IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 3, 2020, accepted January 17, 2020, date of publication January 21, 2020, date of current version February 6, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2968506

3D Object Classification Using a Volumetric Deep Neural Network: An Efficient Octree Guided Auxiliary Learning Approach

A. A. M. MUZAHID[1,2], WANGGEN WAN[1,2], (Senior Member, IEEE), FERDOUS SOHEL[3], (Senior Member, IEEE), NAIMAT ULLAH KHAN[1,2], OFELIA DELFINA CERVANTES VILLAGÓMEZ[4], AND HIDAYAT ULLAH[1,2]

[1]School of Communications and Information Engineering, Shanghai University, Shanghai 200444, China
[2]Institute of Smart City, Shanghai University, Shanghai 200444, China
[3]College of Science, Health, Engineering and Education, Murdoch University, Murdoch, WA 6150, Australia
[4]Computing, Electronics and Mechatronics Department, Universidad de las Américas Puebla, Puebla 72810, Mexico

Corresponding author: A. A. M. Muzahid (muzahid@shu.edu.cn)

This work was supported by the Project of Shanghai Science and Technology Committee under Grant 18510760300.

ABSTRACT We consider the recent challenges of 3D shape analysis based on a volumetric CNN that requires a huge computational power. This high-cost approach forces to reduce the volume resolutions when applying 3D CNN on volumetric data. In this context, we propose a multiorientation volumetric deep neural network (MV-DNN) for 3D object classification with octree generating low-cost volumetric features. In comparison to conventional octree representations, we propose to limit the octree partition to a certain depth to reserve all leaf octants with sparsity features. This allows for improved learning of complex 3D features and increased prediction of object labels at both low and high resolutions. Our auxiliary learning approach predicts object classes based on the subvolume parts of a 3D object that improve the classification accuracy compared to other existing 3D volumetric CNN methods. In addition, the influence of views and depths of the 3D model on the classification performance is investigated through extensive experiments applied to the ModelNet40 database. Our deep learning framework runs significantly faster and consumes less memory than full voxel representations and demonstrate the effectiveness of our octree-based auxiliary learning approach for exploring high resolution 3D models. Experimental results reveal the superiority of our MV-DNN that achieves better classification accuracy compared to state-of-art methods on two public databases.

INDEX TERMS 3D shape analysis, object classification, convolutional neural network, DNNs, volumetric CNN.

I. INTRODUCTION

The rapid development of consumer depth cameras, 3D acquisition and scanning devices make it easier to obtain a 3D view of a real object that is tremendously increasing many real world applications [1] within the areas of computer vision, online gaming, films, TV, engineering project modeling, biology, military research and many more applications in the field of visual reality. Among them, 3D object recognition and classification for autonomous vehicles (i.e., self-driving car) to avoid collisions, 3D object detection for warehouse–robots to restore and collect products for shipping and 3D retrieval for searching target objects in large databases are currently amongst the most demanding applications. However, object classification is a key application of computer vision areas, whereas vision systems are built using the theory of artificial intelligence (AI) systems that machines can recognize what is perceived similar to the human visual system. To develop a computer vision framework a neural network (can be developed using machine learning or deep learning algorithm) is trained with a comparative class of objects. During the training, final feature neurons and weights are iteratively generated and stored as a trained model.

During model testing, an unseen object is given to the neural network, then it generates a feature map using the trained model.

The classification result outputs the prediction score of category information of the object. The final target object is

The associate editor coordinating the review of this manuscript and approving it for publication was Mehul S. Raval.

classified by the highest probability ($\leq$1) of classes among the trained objects. This phenomenon is referred to as object classification in computer vision. Figure 1 depicts a basic block diagram of CNN-based 3D object classification. The research in object classification in the two-dimensional field of image processing started over a few decades ago and achieved very successful results employing deep convolutional learning methodology. Deep learning consists of a large neural network including many hidden layers to learn a complicated and complex model in a hierarchical or multilayer manner.

The design of a vision system with neural networks for an ordered sample 2D image to an unordered complex 3D model by typical 3D data formats (i.e., triangle mesh or point cloud) is usually deemed a more challenging task. Research on deep learning for 3D Shape analysis [2] including object classification excelled with the introduction of the ModelNet [3] dataset. Recently, several deep learning approaches have been proposed for 3D object classification, detection and recognition purposes using various tactics to make regular sampled input features to the networks [4]–[8]. The most commonly used 3D feature representations for classification are volumetric, shape-descriptors, 3D projection, RGB-D, multi-view, point clouds, 3D graphs, and meshes [9]–[11].

To capture the full geometric features of a 3D object, volumetric representation provides very detailed information with regular grid-style data. The most common practice of volumetric representation is voxel to describe the 3D model in three-dimensional distributions. The very first attempt of deep learning (DL) to object classification was proposed by Wu et al. and named 3D ShapeNets [4] by deep belief networks [12], [13]. Therefore, due to the encouraging results of 3D ShapeNets, several papers have been published that consider volume-based DNNs [5], [6], [14] for object classification. Despite the great and promising consequences of such volumetric deep learning approaches [15], these kinds of networks are challenging to train in terms of high computational cost. In comparison to 2D image resolution (250$\times$250) with 3D voxel resolution (32$\times$32$\times$32), the computational cost increases cubically [16] as voxel represents both occupied and nonoccupied parts of the object. The most straightforward efficient way of volumetric representations is octree-based deep learning approaches. Octree has a long history in shape representations [17]–[19], and it is becoming popular in CNN-based 3D shape analysis because of its compatibility in obtaining fine detail of a 3D model [19], [20]. Therefore, the best 3D classification results, up until recently, have been achieved by 2D image-based deep learning convolutional neural networks, where 2D rendered images are obtained from 3D shape as a projection of 3D data [22], [23]. However, how to determine the number of perspective views to cover global features is unclear and losing the inherent geometric features of the 3D model are the main constraints of 2D-based CNN for 3D object analysis.

Considering the deep neural network and computational limitations of voxel representations, this paper proposes an effective octree-based auxiliary learning approach for 3D object classification based on a multiorientation volumetric deep neural network (MV-DNN). The availability of 3D scanning tools and the advent of computational power led us to investigate high resolution data to explore rich global features in DNNs for 3D object classification. However, deep learning (DL) requires a large training dataset and due to the limited training 3D dataset, we utilize an augmentation approach uniformly using a well-known multiorientation 3D object on the scale of 360 degrees around the horizontal axis.

We implement GPU-based octree data structures to design high resolution supported CNN that consumes lower memory. The hierarchical octree data structure is similar to the quadtree structure [24], and CNN is performed just on sparse octants where computational cost increases quadratically. Our contributions in this paper are threefold and can be summarized as follows:

- We propose to use auxiliary learning on octree structured 3D data to the subvolume parts of a 3D object. It aims to learn edge detail information and increase the prediction ability by observing some parts of an object.
- We propose to preserve all octants information to a certain partition level based on the predefined input voxel resolution to store high-precision contour features at the beginning of the octree partition. This effective feature helps to enhance the geometric resolution to the convolution filter and improve the classifier performance compared with conventional octree representation.
- We conduct extensive experiments to determine the influence of input volume resolutions and multiorientation effects on the classification task. Experimental results show that the classification performance of our MV-DNN improves gradually with volume resolutions and augmented samples.

In addition, our proposed MV-DNN is a GPU-based volumetric deep convolutional neural network that directly inputs octree structures of a 3D object. Our MV-DNN learns complex hierarchical features of a 3D object in a supervised manner. To implement auxiliary learning, we divide the bounding box of a 3D object into subvolume parts utilizing layer slicing methodology. Classification experiments are evaluated on the ModelNet40 and ModelNet10 datasets (contains 3D CAD models), and our proposed MV-DNN outperforms the following state-of-art methods.

The rest of this paper is organized as follows. An overview of deep learning approaches on different 3D representations is presented in Section 2. Our proposed method and network architecture are provided in Section 3. The analysis of the experimental results is conducted in Section 4. Finally, in Section 5, we provide conclusions and present the future work of our research.

II. RELATED WORK

During the last few years, the DL approach achieved a very satisfactory result in the 2D computer vision field. From that point forward, DL attracts 3D computer vision researchers

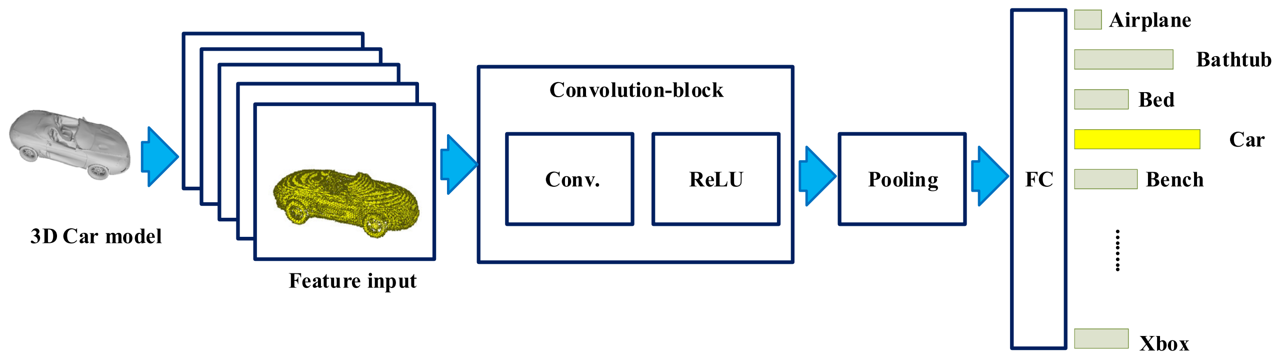

FIGURE 1. A basic block diagram of CNN-based 3D object classification.

to learn the complex structure of 3D data. Until recently, DL research has been directed toward 3D object classification, which is far less than that of 2D areas. In this section, we briefly review some recent DL advances for 3D object classification. We pick some of the most recent DL frameworks and then discuss their 3D representation techniques, including data preparation and training details.

A. DEEP NEURAL NETWORKS FOR 3D OBJECT CLASSIFICATION

Selecting or extracting features is the key task for using a neural network for computer vision application. Unlike 2D feature extraction, a 3D model is complex by nature. There is no ideal or universal method for learning a 3D model. However, several feature extraction methods have been developed for 3D objects [25]–[28]. Based on the representation methods, the neural networks are designed for different applications, such as 3D model classification, retrieval, and segmentation. In this section, we focus on object classification methods only.

Wu et al [4] presented 3DShapeNets which is the earliest deep learning approach on volumetric 3D data. 3DShapeNets was developed with a convolutional deep belief network (CDBN) for 3D shape analysis including classification. The geometry of the 3D shape was represented on a 3D voxel grid. The grid size was $30 \times 30 \times 30$ which can be compared with a 2D image resolution of 165×165. The CDBN consists of 6 layers with thousands of hidden parameters. The classification experiment was conducted on their ModelNet40 dataset, where each 3D model was augmented by rotating 30 degrees (12 poses/model) in arbitrary poses. The linear SVM method was used to train the network and it costs approximately 48 hours on NVIDIA Tesla K40c GPU. The classification accuracy was 77.32% for 40 categories.

The 3D ShapeNets further improved by Maturana and Scherer [14] who proposed VoxNet, which is a 3D convolutional neural network, by integrating a volumetric occupancy grid representation. VoxNet experimented on three different data sources where the classification accuracy on the ModelNet40 dataset improved to 83%. VoxNet also demonstrated several experiments to determine the effects of augmentation, voting, occupancy grids, and multiresolutions. Among them, training the network on the augmented 3D model and using the voting method provides high classification accuracy on the Modelnet40 dataset. Unlike 3D ShapeNets, VoxNet uses a pooling layer, and dropout regularization is also added at the end of each layer. Stochastic gradient descent (SGD) with momentum was used to train networks. However, both 3D ShapeNets and VoxNet are limited to small voxel volumes and consumed high memory and time.

To avoid the limitation of voxel representations, deep learning on 3D sparse data using octree structured 3D data provides very promising results for 3D shape analysis [20], [21]. Riegler at el. proposed OctNet [16] for 3D object classification on ModelNet10 dataset [3]. The input 3D object was hierarchically partitioned into an octree structure, and pooled features were stored in the leaf nodes. The authors also observed that the higher probability of activation was made closer to the object boundary. The input triangle meshes were converted to octree grids of several resolutions (8^3 to 256^3), and each network consists of a number of blocks where each block consists of two convolutional layers (stride of 1) and one max-pooling layer with a stride of 2. OctNet achieved the state-of-art performance on a low-scale Modelnet10 database.

Considering massive image databases and advances in image descriptor, Su et al. proposed multiview convolutional neural networks (MVCNN) [22] where 3D models were rendered to the 2D image under a perspective projection. Every 3D model was captured with 12 rendered views by placing 12 virtual cameras around the 3D model. The network was pretrained on the ImageNet dataset and then fine-tuned on the ModelNet40 dataset for 3D classification. Linear SVM was used to train the network where SVM was applied on a 12-view of a 3D object at the test time, and the highest sum reflects the class of the object. MVCNN achieved 89.9% classification accuracy on the ModelNet40 dataset. Recently, Ma et al [29] introduced an LSTM module to learn low-level features by CNN and outperform MVCNN for the classification task. In the following 3D2SeqViews is proposed by Han et al [30], where VGG was finetuned to encode low-level features and sequential views were aggregated in a hierarchical attention fashion to generate globalfeature. It performs better than LSTM for the 3D classification task. However, the selection of number of views remains an open issue for imaged-based 3D object classification. Inspired by multiview, Qi et al. [31] proposed volumetric and multiview CNNs for object classification. Instead of multiview, Qi et al. proposed using multiorientation pooling by concatenating fc7, and data augmentation was performed in both azimuth and elevation. The volumetric approach by Qi et al. was mainly focused on network design considering the higher resolution and proposed several volumetric CNNs, including long anisotropic kernel and multiorientation pooling strategies. Multilayer perceptron convolution layers [32] were adopted to avoid overfitting and increase feature extraction capability. Each multilayer perceptron contains three convolutional layers and a rectified linear unit (ReLU) layer. All proposed networks were trained using augmented 3D models where each 3D model was captured with 20 views and applied to both azimuth and elevation rotation. The best average class accuracy of 91.4% was achieved using multiorientation pooling strategy To reduce the computational cost of voxel representation, NormalNet [10] and LP-3DCNN [33] reduced the number of parameters by introducing RCC and ReLPV modules respectively with better classification accuracy than state-of-art-methods. Recently, Khan et al. proposed an unsupervised primitive GAN [34] model, where generated volumetric features were used to train the network on ModelNet10 dataset for the classification task but tested on ModelNet10 and ModelNet40 dataset. The accuracy of this unsupervised method is inferior to supervised methods.

Hegde et al. proposed FusionNet [35] by merging voxel and 2D-based representations in a system. They proposed voxel-based two volumetric CNNs where $30 \times 30 \times 30$ voxel resolution was used in V-CCN I (3-convolution and 2-FC layers) with 60 orientations of each 3D model. Similar to AlexNet [36], different sizes of filters (1×1, 3×3 and 5×5) were concatenated in V-CNN II with similar inputs of V-CCN I. Both V-CNN I and V-CNN II used a weight-sharing approach over 60 orientations. The accuracy results of V-CNN I and V-CNN II were 82.41% and 80.63%, respectively, on the ModelNet40 dataset. Above all volumetric-based voxel representations faced the same issues of GPU memory that limits using the high resolution.

The point cloud 3D data format also has gained popularity in 3D shape analysis [37]–[39] because of the availability of a point cloud 3D scanner. Charles et al. [37] proposed a DL-based 3D classifier named PointNet using point cloud representation directly as input data to the network. PointNet was further improved by Qi et al. [37] who proposed PointNet++, where PointNet was executed recursively to the nested input point set. You et al. proposed PVNet [8] with improved classification results using point cloud and 2D rendered images simultaneously as input features to CNNs. Recently, ComposeCaps is introduced in 3DCapsule [40] framework by Cheraghian et al. to explore meaningful features for point cloud classification To learn high-level relation expressions, Liu et al proposed RS-CNN [41] and achieved state-of-art-result for point cloud classification.

Yang et al. proposed unsupervised DL networks called FoldingNet [42] and introduced a folding-based auto decoder on point cloud data. Another unsupervised approach was introduced by Li et al. [43] proposed SO-Net for point cloud shape analysis. However, the point cloud was associated with unordered 3D point data, so typically it required a transformation to make data in order and fed further to 3D shape analysis networks. In the definition of Euclidean and non-Euclidean data format, the point cloud carries a small Euclidean subset that has global parametrizations; however, researchers mostly treated point clouds as nonvolumetric representations [9]

III. METHOD

Due to the computational limitation of voxel representation in deep learning, we likewise consider the octree as volumetric portrayals in view of its monetary computational qualities which has been used in different 3D data application [44]–[47]. Octree representations allow for deeper networks with high voxel resolutions for the 3D object classification task. Our investigation in this paper focuses on improving the performance of volumetric CNN and reducing the performance gap between volumetric and nonvolumetric representations for 3D object classification.

Riegler et al. [16] proposed OctNet for 3D object classification with a hybrid octree representation of a 3D CAD model. Instead of the regular octree structure, OctNet restricts the maximal depth of octree to 3, namely shallow octree. To increase the input resolution, multiple shallow octrees are required, e.g., two shallow octrees are required to generate a resolution of 16^3 voxels. Apart from conventional volumetric CNN, OctNet accepts high-resolution input and improves classification accuracy significantly on ModelNet10 dataset only.

Although we also have considered octree as a volumetric representation of the 3D CAD model but there are distinct differences between OctNet and our technique in terms of octree representation and developing the network. Instead of restricting the maximal octree depth of OctNet, we follow the standard octree partition which generates octrees continuously until the maximum depth is reached. In addition, we proposed to store all octants to a certain depth (e.g., 2 or depth 3) depending on the maximal octree depth for reserving orientation information and adjusting all features from various octree depths to produce more discriminative features. In addition, our proposed MV-DNN is more advanced and obtains better classification accuracy on ModelNet40 and ModelNet10 dataset than OctNet (see in Table 3).

In the following, we describe our proposed data structure and the architecture of our multiorientation volumetric neural network in detail.

A. VOLUMETRIC OCTREE REPRESENTATION

To make an effective hierarchical 3D data structure, we propose a crossover octree structure by holding all octants' features up to a specific depth depending on the predefined volume resolution. First, we put a 3D model into a bounding cube box, then it is subdivided into 8 equal pieces according to first order octree (depth 1st). This subdivision process is conducted until the predefined octree depth (d) is reached. The maximum octree depth is equal to target volumetric resolution (i.e., $d = 5$ for 32^3 voxel resolution). However, subdivision only occurs on the occupied parent's octants, and the resulting octants are known as leaf or child octants. A shuffle key (S) is generated for each octant according to their corresponding octree depth (d) to ensure their position in the 3D cube. The l-bit string key can be written as:

$$S_l = x_1y_1z_1x_2y_2z_2x_3y_3z_4 \ldots\ldots\ldots . x_ly_lz_l \tag{1}$$

where $x_iy_iz_i \in \{0, 1\}$, $(i = 1, 2, \ldots, l)$, represent the position of its parent octants. All octants are collected according to their corresponding shuffle key (S_l) in ascending format in each depth and stored as a one-dimensional vector. The size of the shuffle key (S_l) is a 32-bit integer value. The average value of each leaf node is stored in their parent nodes as features of the 3D object computed as:

$$L_n = {}^1\!/_n \sum_{n=1}^{n_{\max}} v_c(1-n), \tag{2}$$

where L_n denotes leaf node value, $n_{\max}$ is equal to 8 and refers to the number of nodes in each leaf, and v_c is the value of each child octant.

To extract a high-level feature, we propose to limit the full layer octree to a certain level of depth based on the predefined voxel resolution. The reservation of full voxels up to certain depths may increase the size of input features and computational cost slightly more than the regular octree structure, but the lower computational cost is still required as compared to the full voxel method (see Table 4).

The reservation of full layer features is determined by the following equations:

$$O_{FL}^{\varphi_n} = \begin{cases} 2 & \text{if } \varphi_n \le V_R = 4 \\ 3 & \text{else} \end{cases} \tag{3}$$

where $O_{FL}^{\varphi_n}$ represents the expected full layer octree depth calculated by Eq. (3), φ_n is the current octree depth in the loop, and V_R is the predefined input volume resolution. If we consider the volume resolution to be higher than 16^3, then the full layer features until depth 3rd will be stored unless the number of full layer values is 2.

To find the relationship between parent and leaf nodes, a label p_k is assigned where the value of p_k is a positive integer number to all nonempty or 0 to empty nodes. All labels of a specific depth (d^{th}) are stored in a vector L_{dk} can be written as:

$$L_{dk} = \{p_1p_2p_3p_5p_6p_7p_8p_9p_{10}p_{11}p_{12} \cdots\cdots p_{dk-1}p_{dk}\} \tag{4}$$

where k is the total number of octants in the d^{th} octree. Only nonempty nodes are further subdivided, and the storing order of the leaf nodes is sequentially made according to their parents' nodes. Figure 2a shows a very detailed map of a generated octree in 2D view at different depths for a 3D airplane model. We can see that the airplane model occupied all the nodes (4-node for the 2D view) in 1^{st} depth (d_1) and is subdivided into 32-leaf nodes in 2nd depth; however, we can see only 16 new leaf nodes in the 2D view. In Figure 2a, shuffle key maps of d_2 show that the airplane occupies only 6 nodes among 16 are 1, 3, 4, 6, 9 and 12, respectively. These occupied nodes are labeled with a positive integer number (i.e., 1, 2, 3..), and 0 will be added to the empty nodes (Figure 2b, L_2). In the following depth-3, these 6-occupied nodes will be further subdivided into 24 nodes. This is the benefit of octree representation, where the octree resolution always changes and reduces the computation burden by taking only occupied node data rather than full voxels. However, the octree subdivision originally takes place in 8 new nodes where Figure 2a shows only 4 nodes because of considering the 2D view for the sake of simplicity. Unlike max-pooling [16], the input signal is formed by averaging the normal vector of the leaf nodes because the performance of the normal signal is better than a binary signal [9]. The size of the input vector is an equal length of leaf nodes. This input vector is directly fed to our proposed network.

B. THE ARCHITECTURE OF MV-DNN FRAMEWORK

Our proposed convolutional deep neural network introduces auxiliary learning using octree-structured volumetric input to improve the level prediction rate over time. The auxiliary learning is inspired by the perceptual human vision system, as we have an ability to recognize an object by observing some parts of it. In similar way, the network is forcibly trained to learn feature on subvolume parts to predict object label. So, the output of the subvolume classification is strongly correlated with the main task. Theoretically, overfitting is the main barrier of volumetric CNN (3D CNN) and if it overfits to training data then it stops learning. In that case, auxiliary learning continues to learn and it is difficult to overfit because of exploiting only subvolume parts of the input. Our multiorientation deep neural network (MV-DNN) is a convolution neural network consisting of the auxiliary learning module at the top (see Figure 3). The main blocks of our network are multilayer octree convolution (MOC), max-poling (PL), typical convolution layer (TCN) and fully connected layer (FC). Each MOC block consists of two convolution layers in a feed-forward connection. These two consecutive convolution layers will produce meaningful and precise features from low level input feature using a few hundred parameters only (see Table 1). The MOC block will help to make a feature in hierarchical order, reduce underfitting the network and increase feature learnability. MOC uses a fixed filter (conv kernel) of size 3 but the number of filters will increase twice in the following convolution layers to extract high-level features. Figure 3 demonstrates that our auxiliary learning model for octree depth 5th contains only two MOC blocks and the list of parameters is provided in Table 1. The number of MOC layers decreases or increases according to lower and higher octree depths to adjust the input volume size for different depths, respectively. The framework of depth 5th is extended

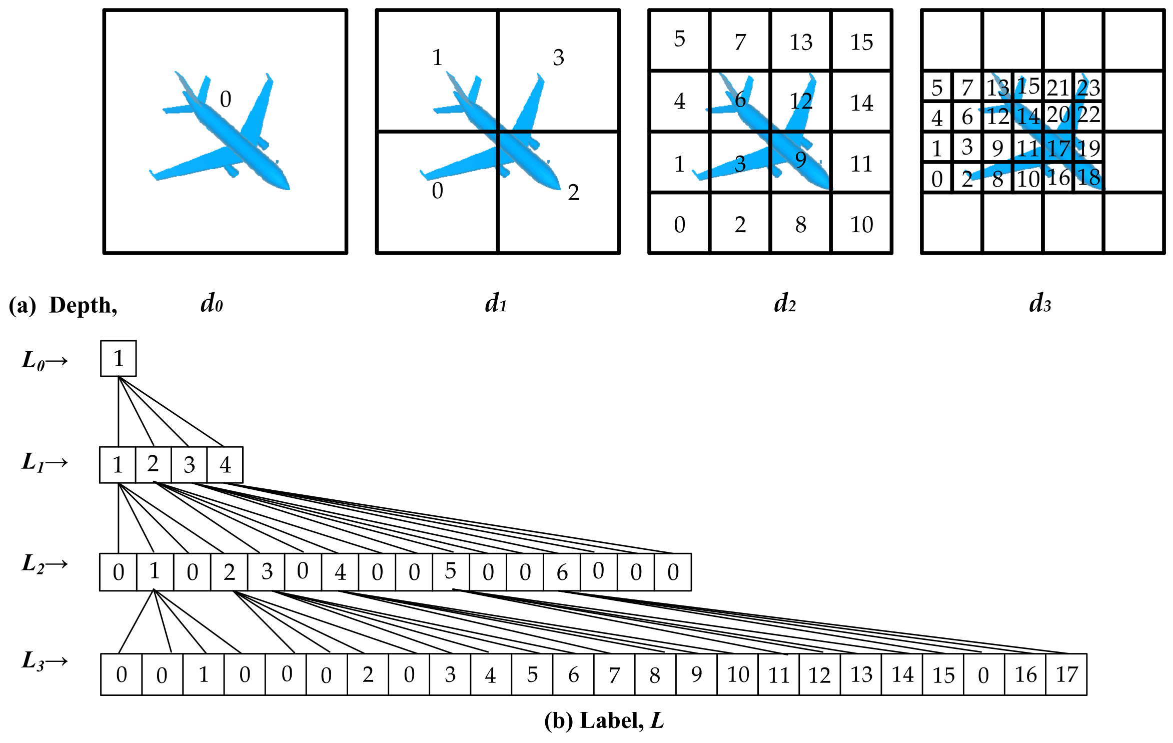

FIGURE 2. Octree illustration of 3D model in 2D view (a) octree representation according to depth (b) Labeling of nodes to build parent-child (leaf) relationship and decide which nodes will be further subdivided.

for depth 6th and 7th by adding one and two MOC blocks respectively at the bottom.

During convolution operation, the octree depth will be unchanged when striding of 1. So, each MOC block will breakdown the input complex data into smaller and smaller features and produce invariant global features through long range convolution operation without reducing the octree depth.

The output features of each convolution layer are generalized with a batch normalization layer and all neurons are activated by passing through a rectified linear unit (ReLU). The convolution operation can be written as follows:

$$g_c(U) = \sum_n \sum_x \sum_y \sum_z W^n_{xyz}.M^{(n)}_d(U_{xyz}), \quad (5)$$

where g_c is the convolution operator, U_{xyz} represents a neighboring octant of U and convolution results of all octants at d^{th} depth are recorded in a vector M^n_d as n^{th} channel feature.

All property vectors of a 3D object S_d, L_d and M_d are concatenated to form a super octree directly can be represented as respectively S^*_d, L^*_d and M^*_d, which will be used to train the CNN. The features will be downsampled only by a pooling operation where max-pooling is employed with a kernel size of 2^3 and a stride of 2. Pooling operation $\psi^{mp}_{i,j,k}$ will select the maximum octant value from every 8 adjacent leaf octants can be written from Eq. (5) as

$$\psi^{mp}_{i,j,k} = \underset{x,y,z\in\{0,1\}}{max\_pooling} \sum_{o}^{n-1} (M^{(n-1)}_d U_{2i+x,2j+y,2k+z}). \quad (6)$$

To apply auxiliary learning, we divide voxel features inside the network according to a 3-dimensional data tensor. However, the octree to voxel (O-V) layer converts octree data (8×1) to voxel form with 1^{st} depth (2×2×2). The reason for making octree to voxel conversion in 1^{st} depth because of slicing the voxel can be divided into two equal points, whereas the model can learn with 50% features of the 3D object at 1^{st} slicing point. Every slice operation will make two new slice layers where 3 consecutive slices will create 8 new slicing layers (or 8 subdivisions of the voxel volume). These 8 subdivision layers predict the object class by learning features from the subpart of an object.

In contrast, the main branch of the classifier includes three fully connected (FC) layers (4096, 512, N). The last FC is used as a classifier, where N is the number of object classes. A single FC layer can be compared to several 1-dimensional convolution layers, but the main difference between them is making a decision that only FC can do. A single typical convolution (TCN) layer with kernel size of 1 is added after the O-V layer to increase the feature transformation with little cost, and the flattened layer (after O-V) shapes the feature vector in a simple form $n \times c \times h \times w$ to $n \times (c \times h \times w)$. To reduce overfitting, we added a dropout layer between FC-1 and FC-2 with a 50% drop ratio that works only during the training session. The object class label is predicted by FC-3, and the highest prediction among N nodes is the final class of an object.

We calculate softmax loss to evaluate the network performance with respect to the negative log-likelihood function.

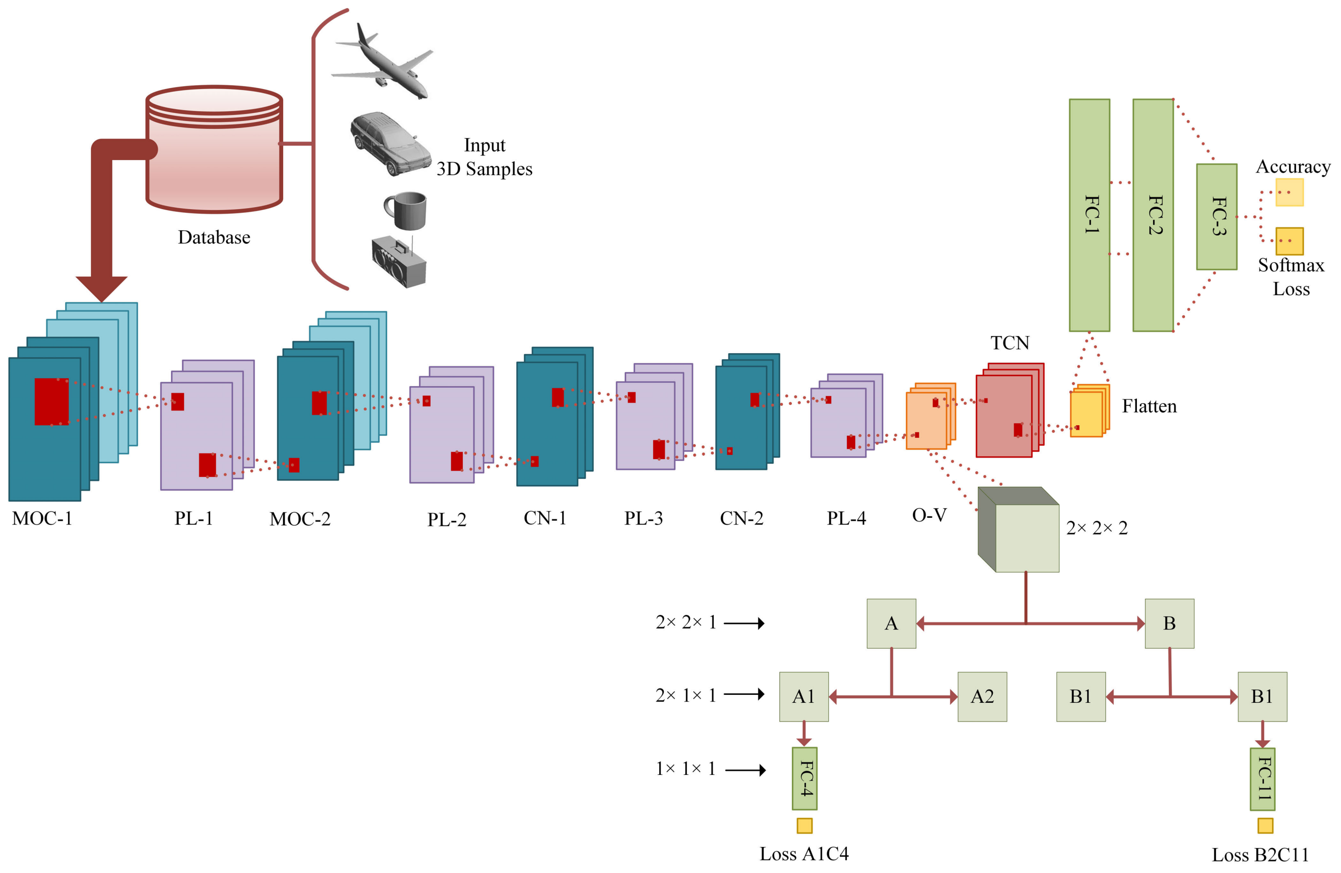

FIGURE 3. **The network architecture of MV-DNN. An illustration of auxiliary learning by slicing voxel feature in subvolume parts.**

The multiclass loss function can be written as:

$$l(y, \hat{y}) = -\sum_{j}^{N} y_j . \log \hat{y}_j, \tag{7}$$

where y and $\hat{y}$ are the ground truth and output scores of CNN for the j^{th} class in N classes. N is equal to 40 for the ModelNet40 and 10 for ModelNet10 databases. The parameter $\hat{y}$ is the probability function of predicting the target class by the softmax function and is written as:

$$\hat{y} = \frac{e^{x_i}}{\sum_{j}^{N} e^{x_j}}, \begin{cases} i = 1 \text{ to } M \\ j = 1 \text{ to } N, \end{cases} \tag{8}$$

where x_i is the logit score of input data and x_j is the score referred by the network for each class in N. x_i and x_j are used to predict the input class level by the softmax classifier probability function $\hat{y}(\in \ 0, 1)$. M is the number of total training samples. The final average loss function can be written from Eq. (7) as:

$$l(y, \hat{y}) = -\frac{1}{M}\left[\sum_{i=1}^{M}\sum_{j}^{N} 1\{y_i = j\} . \log\left(\frac{e^{x_i}}{\sum_{j}^{N} e^{x_j}}\right)\right]. \tag{9}$$

IV. CLASSIFICATION EXPERIMENTS

To implement the classification system, we used the Caffe framework [48] to train and test the model on a desktop PC with the following specifications: Intel Xeon-X5650, memory 24 GB, Windows 10 (64-bit), and Tesla K20C (5 GB).

A. DATA PREPARATION

To perform 3D object classification, we use the ModelNet40 [12] as the primary dataset (one of the largest 3D classification datasets) which contains 12,311 CAD models with 40 different classes. The dataset provides training and test sets, and contain 9,843 models and 2,468 models, respectively. We augment the dataset by rotating uniformly around the horizontal direction to produce multiorientations of each 3D model. We prepare several augmented datasets on depths 4th to 7th and multiorientation (3, 6, 9, 12, 18, 24 and 30 rotations) takes place only on depth 7th. This multiorientation of each 3D model is applied both the training and test datasets separately.

The second dataset is ModelNet10 [12], which consists of 4899 indoor CAD models of 10 different classes. The overall dataset for our experiments was prepared in the same way as ModelNet40 as discussed above.

TABLE 1. The list of parameters and architectural details of MV-DNN. FL and DR refer to filter size and dropout ratio respectively.

Layer Type	FL. / DR.	Stride	Features	Parame-ters
MOC-1: Conv.	3×3	1	32×16×8 ×1	435
BN, Scale, ReLU	---	--	(32×16×8×1)*3	
Conv.	3×3	1	32×32×8×1	4624
BN, Scale, ReLU	---	---	(32×32×8×1)*3	
PL	2×2	2	32×32×8× 1	
MOC-2: Conv.	3×3	1	32×64×8 ×1	18464
BN, Scale, ReLU	---	---	(32×64×8×1) *3	
Conv.	3 ×3	1	32×128×8×1	73792
BN, Scale, ReLU	----	---	(32×128×8×1) *3	
PL	2 ×2	2	32×128×8×1	
Conv.	3×3	1	32×256×8×1	295040
BN, Scale, ReLU	---	---	(32×256×8×1) *3	
PL	2×2	2	32×256×8 ×1	
Conv.	3×3	1	32×512×8×1	1179904
BN, Scale, ReLU	---	---	(32×512×8×1) *3	
PL	2*2	2	32×512×8×1	
O-V	---	---	32×512×2 ×2× 2	
Dropout	0.5	---	32×512×2×2×2	
TCN	1×1	1	32×512×2×2×2	262656
Flatten	---	---	32×512×2×2×2	
FC	----	----	32×4096	
BN, Scale,	---	---	32×512	
ReLU, Dropout	0.5	---	(32×512) *4	
FC	---	---	32×512	2097153
BN, Scale,	0.5	---	(32×512) *4	
ReLU, Dropout				2097153
FC	---	---	32×40	
Slice-1	---	---	(32×512×2×2 ×1) *2	20481
Slice-2	---	---	(32×512×2×1×1) *4	
Slice-3	---	---	(32×512×1×1×1) *8	
FC*	----	---	(32×40) *8	163848

FC* indicates 8-slicing classifiers for auxiliary learning. In the first MOC, conv-1 assumes the number of channels to be 3. MV-DNN contains approximately 6.21 M parameters.

In this experiment, the octree format is generated with a predefined number of rotations from 3D CAD (.off) models directly downloaded on the ModelNet dataset. Therefore, all augmented octree files with their corresponding labels (0 to 39) are put randomly into an LMDB file. These procedures are the same for making both the training and testing datasets. Figure 4 shows the comparison between the voxel and our efficient octree representation at various depths. Although we demonstrate depths 4th to 8th in Figure 4, our experiments are conducted up to depth 7th due to our GPU limitation. However, the computational cost of the octree network is $U(n^2)$ where full-voxel cost is $U(n^3)$, and here n is the voxel resolution. Figure 5 shows the multiorientation approach of a 3D human CAD model in octree on 12-view.

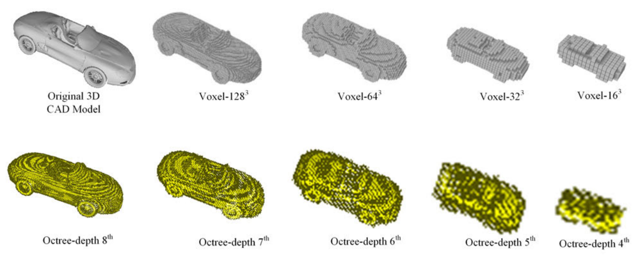

FIGURE 4. Comparison between voxel and our proposed octree representation of a 3D car model in various resolutions on ModelNet40.

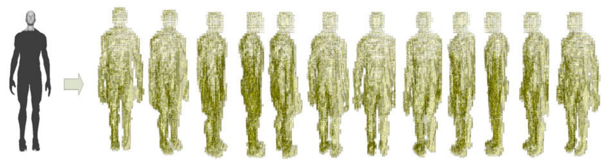

FIGURE 5. An example of multiorientation around the horizontal axis of a 3D person model in octree (d^{5th}) for 12-view on ModelNet40 dataset.

B. TRAINING DETAILS

All the training and testing experiments are conducted on the aforementioned desktop PC. Our MV-DNN is a feedforward supervised deep convolution network optimized by the SGD (stochastic gradient descent) method. We set a stepwise learning policy in which initial base learning is set to 0.1 and decreased by a factor of 10 every 10 epochs approximately. The classification and learning rates are plotted for different training sessions as follows. The network testing is carried out every 2000 iterations during the training. The batch size is set to 32 for all depths. We stop network optimization approximately 40 epochs in general, but we store the training model after passing every 5-epoch. During the training, we also observe the learning behavior and classification accuracy on the training dataset. We continuously store the accuracy and loss in the interval of 2000 iterations.

C. CLASSIFICATION RESULT ANALYSIS

This network employs the auxiliary learning tactics that continues to learn by the subvolume part. During weight updating, these subvolume classifiers modify the weights of the convolution filters that assist in predicting more accurate object classes. Initially, we prepare the octree data by uniformly rotating 12 poses of each 3D model for different octree depths. Therefore, the number of training samples on ModelNet 40 dataset is augmented from 9,843 to 118,116 samples, and the testing dataset is augmented from 2,468 to 2,9616 samples. All the samples are randomly placed into the LMDB dataset to make more realistic learning.

We prepared a couple of training and testing augmented datasets by varying input resolutions (depth) and orientations of each 3D model. We converted all 3D CAD models to different octree depths from the 4th to the 7th with 12 views. Then, the individual training was directed using different networks based on input depths. The impact of octree depths as evaluated to select the best input depth and multiple experiments

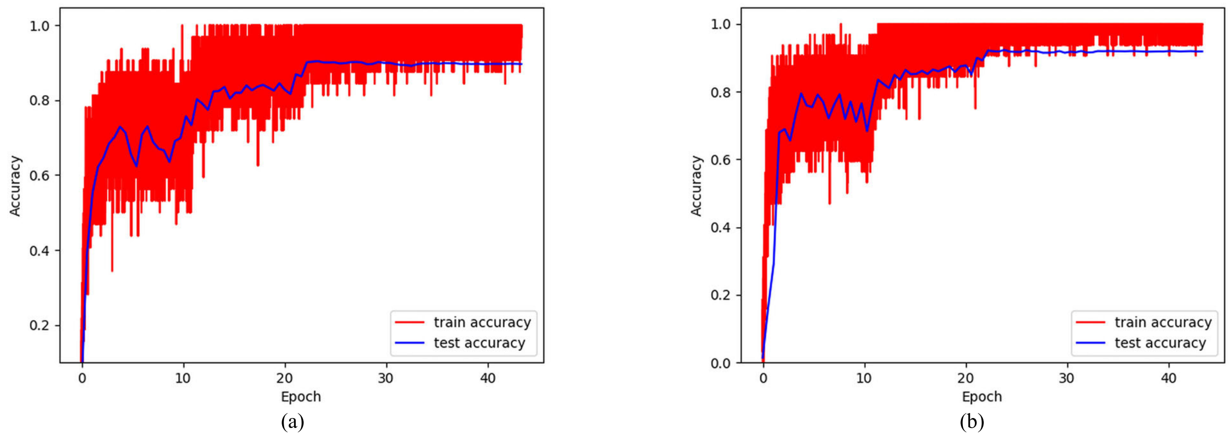
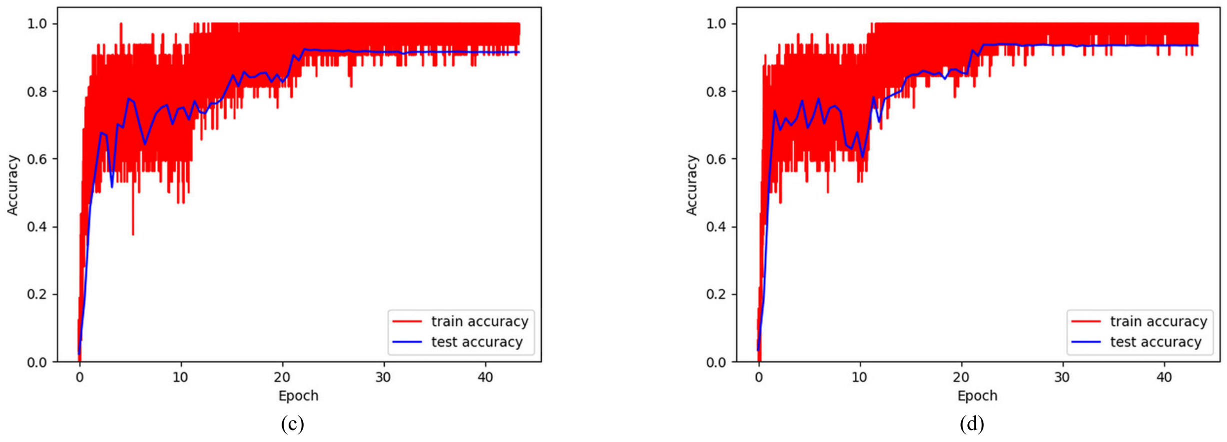

FIGURE 6. Classification accuracy during the training on 12 views, a) depth-4th, b) depth-5th, c) depth-6th and d) depth-7th.

were conducted to select the best orientation on perspective depth. We first selected the best depth on a 12-view dataset. Figure 6 depicts the training and the testing classification accuracy map during the training over the epochs. We can see that all the models reach an approximately stable point after 20 epochs. To validate the system, the trained model was tested on the testing dataset and these samples have never been used to train the network. We applied the voting strategy to use the average score from all orientations of a test sample and average instance accuracy is provided.

Figure 7 shows the comparison of classification accuracy in percentage among various input depths (resolution from 16^3 to 128^3) where the maximum accuracy achieved by our MV-DNN using depth 7th (12-view) is 92.1%.

Another question about multiorientation is how many orientations is better to make an augmented input training dataset, which is still an open issue for 3D object classification. To find the optimum input augmentation and maximize the classification accuracy, a number of training datasets are prepared using several views (3,6,12,18,24 and 30) of a 3D object. These views are made by rotating each 3D model uniformly outside the network during the data preparation stage.

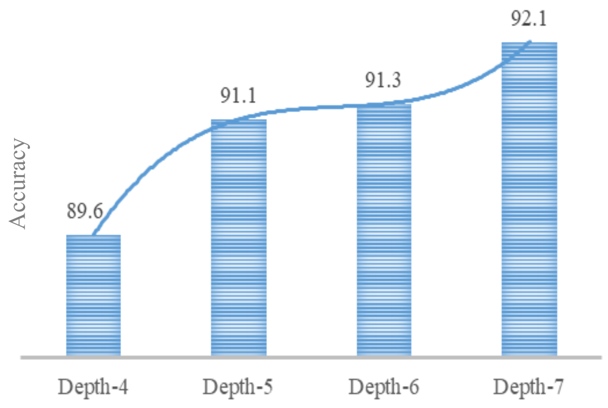

FIGURE 7. The best depth selection by MV-DNN on ModelNet40.

We took depth 7th (128^3) as a reference and observed the multioriented impacts over the epoch during the training. Figure 9 shows the classification result over these long-range augmented training datasets, and the best classification result is recorded on 24 views and is 92.6% in Figure 8. Our investigation reveals that the classification performance by our MV-DNN increases consecutively with the resolution and the

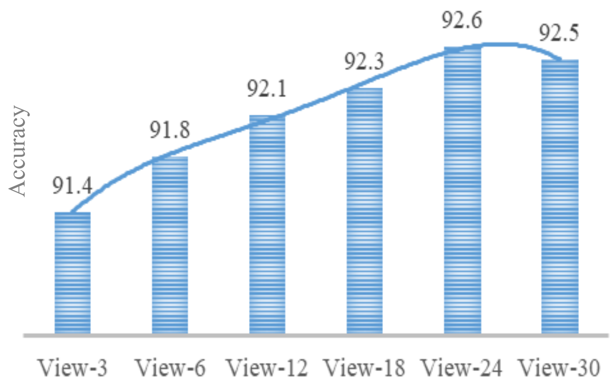

FIGURE 8. The best view selection on depth 7th by MV-DNN on different views on ModelNet40.

TABLE 2. A comparison between full voxel and octree representation with 32^3 volume resolution on ModelNet40 with effect of auxiliary learning.

Methods	Auxiliary Learning	Classification Accuracy (%)
MV-DNN (octree)	×	89.9
MV-DNN (full voxel)	×	89.2
MV-DNN (octree)	✓	91.1
MV-DNN (full voxel)	✓	90.3

augmented view. Average classification accuracy increases by 0.5% while doubling the training samples (12 to 24 views). It suggests that more samples in classes help to improve performance.

In contrast, we observed that the accuracy growth rate has slowed comparatively at depth-6 (64^3) and a small accuracy of 0.1% drops while using view-30 (augmented dataset) on depth 7th. It is a special case may happen only if the similarity (visual) occurs among some 3D objects in specific voxels representation [16] and also may happen because of using too many views of a sample. To tackle this situation, it requires a bigger dataset with invariant input samples and more advance networks need to be investigated.

However, we also compare the classification accuracy of our hybrid octree with full voxel representation [14] under the same network using a volume resolution of 32^3. Table 2 shows that our proposed octree with auxiliary learning gained classification accuracy by 0.8% than the full voxel method. The full voxel method assembles some redundant information as it stores both occupied and non-occupied spaces which increase the memory space. In contrast, octree stores occupied spaces only and contains more precise information than the full voxel method. According to experimental results, auxiliary learning noticeably improved classification accuracy by 1.2% which demonstrates the ability of our MV-DNN to extract more discriminative features from incomplete parts of a 3D CAD model. So, the improvement comes from auxiliary learning with octree representation.

In each training session, the loss function (softmax loss) was computed using Eq. (9) to measure the classification performance by considering the uncertainty of our network prediction. The accuracy (in percentage) is used in the applied perspective to measure the average classification, but it is not a very good estimation to judge a network because of its binary prediction behavior (yes or no). The higher the probability implies that the lower the loss is a symbol of a

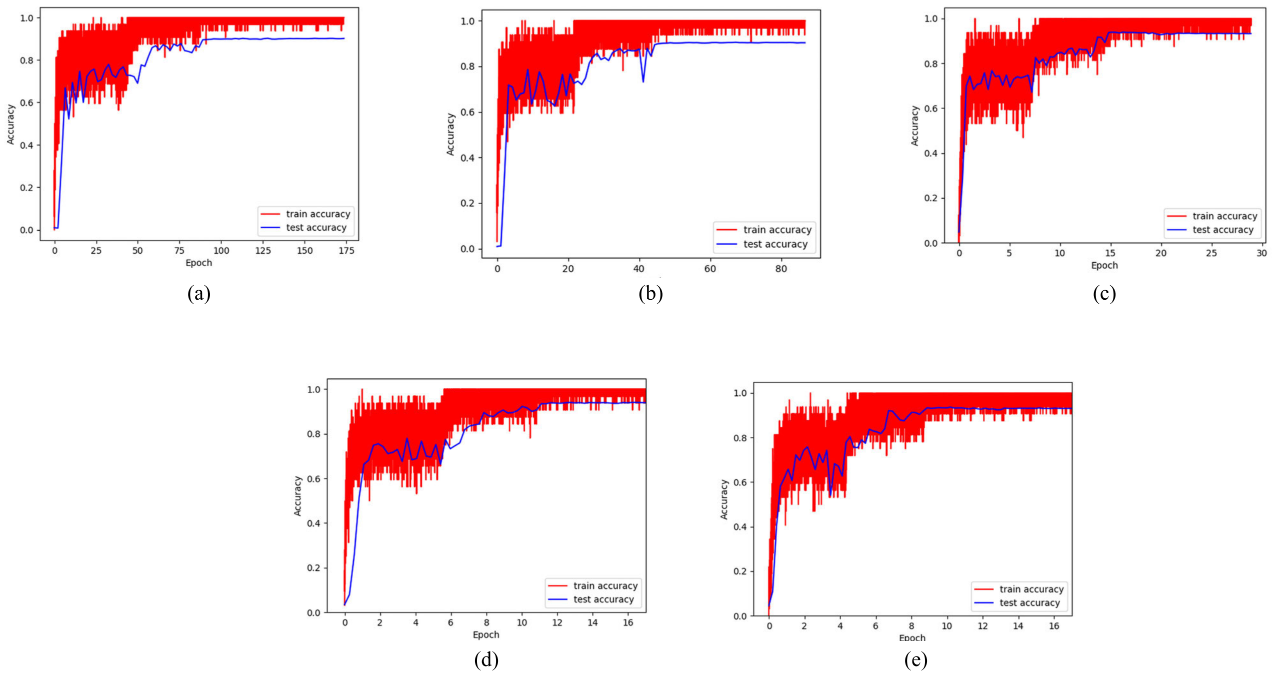

FIGURE 9. Accuracy plots of depth-7th with different views on ModelNet40 (a) view-3, (b) view-6, (c) view-18, (d) view-24 and (e) view-30.

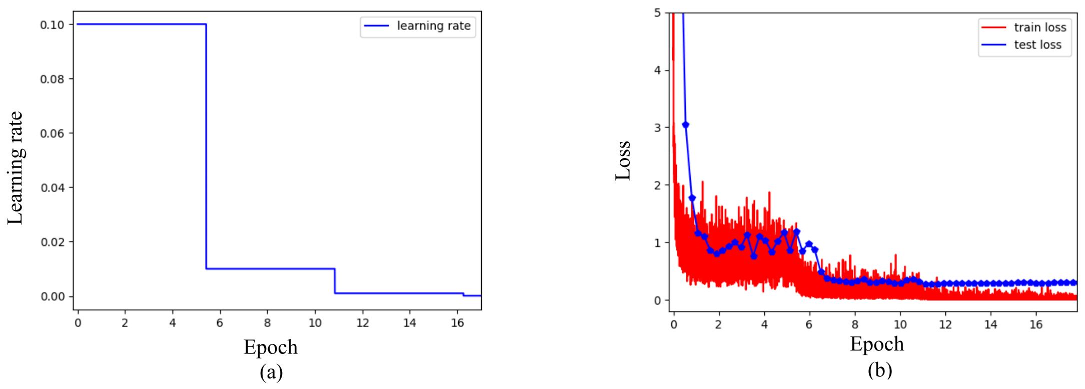

FIGURE 10. Depth-7th(24 views) on ModelNet40 (a) Epoch vs Learning rate (b) Epoch Vs Training loss.

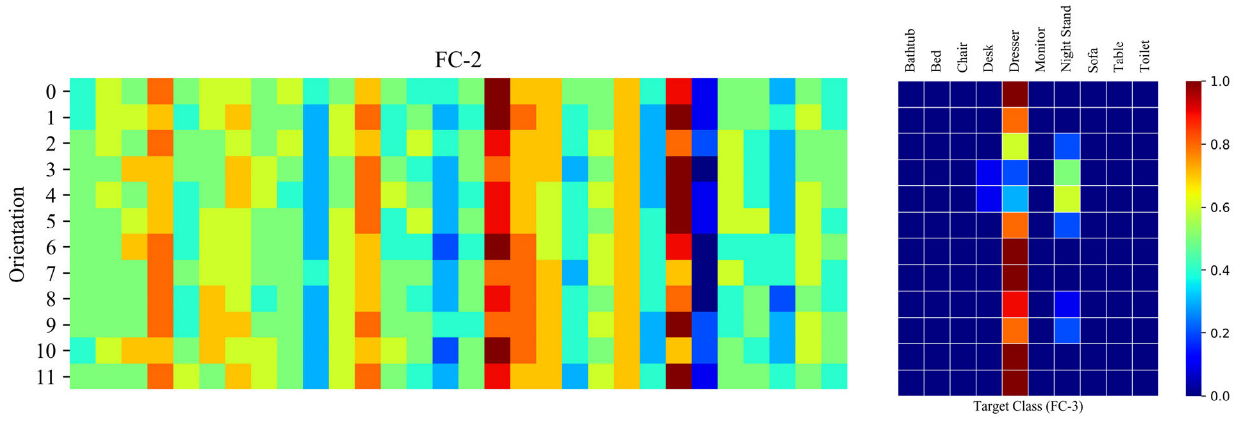

FIGURE 11. Visualization of learned features for dresser in multiple orientations. On the left, the second fully connected (FC-2) layer only 30 neurons are visualized. On the right, third fully connected (FC-3) layer outputs the corresponded object class on ModelNet10.

well-functioning network. Our target is to minimize the loss function for each training on the training dataset over the time which will have an impact to minimize the loss while testing using unseen objects. We can observe in Figure 10b, how the loss on the training dataset decreased over the epochs and influenced to decrease the loss on the test dataset ($\cong$0.32). However, to explain the classification accuracy over multioriented datasets, we can see in Figure 9a, MV-DNN keeps in learning after 40 epochs on view-3 dataset where it almost stops learning after 18 epochs and 12 epochs on the higher views 18 and 24 respectively (see in Figure 9c and 9d). This occurs because at the higher views, i.e., view-24, the number of augmented 3D models from an individual model help to converge the network earlier and gives comparatively better classification accuracy among others in terms of the required number of epochs. In view-3 and view-6, one epoch contains only 29,529 and 59,058 3D models for training and requires 922 and 1,845 iterations, respectively (batch size = 32), to pass all models. In contrast, view-12, view-24 and view-30 contain 118,112, 236,232 and 295,290 models that require 3691, 7382 and 9227 iterations, respectively, to make one epoch. Thus, the higher views contain more iterations in one epoch, assist in learning faster and achieving better classification accuracy earlier than lower views.

Therefore, if we increase the training session of view-3 and view-6, it also boosts classification accuracy, and we record the better results achieves by view-6 using approximately 80 epochs where we stop network optimization. The accuracy plot in Figure 9d shows that the test accuracy line using view-24 reaches an approximately stable position after 12 epochs of training, but we continue training the network further to achieve the best result. In addition, we also observe the learning rate and loss continuously during the training.Figure 10a depicts the learning rate over the epochs, and it seems the learning rate reached almost zero (0) after 16 epochs as did the loss function in Figure 10b. Therefore, we decided to stop training at approximately 16 epochs.

We tested the network using the last stored training model on the augmented test dataset containing 59,232 models, and the offline test accuracy achieved was 92.6%. By using the same setup of data augmentation and input resolution (Depth-7th and 24 views), our MV-DNN achieved classification accuracy of 95.1% on ModelNet10 dataset using a pretrained model on ModelNet40 dataset. We also used the original training and testing splits of Modelnet10 dataset. Figure 11 demonstrates an approximate rotational invariance learning by our MV-DNN across12 rotations of Dresser on ModelNet10. The fifth column of FC-3 is the correct response

TABLE 3. Classification accuracy on ModelNet datasets. M-40 and M-10 refer to ModelNet40 and ModelNet10 dataset respectively.

Framework Type	Networks	Input	Size	Views	M-40	M-10
Volumetric, Single	3DShapeNets [4]	Voxel grid, 24^3	38M	12	77.3	83.5
	VoxNet [14]	Voxel grid, 32^3	0.92M	12	83	92
	OctNet [16]	Octree, 128^3	-	--	86.5	90.9
	LightNet [7]	Voxel grid, 24^3	0.30M	12	88.93	93.61
	MO-AniProbing [31]	Input-3D $30^{3,}$ *Fn-2D Plane	-	-	89.9	-
	LP-3DCNN [33]	Voxels, 32^3	2M	24	92.1	94.4
	VRN [49]	Voxel, 32^3	18M	24	91.3	93.61
	Voxception [49]	Voxel, 32^3		24	90.56	93.28
	MV-DNN (Ours)	Octree, $128^{3,}$	6.22M	24	**92.6**	**95.1**
Volumetric, Ensemble	VRN Ensemble [49]	Voxel, 32^3	108M	24	95.54	97.14
	NormalNet [10]	Voxel30^3	6.5M	20	91.9	93.1
Hybrid, Ensemble	FusionNet [35]	Voxels,32^3, 2D image	118M	60	90.8	93.1
	Multibranch CNN [6]	Voxels, 32^3, 2D Image	-	6	89.3	93.6
Multiview	MVCNN [22]	2D Image	-	80	90.1	-
	RoataionNet [50]	2D Image	-	12	97.4	98.5
	3D2SeqViews [30]	2D Image	-	12	93.40	94.71
	LSTM [29]	2D Image	-	-	91.05	95.29
	PANO-ENN [51]	2D Image	-	-	95.6	96.9
Point Cloud	PointNet [37]	Points, 1k	3.5M	-	89.2	-
	ECC [52]	Points, 1k	-	12	87.4	90.8
	PointNet++ [38]	Points, 5k	1.48	-	91.9	-
	Kd-Net [39]	Points, 32k	-	-	91.6	93.3
	RS-CNN [41]	Points, 1k	1.41M	-	93.6	-
	3DCapsule [40]	Points, 1k	-	-	92.7	94.7

for Dresser. Moreover, a confusion takes place between second and fifth rotations with Night Stand but we obtain the target class by voting across all orientations. Table 3 shows the comparison of classification accuracy with state-of-art methods in percentage and all the classification results on ModelNet40 and ModelNet10 dataset are collected from respective papers.

Our proposed MV-DNN achieved better performance among volumetric single CNNs by using depth 7th on 24 views but performed inferior to VRN Ensemble [49]. Overall, multiview and point cloud CNNs are dominating the 3D object classification problem, except VRN Ensemble. MV-DNN obtains 92.6% accuracy on Model40 and 95.2% accuracy on ModelNet10 test dataset, which are respectively better than any other volumetric single networks and also comparable to point cloud and multiview CNNs. The acquired result by our MV-DNN indicates the design superiority with octree representation.

In addition, we also compared GPU memory consumption and the computational speed for one complete pass of our octree with full voxel method. To record the average time

TABLE 4. A comparison of GPU and time consumption between our sparse occupied octree and full voxel methods. The total GPU memory was 5 GB, and the batch size was 32.

Input Resolution	Representation	GPU Memory	Forward Pass	Backward Pass
32^3	**Octree**	1.032 GB	101.24 ms	189.63 ms
	Full voxel	4.22 GB	715 ms	1034.2 ms
64^3	**Octree**	1.80 GB	198 ms	293.193 ms
	Full voxel	Out of memory	-	-
128^3	**Octree**	3.925 GB	1063 ms	816 ms
	Full voxel	Out of memory	-	-

per iteration, we ran 500 forward and backward iterations on the GPU, and the total GPU consumption was also calculated. Both experiments were conducted using our MV-DNN framework. We found that our MV-DNN using the sparsely occupied (octree) method consumed less GPU memory under all resolutions presented in Table 4 compared to the full voxel method. The full voxel method cost approximately 4 times more memory. The average time for one iteration (batch = 32) by octree cost seven times less for a forward pass and approximately ten times less for a backward pass than the full voxel method.

V. CONCLUSIONS AND FUTURE WORKS

In this paper, we proposed a GPU-based volumetric deep convolution neural network for 3D object classification using multioriented 3D input data. We addressed two general problems of using DNN to 3D object classification that require a large-scale training dataset and high computational memory. We introduced a low cost sparsely occupied volumetric representation to the newly proposed auxiliary learning DNN framework by employing a layer slicing methodology to learn from a subdivision part of a 3D object. We used standard training and testing splits as provided by the ModelNet dataset. The experimental results show that our DNN architecture performed well with comparatively small training effort and classification accuracy is improved than state-of-art volumetric methods. Due to use of octree generating low-cost volumetric features, our proposed network requires less computational memory and time compared to full voxel methods. In addition, to validate our network architecture, a classification loss map was also drawn during the training session. However, our network takes very cleaned 3D data as the input. In real world scenarios, there are multiple objects with overlap positions. In that case, our network has a lack of capability for taking those inputs to classify.

In our future work, this new approach of MV-DNN (octree-based auxiliary learning) can be investigated further to solve more challenging tasks where high resolutions are required including real time 3D scene analysis and 3D appearance or motion patterns recognition.

REFERENCES

[1] Z. Bi and L. Wang, "Advances in 3D data acquisition and processing for industrial applications," *Robot. Comput.-Integr. Manuf.*, vol. 26, no. 5, pp. 403–413, Oct. 2010, doi: 10.1016/j.rcim.2010.03.003.

[2] J. Zeng, M. Liu, X. Fu, R. Gu, and L. Leng, "Curvature bag of words model for shape recognition," *IEEE Access*, vol. 7, pp. 57163–57171, 2019, doi: 10.1109/access.2019.2913688.

[3] A. X. Chang, "ShapeNet: An information-rich 3D model repository," *CoRR*, vol. abs/1512.03012, pp. 1–11, Dec. 2015.

[4] Z. Wu, "3D ShapeNets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Boston, MA, USA, Jun. 2015, pp. 1912–1920, doi: 10.1109/CVPR.2015.7298801.

[5] F. Gomez-Donoso, A. Garcia-Garcia, J. Garcia-Rodriguez, S. Orts-Escolano, and M. Cazorla, "LonchaNet: A sliced-based CNN architecture for real-time 3D object recognition," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 412–418, doi: 10.1109/ijcnn.2017.7965883.

[6] L. Minto, P. Zanuttigh, and G. Pagnutti, "Deep learning for 3D shape classification based on volumetric density and surface approximation clues," in *Proc. 13th Int. Joint Conf. Comput. Vis., Imag. Comput. Graph. Theory Appl.*, Funchal, Portugal, 2018, pp. 317–324, doi: 10.5220/0006619103170324.

[7] S. Zhi, Y. Liu, X. Li, and Y. Guo, "Toward real-time 3D object recognition: A lightweight volumetric CNN framework using multitask learning," *Comput. Graph.*, vol. 71, pp. 199–207, Apr. 2018, doi: 10.1016/j.cag.2017.10.007.

[8] H. You, Y. Feng, R. Ji, and Y. Gao, "PVNet: A joint convolutional network of point cloud and multi-view for 3D shape recognition," Aug. 2018, *arXiv:1808.07659*. [Online]. Available: https://arxiv.org/abs/1808.07659

[9] E. Ahmed, A. Saint, A. E. R. Shabayek, K. Cherenkova, and D. Aouada, "Deep learning advances on different 3D data representations: A survey," 2019, *arXiv:1808.01462v2*. [Online]. Available: https://arxiv.org/abs/1808.01462v2

[10] C. Wang, M. Cheng, F. Sohel, M. Bennamoun, and J. Li, "NormalNet: A voxel-based CNN for 3D object classification and retrieval," *Neurocomputing*, vol. 323, pp. 139–147, Jan. 2019, doi: 10.1016/j.neucom.2018.09.075.

[11] M. Rezaei, M. Rezaeian, V. Derhami, F. Sohel, and M. Bennamoun, "Deep learning-based 3D local feature descriptor from Mercator projections," *Comput. Aided Geometric Des.*, vol. 74, Oct. 2019, Art. no. 101771, doi: 10.1016/j.cagd.2019.101771.

[12] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006, doi: 10.1162/neco.2006.18.7.1527.

[13] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proc. 26th Annu. Int. Conf. Mach. Learn. (ICML)*, Montreal, QC, Canada, 2009, pp. 1–8, doi: 10.1145/1553374.1553453.

[14] D. Maturana and S. Scherer, "VoxNet: A 3D convolutional neural network for real-time object recognition," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Hamburg, Germany, Sep. 2015, pp. 922–928, doi: 10.1109/IROS.2015.7353481.

[15] A. Caglayan and A. B. Can, "Volumetric object recognition using 3-D CNNs on depth data," *IEEE Access*, vol. 6, pp. 20058–20066, 2018, doi: 10.1109/access.2018.2820840.

[16] G. Riegler, A. O. Ulusoy, and A. Geiger, "OctNet: Learning deep 3D representations at high resolutions," Nov. 2016, *arXiv:1611.05009*. https://arxiv.org/abs/1611.05009

[17] D. Meagher, "Geometric modeling using octree encoding," *Comput. Graph. Image Process.*, vol. 12, no. 2, pp. 129–147, 1982.

[18] R. Wm Franklin and V. Akman, "Octree data structures and creation by stacking," in *Computer-Generated Images*, N. Magnenat-Thalmann and D. Thalmann, Eds. Tokyo, Japan: Springer, 1985, pp. 176–185.

[19] S. Laine and T. Karras, "Efficient sparse voxel octrees," *IEEE Trans. Visual. Comput. Graph.*, vol. 17, no. 8, pp. 1048–1059, Aug. 2011, doi: 10.1109/tvcg.2010.240.

[20] M. Tatarchenko, A. Dosovitskiy, and T. Brox, "Octree generating networks: Efficient convolutional architectures for high-resolution 3D outputs," Mar. 2017, *arXiv:1703.09438*. [Online]. Available: https://arxiv.org/abs/1703.09438

[21] W. Kehl, T. Holl, F. Tombari, S. Ilic, and N. Navab, "An octree-based approach towards efficient variational range data fusion," Aug. 2016, *arXiv:1608.07411*. [Online]. Available: https://arxiv.org/abs/1608.07411

[22] 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. Vis. (ICCV)*, Dec. 2015, pp. 945–953, doi: 10.1109/iccv.2015.114.

[23] J.-C. Su, M. Gadelha, R. Wang, and S. Maji, "A deeper look at 3D Shape Classifiers," in *Proc. 2nd Workshop 3D Reconstruction Meets Semantics (ECCV)*, 2018, p. 20.

[24] S. Har-Peled, "Quadtrees-hierarchical grids," in *Geometric Approximation Algorithms*, vol. 173. Providence, RI, USA: American Mathematical Society, 2011, pp. 19–30.

[25] Z. Xie, K. Xu, W. Shan, L. Liu, Y. Xiong, and H. Huang, "Projective feature learning for 3D shapes with multi-view depth images," *Comput. Graph. Forum*, vol. 34, no. 7, pp. 1–11, Oct. 2015, doi: 10.1111/cgf.12740.

[26] S. Bu, P. Han, Z. Liu, J. Han, and H. Lin, "Local deep feature learning framework for 3D shape," *Comput. Graph.*, vol. 46, pp. 117–129, Feb. 2015, doi: 10.1016/j.cag.2014.09.007.

[27] K. Sfikas, T. Theoharis, and I. Pratikakis, "Exploiting the PANORAMA representation for convolutional neural network classification and retrieval," in *Proc. Eurograph. Workshop 3D Object Retr.*, 2017, p. 7, doi: 10.2312/3DOR.20171045.

[28] Y. Zhang, Z. Liu, T. Liu, B. Peng, and X. Li, "RealPoint3D: An efficient generation network for 3D object reconstruction from a single image," *IEEE Access*, vol. 7, pp. 57539–57549, 2019, doi: 10.1109/access.2019.2914150.

[29] C. Ma, Y. Guo, J. Yang, and W. An, ''Learning multi-view representation with LSTM for 3-D shape recognition and retrieval,'' *IEEE Trans. Multimedia*, vol. 21, no. 5, pp. 1169–1182, May 2019, doi: 10.1109/tmm.2018.2875512.

[30] Z. Han, H. Lu, Z. Liu, C.-M. Vong, Y.-S. Liu, M. Zwicker, J. Han, and C. L. P. Chen, ''3D2SeqViews: Aggregating sequential views for 3D global feature learning by CNN with hierarchical attention aggregation,'' *IEEE Trans. Image Process.*, vol. 28, no. 8, pp. 3986–3999, Aug. 2019, doi: 10.1109/tip.2019.2904460.

[31] C. R. Qi, H. Su, M. NieBner, A. Dai, M. Yan, and L. J. Guibas, ''Volumetric and multi-view CNNs for object classification on 3D data,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 5648–5656, doi: 10.1109/CVPR.2016.609.

[32] M. Lin, Q. Chen, and S. Yan, ''Network in network,'' Dec. 2013, *arXiv:1312.4400*. https://arxiv.org/abs/1312.4400

[33] S. Kumawat and S. Raman, ''LP-3DCNN: Unveiling local phase in 3D convolutional neural networks,'' Apr. 2019, *arXiv:1904.03498*. https://arxiv.org/abs/1904.03498

[34] S. H. Khan, Y. Guo, M. Hayat, and N. Barnes, ''Unsupervised primitive discovery for improved 3D generative modeling,'' Jun. 2019, *arXiv:1906.03650*. [Online]. Available: https://arxiv.org/abs/1906.03650

[35] V. Hegde and R. Zadeh, ''FusionNet: 3D Object classification using multiple data representations,'' Jul. 2016, *arXiv:1607.05695*. [Online]. Available: https://arxiv.org/abs/1607.05695

[36] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ''ImageNet classification with deep convolutional neural networks,'' in *Adv. Neural Inf. Process. Syst.*, vol. 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Red Hook, NY, USA: Curran Associates, 2012, pp. 1097–1105.

[37] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, ''PointNet: Deep learning on point sets for 3D classification and segmentation,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 77–85, doi: 10.1109/CVPR.2017.16.

[38] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, ''PointNet++: Deep hierarchical feature learning on point sets in a metric space,'' Jun. 2017, *arXiv:1706.02413*. [Online]. Available: https://arxiv.org/abs/1706.02413

[39] 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. Vis. (ICCV)*, Venice, Italy, Oct. 2017, pp. 863–872, doi: 10.1109/ICCV.2017.99.

[40] A. Cheraghian and L. Petersson, ''3DCapsule: Extending the capsule architecture to classify 3D point clouds,'' in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2019, pp. 1194–1202, doi: 10.1109/wacv.2019.00132.

[41] Y. Liu, B. Fan, S. Xiang, and C. Pan, ''Relation-shape convolutional neural network for point cloud analysis,'' May 2019, *arXiv:1904.07601*. [Online]. Available: https://arxiv.org/abs/1904.07601

[42] Y. Yang, C. Feng, Y. Shen, and D. Tian, ''FoldingNet: Point cloud auto-encoder via deep grid deformation,'' in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 206–215, doi: 10.1109/cvpr.2018.00029.

[43] J. Li, B. M. Chen, and G. H. Lee, ''SO-Net: Self-organizing network for point cloud analysis,'' in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 9397–9406, doi: 10.1109/CVPR.2018.00979.

[44] F. Calakli and G. Taubin, ''SSD: Smooth signed distance surface reconstruction,'' *Comput. Graph. Forum*, vol. 30, no. 7, pp. 1993–2002, Sep. 2011, doi: 10.1111/j.1467-8659.2011.02058.x.

[45] S. Fuhrmann and M. Goesele, ''Fusion of depth maps with multiple scales,'' in *Proc. SIGGRAPH Asia Conf.*, New York, NY, USA, 2011, pp. 148:1-148:8, doi: 10.1145/2024156.2024182.

[46] B. Ummenhofer and T. Brox, ''Global, dense multiscale reconstruction for a billion points,'' *Int. J. Comput. Vis.*, vol. 125, nos. 1–3, pp. 82–94, Dec. 2017, doi: 10.1007/s11263-017-1017-7.

[47] F. Steinbrücker, J. Sturm, and D. Cremers, ''Volumetric 3D mapping in real-time on a CPU,'' in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May/Jun. 2014, pp. 2021–2028, doi: 10.1109/ICRA.2014.6907127.

[48] Y. Jia, ''Caffe: Convolutional architecture for fast feature embedding,'' in *Proc. ACM Multimedia (ACMMM)*, 2014, pp. 675–678.

[49] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, ''Generative and discriminative Voxel modeling with convolutional neural networks,'' Aug. 2016, *arXiv:1608.04236*. [Online]. Available: https://arxiv.org/abs/1608.04236

[50] A. Kanezaki, Y. Matsushita, and Y. Nishida, ''RotationNet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints,'' Mar. 2016, *arXiv:1603.06208*. [Online]. Available: https://arxiv.org/abs/1603.06208

[51] K. Sfikas, I. Pratikakis, and T. Theoharis, ''Ensemble of PANORAMA-based convolutional neural networks for 3D model classification and retrieval,'' *Comput. Graph.*, vol. 71, pp. 208–218, Apr. 2018, doi: 10.1016/j.cag.2017.12.001.

[52] M. Simonovsky and N. Komodakis, ''Dynamic edge-conditioned filters in convolutional neural networks on graphs,'' Apr. 2017, *arXiv:1704.02901*. [Online]. Available: https://arxiv.org/abs/1704.02901

A. A. M. MUZAHID received the M.E. degree in communication and information engineering from the Chongqing University of Posts and Telecommunications, Chongqing, China, in 2016, and the B.Sc. degree in electronics and telecommunications engineering from Daffodil International University, Dhaka, Bangladesh, in 2011. He is currently pursuing the Ph.D. degree in communication and information systems with Shanghai University, Shanghai, China. He is a Research Member of the Institute of Smart City, Shanghai University, and leading the ''Computer Vision and 3D Virtual Reality'' research team. His current research interests include 3D Shape analysis, computer vision, and 3D virtual reality.

WANGGEN WAN (Senior Member, IEEE) received the Ph.D. degree from Xidian University, China, in 1992. He was a Postdoctoral Research Fellow of the Information and Control Engineering Department, Xi'an Jiao-Tong University, from 1993 to 1995. From 1995 to 1998, he was working at the Electronic and Information Engineering Department, Shanghai University (SHU), China. He was a Visiting Scholar with the Electrical and Electronic Engineering Department, The Hong Kong University of Science and Technology (HKUST), in 1998 and 1999, respectively. He was a Visiting Professor and the Section Head of the Multimedia Innovation Center, HKPU, from 2000 to 2004. He is currently a Professor with the School of Communication and Information Engineering, SHU. His research interests include computer graphics, computer vision, signal processing, and data mining. He became an IET Fellow, in 2006.

FERDOUS SOHEL (Senior Member, IEEE) received the Ph.D. degree from Monash University, Australia, in 2009. He is currently an Associate Professor in information technology with Murdoch University, Australia. Prior to his joining Murdoch University, he was a Research Assistant Professor/Research Fellow of the University of Western Australia, from 2008 to 2015. His research interests include computer vision, image processing, machine learning, pattern recognition, multimodal biometrics, digital agritech, cyber forensics, and video coding. He has been serving as an Associate Editor of the IEEE Transactions on Multimedia. He is a member of the Australian Computer Society.

NAIMAT ULLAH KHAN was born in Bannu, Khyber Pakhtunkhwa, Pakistan, in 1985. He received the B.S.C.S. from the Kohat University of Science and Technology, and the M.S. degree in computer science from Preston University Islamabad, in 2014. He is currently pursuing the Ph.D. degree in communication and information systems with Shanghai University, China. He worked as a Lecturer with Preston University, Islamabad, from 2015 to 2016. He has been working in the field of artificial intelligence, machine learning, and big data at the School of Smart City, Shanghai University, and the University of Technology Sydney, under Dual-Doctoral program, mainly focusing on big data analysis through machine learning and AI.

OFELIA DELFINA CERVANTES VILLAGÓMEZ received the M.Sc. degree in computer systems from the École Supérieure en Informatique, Grenoble, France, and the Ph.D. degree from the Institut National Polytechnique de Grenoble, France. She is currently a Professor of computer science and the Director of the Innovation for Development (INNOVA4D) Research Group, Universidad de las Américas Puebla (UDLAP), Mexico. She is a co-founding member of the Laboratorio Nacional de Informática Avanzada, A.C. (National Laboratory for Advanced Informatics, A.C.) established in Jalapa. She has served in several CONACYT committees to promote cooperation among Information Technologies research groups from Mexico, France, USA, and Germany. She has served as the President of the Mexican Artificial Intelligence Society, and the President of the Mexican Association for International Education. Her research interests include the areas of data science, natural language processing, and human–computer interaction.

HIDAYAT ULLAH received the master's degree from Preston University, Islamabad, Pakistan, in 2015. He is currently pursuing the Ph.D. degree with the School of Communication and Information Engineering, Shanghai University, Shanghai, China. He is also a member of the Institute of Smart City, Shanghai University. His research interests mainly focused on big data, social media, environmental science, smart city, and urban planning. He has been involved in several research projects, including the National Natural Science Foundation of China and Shanghai Science and Technology. He has been awarded the CSC Scholarship (China Scholarship Council) to study at Shanghai University.