR modules are capable of utilizing such correspondences. (b) Without exploiting viewpoint relations in the CAR modules, the performance of CAR-Net has certain decrease. This indicates that viewpoint relations in multi-view methods do help improving the recognition on 3D shapes. (c) The spatial relative indices also contribute to the performance improvement. (d) The performance of using sum or mean pooling in the CAR module is inferior to that using max pooling. This indicates that useful clues provided by the CAR module are mainly contributed by the most interesting correspondence (*i.e.* the one with strongest response captured by max pooling). (e) Compared to our CAR module, the nonlocal module brings smaller improvement over the baseline and yields noticeably worse results.

TABLE III: Results of ablation study in terms of classification accuracy (%). Twelve input views are used.

Model	ModelNet40	ModelNet10
CAR-Net	95.22	95.82
CAR-Net (w/o CAR modules)	93.68	94.38
CAR-Net (w/o viewpoint relation)	94.33	95.37
CAR-Net (w/o spatial relative indices)	94.96	95.39
CAR-Net (sum pooling)	94.32	95.48
CAR-Net (mean pooling)	94.49	95.37
Backbone + Nonlocal modules	93.87	95.04

E. More Analysis

Additional experiments are conducted for more analysis.

1) Position of CAR modules: Recall that we attach the CAR module after each residual block in RES_4 of ResNet-18, as the feature tensors at those blocks have moderate sizes for fast correspondence matching and contain sufficient semantic information for accurate matching. It is interesting to see how the performance is influenced when putting the CAR modules to other stages including RES_3 and RES_5 of ResNet-18. For convenience, we use 'Model(RES_p)' to denote the model using the CAR modules at RES_p.

The results on ModelNet40/10 are shown in Table IV, in terms of both classification accuracy and relative training time cost. We can see that Model(RES_5) has slightly lower accuracy than Model(RES_4). This is probably due to that the resolution of feature maps in RES_5 is much smaller than that in RES_4, which contains less information for discrimination. On the other hand, the smaller sizes of feature maps in RES_5 leads to faster speed, as less pairs are compared in the kNN grouping. As for Model(RES_3), its performance is much worse than Model(RES_4) and Model(RES_5). The reason is probably that the RES_3 is too shallow with less feature channels than RES_4 such that the features on it contain insufficient semantics for accurate correspondence matching. What's worse, due to the higher resolutions of feature maps of RES_3, the time cost of Model(RES_3) is much higher than the other two. In Fig. 8, we show the distance maps computed for kNN search on a source region. It can be seen that the distance maps at RES_3 are not discriminative and may be confusing, while those at RES_5 are discriminative but with much lower resolutions. In comparison, RES_4 wins the trade-off.

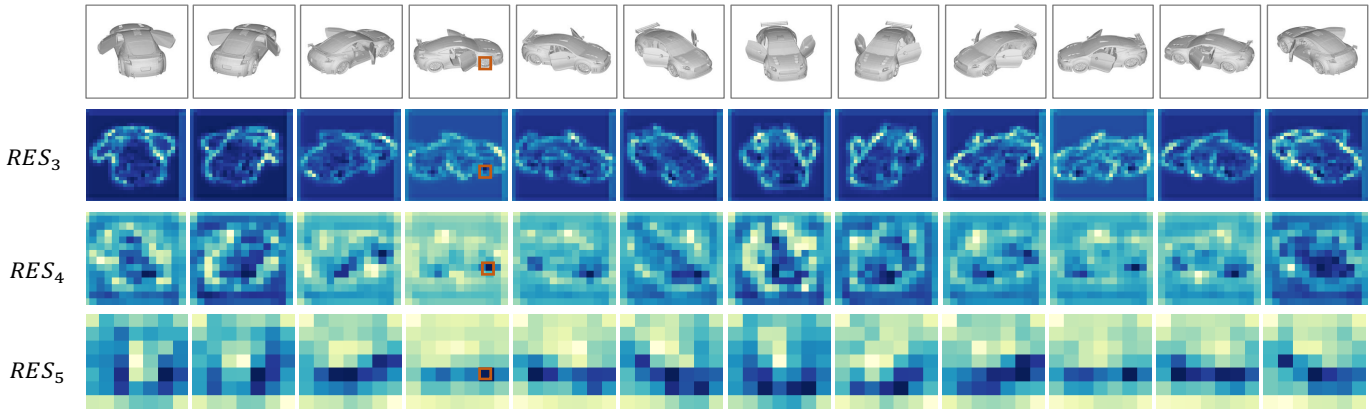


Fig. 8: Distance maps at different stages of the ResNet-18 backbone. The first row provides the views of a 3D car model, where the blue box in the 4th view denotes the source region whose center location is denoted by (r_0, c_0). The last three rows provide the similarity maps at RES_3, RES_4, RES_5 respectively. Each element at (r, c) in the similarity map of the mth view denotes the similarity between the point at (r, c) in the mth view and the point at (r_0, c_0) in the 4th view, measured by the ℓ_2 distance in the feature space defined by $RES_3/RES_4/RES_5$. Darker color denotes higher similarity.

TABLE IV: Performance comparison in terms of classification accuracy (%) on using CAR modules in different stages of ResNet-18. Twelve input views are used.

Model	ModelNet40	ModelNet10	Relative Time Cost
Model(RES_3)	94.61	95.59	4.49×
Model($RES4$)	95.22	95.82	1.00×
Model(RES_5)	95.02	95.70	0.81×

2) Number of CAR modules: We test the performance of CAR-Net using different numbers of CAR modules, including (a) Case-1: using only one CAR module at RES_4; (b) Case-2: original setting where two CAR modules are used at RES_4; (c) Case-4: using two CAR modules at RES_4 and RES_5 respectively; (d) Case-6: using two CAR modules at RES_3, RES_4 and RES_5. The results on ModeNet40/10 are listed in Table V. It can be seen that using CAR modules at multiple stages of ResNet-18 does not bring improvement but leads to some performance degradation. The reason is probably that the feature refinement by the CAR modules is weakened in such cases. In addition, using only one CAR module also leads to slight performance decrease.

TABLE V: Performance comparison in terms of classification accuracy (%) on different configurations of CAR modules.

Choice	Configuration	ModelNet40	ModelNet10
Case-1	RES_4(1 CAR)	95.10	95.26
Case-2	RES_4(2 CAR)	95.22	95.82
Case-4	$RES_{4,5}$(2 CAR)	95.10	95.26
Case-6	$RES_{3,4,5}$(2 CAR)	94.12	94.93

3) Setting of K in correspondence-aware learning: One key hyper-parameter in our CAR module is K, the number of potential correspondences to collect. Recall that $K = K_1 + K_2$ where K_1, K_2 are the number of inter-view/intra-view correspondences respectively. We first test the performance of CAR-Net using varied K_2, with the default value $K_1 = 2$. See Fig. 9 for the results on ModelNet40/10. The highest performance on ModelNet40 and ModelNet10 is achieved with $K_2 = 2$ and $K_2 = 8$ respectively. Overall, there is no significant performance drop/boost as K_2 varies in a moderate range. It can also be seen that the performance first increases and then decreases as K_2 becomes larger. This is because using a too large K_2 may potentially introduce many additional wrong correspondences, while using a too small one may not make full use of cross-view correspondences for improvement.

Next, we study how K_1 influences the performance, which is done by keeping $K_2 = 8$ and varying K_1 from 1 to 6. See Fig. 10 for the results on the ModelNet datasets. Overall, the performance with tends to decrease with K_1 being increased from 2 to 6. This suggests that only a small number of intra-view correspondences are sufficient for correspondence learning, while using too many of them may contaminate useful information. In addition, using $K_1 = 1$ is insufficient to make use of intra-view correspondences and leads to worse result than $K_1 = 2$.

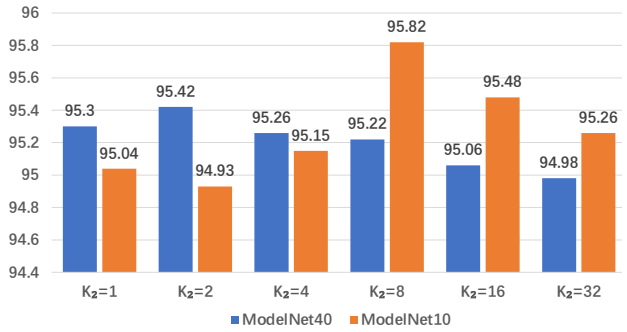


Fig. 9: Classification accuracy (%) of CAR-Net on ModelNet datasets with varied K_2.

4) Viewpoint relation description: Recall that we use $\boldsymbol{d}_k^j = (\boldsymbol{f}^j, \boldsymbol{v}^{j_k} - \boldsymbol{v}^j, \|\boldsymbol{v}^{j_k} - \boldsymbol{v}^j\|_2^2, h^{j_k} - h^j, w^{j_k} - w^j)$ in (14) as the input descriptor of a shared MLP for learning viewpoint/correspondence-aware features. For more analysis on the descriptor, we construct and test the following vari-

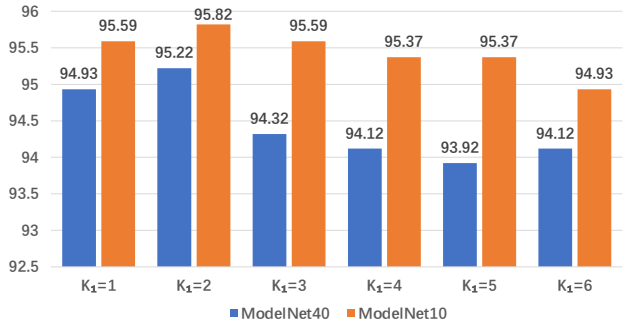


Fig. 10: Classification accuracy (%) of CAR-Net on ModelNet datasets with varied K_1.

ants respectively: $\bar{\boldsymbol{d}}_k^j = (\boldsymbol{f}^j, \boldsymbol{v}^{j_k}, \boldsymbol{v}^j, h^{j_k}, h^j, w^{j_k}, w^j)$, $\hat{\boldsymbol{d}}_k^j = (\boldsymbol{f}^j, \boldsymbol{v}^{j_k} - \boldsymbol{v}^j, h^{j_k} - h^j, w^{j_k} - w^j)$, $\tilde{\boldsymbol{d}}_k^j = (\boldsymbol{f}^{j_k}, \boldsymbol{f}^j, \boldsymbol{v}^{j_k} - \boldsymbol{v}^j, \|\boldsymbol{v}^{j_k} - \boldsymbol{v}^j\|_2^2, h^{j_k} - h^j, w^{j_k} - w^j)$. See Table VI for the results on the ModelNet datasets, from which we can have the following observations. (a) $\bar{\boldsymbol{d}}_k^j$ that uses absolute positions leads to worse results than $\boldsymbol{d}_k^j$ that uses relative positions. This is because explicitly using relative positions emphasizes how viewpoints/correspondences relate instead of where they appear, which makes the model focus on learning relations and prevents it overfitting absolute positions. That is also why we use relative positions in our descriptor. (b) The additional term $\|\boldsymbol{v}^{j_k} - \boldsymbol{v}^j\|_2^2$ in $\hat{\boldsymbol{d}}_k^j$ over $\boldsymbol{d}_k^j$ brings slight improvement on ModelNet40. We use it for further improvement. (c) The additional term $\boldsymbol{f}^{j_k}$ in $\tilde{\boldsymbol{d}}_k^j$ over $\boldsymbol{d}_k^j$ gives very minor improvement, which is mainly due to that $\boldsymbol{f}^{j_k}$ is very similar to $\boldsymbol{f}^j$ such that it brings little useful information for the correspondence learning. Thus, our descriptor does not include $\boldsymbol{f}^{j_k}$.

TABLE VI: Performance comparison in terms of classification accuracy (%) using different correspondence descriptors.

Dataset	$\boldsymbol{d}_k^j$	$\bar{\boldsymbol{d}}_k^j$	$\hat{\boldsymbol{d}}_k^j$	$\tilde{\boldsymbol{d}}_k^j$
ModelNet40	95.02	94.81	94.94	95.03
ModelNet10	95.70	95.37	95.70	95.70

TABLE VII: Results of CAR-Net on ModelNet40 and its subsets. Twelve input views are used.

Subset	# Training samples	# Test samples	Accuracy (%)
Subset 1	1736	600	96.17
Subset 2	1656	440	98.18
Subset 3	2460	520	99.03
Subset 4	3991	908	95.82
Average	2460	617	97.16
ModelNet40	9843	2468	95.22

5) Influence of number of classes / shape types: We split ModelNet40 into four non-overlapping subsets, each of which contains 10 classes. Then we train and evaluate our 10-class CAR-Net on these four subsets separately using the same protocol as that on ModelNet10. The number of training/test samples of the 10-class splits as well as the results are listed in Table VII. The average classification accuracy over the four 10-class subsets is much higher than that on the whole ModelNet40, which is due to that the difficulty becomes higher with the number of classes increased. In addition, noticeable difference is observed among the 10-class subsets, as the challenges from different subsets varies as they contain different types of shapes.

6) Robustness to noise: To see whether the CAR module improves the network's robustness to the noise in test data, we add perturbations sampled from i.i.d. Gaussian distribution $\mathcal{N}(0, 0.2^2)$ to the vertex coordinates of 3D objects in the test split of ModelNet10/40 and then evaluate the performance. For comparison, we re-implement MVCNN [11] by replacing its CNN backbone (VGG-M) with our ResNet backbone, which is denoted by MVCNN-ResNet. We also remove the CAR module from our CAR-Net to form a baseline (denoted by Baseline) for comparison. See Table VIII for the comparison. In terms of of the performance decrease caused by the presence of noise, CAR-Net is better than the other two methods, except that on ModelNet10 it is comparable to MVCNN-ResNet. Such results show that the CAR module brings certain positive contribution to the robustness to noise.

TABLE VIII: Results in noise robustness in terms of classification accuracy (%). Twelve views are used.

ModelNet10	CAR-Net	MVCNN-ResNet	Baseline
Clean	95.82	94.49	94.38
Noisy	94.27 (1.55 ↓)	92.95 (1.54 ↓)	90.30 (4.08 ↓)
ModelNet40	CAR-Net	MVCNN-ResNet	Baseline
Clean	95.22	94.08	93.68
Noisy	87.88 (7.34 ↓)	85.41 (8.67 ↓)	85.57 (8.11 ↓)

7) Computational cost: See Table IX for the results on model size as well as on the average running time in processing one 3D object on ModelNet40. The previous MVCNN-ResNet and Baseline are used for comparison. It can be seen that the additional computational resources caused by our CAR module is not too much and acceptable.

TABLE IX: Comparison on model size and average time for processing a 3D object on ModelNet40. Tested on Tesla V100.

Model	Model Size	Running Time	Relative Time Cost
MVCNN-ResNet	44.8 MB	0.12 seconds	1×
Baseline	44.8 MB	0.12 seconds	1×
CAR-Net	45.7 MB	0.20 seconds	1.65×

V. Conclusion

In this paper, we proposed a multi-view deep learning approach for 3D shape recognition. The key part of our approach is a correspondence-aware learning module with viewpoint awareness. This module reveals the cross-view/intra-view potential correspondences for each feature, and then learns an enhanced feature representation from the correspondence pairs with the corresponding viewpoint relation descriptor. In the experiments, our approach showed excellent performance with impressive results achieved on standard benchmark datasets.

While we used ResNet as the backbone in CAR-Net in this work, our CAR module can be combined with other backbone

CNNs. Also, the CAR module can be directly incorporated into the intermediate convolutional layers of most existing multi-view-based CNNs for 3D shape recognition and for other multi-view tasks. We leave such extensions as our future work. In addition, we will investigate new architectures for better exploiting and integrating view correspondences and viewpoint relations for 3D shape recognition, *e.g.*, using graph CNNs to model/learn viewpoint correspondences with graph structures.

REFERENCES

[1] A. Pfrunder, P. V. Borges, A. R. Romero, G. Catt, and A. Elfes, "Real-time autonomous ground vehicle navigation in heterogeneous environments using a 3d lidar," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.* IEEE, 2017, pp. 2601–2608.

[2] X. Du and K. K. Tan, "Comprehensive and practical vision system for self-driving vehicle lane-level localization," *IEEE Trans. Image Process.*, vol. 25, no. 5, pp. 2075–2088, 2016.

[3] M. Manz, T. Luettel, F. von Hundelshausen, and H.-J. Wuensche, "Monocular model-based 3d vehicle tracking for autonomous vehicles in unstructured environment," in *Proc. IEEE Int. Conf. Robotics Automation*. IEEE, 2011, pp. 2465–2471.

[4] C. Yang, G. Cheung, and V. Stankovic, "Estimating heart rate and rhythm via 3d motion tracking in depth video," *IEEE Trans. Multimedia*, vol. 19, no. 7, pp. 1625–1636, 2017.

[5] C. Zhang and T. Chen, "Efficient feature extraction for 2d/3d objects in mesh representation," in *Proc. Int. Conf. Image Process.*, vol. 3. IEEE, 2001, pp. 935–938.

[6] M. M. Bronstein and I. Kokkinos, "Scale-invariant heat kernel signatures for non-rigid shape recognition," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*. IEEE, 2010, pp. 1704–1711.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, 2016, pp. 770–778.

[8] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, 2015, pp. 1912–1920.

[9] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.* IEEE, 2015, pp. 922–928.

[10] M. Simonovsky and N. Komodakis, "Dynamic edgeconditioned filters in convolutional neural networks on graphs," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, 2017.

[11] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proc. IEEE Int. Conf. Comput. Vision*, 2015, pp. 945–953.

[12] A. Kanezaki, Y. Matsushita, and Y. Nishida, "Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, 2018.

[13] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, vol. 1, no. 2, p. 4, 2017.

[14] R. Klokov and V. Lempitsky, "Escape from cells: Deep kd-networks for the recognition of 3d point cloud models," in *Proc. IEEE Int. Conf. Comput. Vision*. IEEE, 2017, pp. 863–872.

[15] J. Li, B. M. Chen, and G. H. Lee, "So-net: Self-organizing network for point cloud analysis," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, 2018, pp. 9397–9406.

[16] K. Zhang, M. Hao, J. Wang, C. W. de Silva, and C. Fu, "Linked dynamic graph cnn: Learning on point cloud via linking hierarchical features," *arXiv preprint arXiv:1904.10014*, 2019.

[17] X. Liu, Z. Han, Y.-S. Liu, and M. Zwicker, "Point2sequence: Learning the shape representation of 3d point clouds with an attention-based sequence to sequence network," in *Proc. AAAI Conf. Artificial Intell.*, vol. 33, 2019, pp. 8778–8785.

[18] Y. Liu, B. Fan, S. Xiang, and C. Pan, "Relation-shape convolutional neural network for point cloud analysis," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, 2019, pp. 1–10.

[19] N. Sedaghat, M. Zolfaghari, E. Amiri, and T. Brox, "Orientation-boosted voxel nets for 3d object recognition," in *Proc. British Mach. Vision Conf.*, 2017. [Online]. Available: http://lmb.informatik.uni-freiburg.de/Publications/2017/SZB17a

[20] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "Generative and discriminative voxel modeling with convolutional neural networks," *arXiv preprint arXiv:1608.04236*, 2016.

[21] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling," in *Proc. Ann. Conf. Advances in Neural Info. Process. Syst.*, 2016, pp. 82–90.

[22] G. Riegler, A. O. Ulusoy, and A. Geiger, "Octnet: Learning deep 3d representations at high resolutions," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, vol. 3, 2017.

[23] S. Kumawat and S. Raman, "Lp-3dcnn: Unveiling local phase in 3d convolutional neural networks," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, 2019, pp. 4903–4912.

[24] Y. Feng, Z. Zhang, X. Zhao, R. Ji, and Y. Gao, "Gvcnn: Group-view convolutional neural networks for 3d shape recognition," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, June 2018.

[25] T. Yu, J. Meng, and J. Yuan, "Multi-view harmonized bilinear network for 3d object recognition," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, 2018, pp. 186–194.

[26] E. Johns, S. Leutenegger, and A. J. Davison, "Pairwise decomposition of image sequences for active multi-view recognition," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*. IEEE, 2016, pp. 3813–3822.

[27] C. Xu, Z. Li, Q. Qiu, B. Leng, and J. Jiang, "Enhancing 2d representation via adjacent views for 3d shape retrieval," in *Proc. IEEE Int. Conf. Comput. Vision*, 2019, pp. 3732–3740.

[28] Z. Han, M. Shang, Z. Liu, C.-M. Vong, Y.-S. Liu, M. Zwicker, J. Han, and C. P. Chen, "Seqviews2seqlabels: Learning 3d global features via aggregating sequential views by rnn with attention," *IEEE Trans. Image Process.*, 2018.

[29] J. Jiang, D. Bao, Z. Chen, X. Zhao, and Y. Gao, "Mlvcnn: Multi-loop-view convolutional neural network for 3d shape retrieval," 2019.

[30] Z. Han, H. Lu, Z. Liu, C.-M. Vong, Y.-S. Liua, M. Zwicker, J. Han, and C. P. Chen, "3d2seqviews: Aggregating sequential views for 3d global feature learning by cnn with hierarchical attention aggregation," *IEEE Trans. Image Process.*, 2019.

[31] X. He, T. Huang, S. Bai, and X. Bai, "View n-gram network for 3d object retrieval," in *Proc. IEEE Int. Conf. Comput. Vision*, 2019, pp. 7515–7524.

[32] Z. Yang and L. Wang, "Learning relationships for multi-view 3d object recognition," in *Proc. IEEE Int. Conf. Comput. Vision*, 2019, pp. 7505–7514.

[33] Y. Liu, Q. Fu, Y. Liu, and X. Fu, "A distributed computational cognitive model for object recognition," *Science in China Series F: Information Sciences*, vol. 56, no. 9, pp. 1–13, 2013.

[34] Y. Xu, C. Zheng, R. Xu, and Y. Quan, "Deeply exploiting long-term view dependency for 3d shape recognition," *IEEE Access*, vol. 7, pp. 111 678–111 691, 2019.

[35] C. Wang, M. Pelillo, and K. Siddiqi, "Dominant set clustering and pooling for multi-view 3d object recognition." in *Proc. British Mach. Vision Conf.*, 2017.

[36] C. Zheng, Y. Xu, R. Xu, H. Chi, and Y. Quan, "Multi-view rank pooling for 3d object recognition," in *Proc. IEEE Visual Comm. Image Process.* IEEE, 2019, pp. 1–4.

[37] C. Ma, Y. Guo, J. Yang, and W. An, "Learning multi-view representation with lstm for 3d shape recognition and retrieval," *IEEE Trans. Multimedia*, 2018.

[38] G. Dai, J. Xie, and Y. Fang, "Siamese cnn-bilstm architecture for 3d shape representation learning." in *Proc. Int. Joint Conf. Artificial Intell.*, 2018, pp. 670–676.

[39] X. Wei, R. Yu, and J. Sun, "View-gcn: View-based graph convolutional network for 3d shape analysis," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, June 2020.

[40] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Ann. Conf. Advances in Neural Info. Process. Syst.*, 2017, pp. 5105–5114.

[41] Y. Shen, C. Feng, Y. Yang, and D. Tian, "Mining point cloud local structures by kernel correlation and graph pooling," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, vol. 4, 2018.

[42] Z. Xie, J. Chen, and B. Peng, "Point clouds learning with attention-based graph convolution networks," *Neurocomputing*, vol. 402, pp. 245 – 255, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231220304732

[43] S. Bai, X. Bai, Z. Zhou, Z. Zhang, and L. Jan Latecki, "Gift: A real-time and scalable 3d shape search engine," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, 2016, pp. 5023–5032.

[44] K. Sfikas, I. Pratikakis, and T. Theoharis, "Ensemble of panorama-based convolutional neural networks for 3d model classification and retrieval," *Comput. Graphics*, 2017.
[45] K. Sfikas, T. Theoharis, and I. Pratikakis, "Exploiting the panorama representation for convolutional neural network classification and retrieval," in *Eurographics Workshop on 3D Object Retrieval*, vol. 8. The Eurographics Association, 2017.
[46] L. Yi, H. Su, X. Guo, and L. J. Guibas, "Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation." in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, 2017, pp. 6584–6592.
[47] D. Boscaini, J. Masci, E. Rodolà, and M. Bronstein, "Learning shape correspondence with anisotropic convolutional neural networks," in *Proc. Advances in Neural Info. Process. Syst.*, 2016, pp. 3189–3197.
[48] J. Xie, Y. Fang, F. Zhu, and E. Wong, "Deepshape: Deep learned shape descriptor for 3d shape matching and retrieval," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, June 2015.
[49] V. Hegde and R. Zadeh, "Fusionnet: 3d object classification using multiple data representations," *arXiv preprint arXiv:1607.05695*, 2016.
[50] S. Belongie, J. Malik, and J. Puzicha, "Shape context: A new descriptor for shape matching and object recognition," in *Proc. Advances in Neural Info. Process. Syst.*, 2001, pp. 831–837.
[51] Z. Zhang, H. Lin, X. Zhao, R. Ji, and Y. Gao, "Inductive multi-hypergraph learning and its application on view-based 3d object classification," *IEEE Trans. Image Process.*, 2018.
[52] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition*, 2018, pp. 7794–7803.
[53] H. You, Y. Feng, R. Ji, and Y. Gao, "Pvnet: A joint convolutional network of point cloud and multi-view for 3d shape recognition," in *Proc. ACM Int. Conf. Multimedia*, ser. MM '18. New York, NY, USA: ACM, 2018, pp. 1310–1318. [Online]. Available: http://doi.acm.org/10.1145/3240508.3240702
[54] J.-C. Su, M. Gadelha, R. Wang, and S. Maji, "A deeper look at 3d shape classifiers," in *Proc. European Conf. Comput. Vision*, 2018, pp. 0–0.
[55] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. IEEE Int. Conf. Comput. Vision*, 2015, pp. 1026–1034.
[56] H. S. Manolis Savva, Fisher Yu, "Large-scale 3d shape retrieval from shapenet core55," https://shapenet.cs.stanford.edu/shrec17/.

Yong Xu received the B.S., M.S., and Ph.D. degrees in mathematics from Nanjing University, Nanjing, China, in 1993, 1996, and 1999, respectively. He is currently a Professor with the School of Computer Science and Engineering. His current research interests include image analysis, video recognition, and image quality assessment.

Chaoda Zheng received the B.Eng degree in Computer Science from South China University of Technology in 2017. He continues his research life as a Ph.D candidate. His research interests include computer vision, 3D data processing, and object recognition.

Ruotao Xu received the B.Eng degree in Computer Science from South China University of Technology in 2015. He continues his research life as a Ph.D candidate. His research interests include computer vision, image processing, and sparse coding.

Yuhui Quan received the Ph.D. degree in Computer Science from South China University of Technology in 2013. He worked as a postdoctoral research fellow in Mathematics at National University of Singapore from 2013 to 2016. He is currently an associate professor at School of Computer Science and Engineering in South China University of Technology. His research interests include computer vision, image processing and sparse representation.

Haibin Ling received the B.S. and M.S. degrees from Peking University in 1997 and 2000, respectively, and the Ph.D. degree from the University of Maryland, College Park, in 2006. From 2000 to 2001, he was an assistant researcher at Microsoft Research Asia. From 2006 to 2007, he worked as a postdoctoral scientist at the University of California Los Angeles. In 2007, he joined Siemens Corporate Research as a research scientist. From 2008 to 2019, he worked as a faculty member of the Department of Computer Sciences at Temple University. In fall 2019, he joined Stony Brook University as a SUNY Empire Innovation Professor in the Department of Computer Science. His research interests include computer vision, augmented reality, medical image analysis, and human computer interaction. He received Best Student Paper Award at ACM UIST in 2003, and NSF CAREER Award in 2014. He serves as Associate Editors for several journals including IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI), Pattern Recognition (PR), and Computer Vision and Image Understanding (CVIU). He has served as Area Chairs for CVPR 2014, 2016, 2019 and 2020.