IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 25, 2020, accepted August 7, 2020, date of publication September 11, 2020, date of current version October 29, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3023423

Faster Dynamic Graph CNN: Faster Deep Learning on 3D Point Cloud Data

JINSEOK HONG[1,3], KEEYOUNG KIM[1,2], AND HONGCHUL LEE[3]

[1]Artificial Intelligence Research Institute, Seongnam 13120, South Korea
[2]Ingenio AI, Seoul 02841, South Korea
[3]School of Industrial Management Engineering, Korea University, Seoul 02841, South Korea

Corresponding author: Hongchul Lee (hclee@korea.ac.kr)

This work was supported in part by the Ministry of Culture, Sports and Tourism (MCST), and in part by the Korea Creative Content Agency (KOCCA), Culture Technology (CT) Research and Development Program, in 2020.

ABSTRACT Geometric data are commonly expressed using point clouds, with most 3D data collection devices outputting data in this form. Research on processing point cloud data for deep learning is ongoing. However, it has been difficult to apply such data as input to a convolutional neural network (CNN) or recurrent neural network (RNN) because of their unstructured and unordered features. In this study, this problem was resolved by arranging point cloud data in a canonical space through a graph CNN. The proposed graph CNN works dynamically at each layer of the network and learns the global geometric features by capturing the neighbor information of the points. In addition, by using a squeeze-and-excitation module that recalibrates the information for each layer, we achieved a good trade-off between the performance and the computation cost, and a residual-type skip connection network was designed to train the deep models efficiently. Using the proposed model, we achieved a state-of-the-art performance in terms of classification and segmentation on benchmark datasets, namely ModelNet40 and ShapeNet, while being able to train our model 2 to 2.5 times faster than other similar models.

INDEX TERMS Classification, deep learning, graph CNN, point cloud, segmentation.

I. INTRODUCTION

The point cloud is the simplest form in which data can be expressed. Advances in technologies, such as Light Detection and Ranging (LIDAR) and three-dimensional (3D) scanning, have enabled acquiring 3D point cloud forms quickly. Accordingly, a vision and graphic process that can directly processes point clouds without mesh reconstruction or denoising, which is an essential preprocessing step for point cloud data, has recently emerged in applications such as automatic indoor navigation [1], self-driving vehicles [2]–[4] and robotics [5]–[7]. This process uses an algorithm that identifies semantic information and image features, instead of identifying geometric features such as nodes and edges. A learning-based approach, rather than an existing computational framework or a geometric approach, is required to use these features.

In this paper, we introduce an algorithm that ensures the speed and accuracy of point cloud classification and segmentation. Conventional point cloud processing methods use handcrafted features to extract the geometric features of the point cloud. With advancements in deep learning architectures for 2D image processes, various learning-based algorithms for processing 3D point clouds, with the first being proposed in [8], have emerged. These methods outperform the conventional ones.

The learning-based point cloud processing algorithms are more complex than 2D images. The neural network model for a 2D image uses a grid input, whereas 3D point cloud data have a fundamentally irregular shape. The position of the points is distributed continuously in the 3D space, and the typical ordering permutation does not change the spatial distribution. Accordingly, a method for applying the point cloud to deep learning models by converting the point cloud into a 3D grid format has been introduced; however, this approach requires excessive memory, and it is difficult to obtain high-resolution features.

PointNet [9] exhibits permutation invariance by accumulating features using symmetric functions independently at each point. This research enabled inputting point cloud data to

The associate editor coordinating the review of this manuscript and approving it for publication was Mingbo Zhao.

a deep neural network as raw data without any preprocessing. Starting with PointNet, various point cloud processing deep neural networks (DNNs) have been studied by considering point-oriented neighbors and further developing the learning of local features ([10], [11]). The authors in [10] considered the regional information of input data by sampling and grouping and then applying PointNet. However, the geometric information between a point and an adjacent point could not be considered, and there were limitations in extracting the local features of the point cloud.

A dynamic graph convolutional neural network (DGCNN) ([12]) addresses this problem by introducing the concept of EdgeConv, which enables the acquisition of geometric features of the point cloud while maintaining permutation invariance. The edge feature between the points and the adjacent points was utilized instead of directly embedding the points, as in the conventional method. Edge information is also permutative because it can be imported regardless of the order of the neighboring points. EdgeConv can group points in both the Euclidean space and the semantic space because it builds a local graph and learns by embedding the edge.

Amid the development of models that can manage 3D point clouds, research on creating a more efficient model for a 2D image-processing natural network is ongoing. Recent studies have found that the performance of neural networks can be improved by built-in embedding learning algorithms that capture spatial cores without additional control. For example, the inception architecture [13], [14] can modularize multi-scale processes and incorporate them into the network to enhance the performance of the model. The authors in [15], [16] suggested a method to consider the spatial dependency more comprehensively, while the authors in [17] built an efficient model by considering the spatial attention. The ''squeeze-and-excitation'' (SE) network [18] is a fast and high-performing module realized through feature recalibration.

This paper presents a model that can manage 3D point cloud data more quickly and accurately. The proposed model considers the local features between 3D points, maintains its performance, and learns much faster than conventional models through the recalibration process. We conducted classification and segmentation experiments on the ModelNet40 [19] and ShapeNet [20] datasets. Our model has a learning speed twice that of the existing model and demonstrates a state-of-the-art performance.

The key contributions of this study are as follows:

- The expression power of the edge feature and point feature map is improved using the recalibration block on the edge convolution block.
- Using a skip-dense network, we learned a model with more number of layers faster.
- We conducted experiments with the proposed model and achieved a state-of-the-art performance on benchmark datasets with a learning speed 2 to 2.5 times faster than those of other similar models.

II. RELATED WORKS

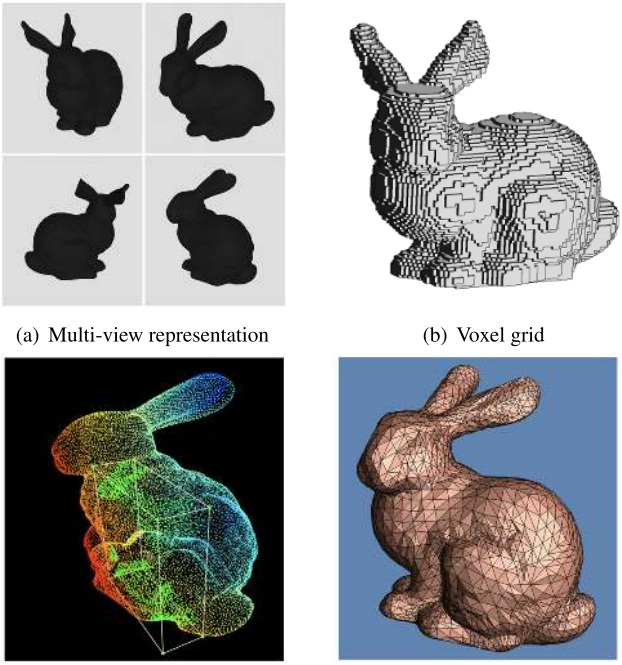
A. DEEP NEURAL NETWORK ARCHITECTURES

VGGNet [21] and the inception model [14] can provide improved depth in a neural network model. Batch normalization [13] can be applied to stabilize the learning process by inserting a module to adjust the layer input. ResNet [22], [23] can learn high-depth models effectively using a skip connection, and a highway network [24] can adjust short connections using a gating mechanism. The authors in [25], [26] further improved the learning features by reforming the connections between the networks.

Other researchers studied how to adjust the functions of the modules entering the network. References [27], [28] proposed grouped convolution to improve performance by increasing the cardinality of the transformation. References [14], [29], [30] proposed a generalized grouped convolution concept—multi-branch convergence—that enabled a more flexible operator configuration. References [31], [32] proposed an automated method of learning and exhibited competitive performance. References [33], [34] exhibited cross-channel correlation mapped to a new combination of features regardless of spatial structure, while [35] proposed cross-channel correlation using standard convolution filters with 1×1 convolutions. These two forms of research have been focused on the goal of reducing model and computational complexity, reflecting the assumption that channel relationships can be formalized with the configuration of instance agnostic functions and local receptive fields.

Other studies have been conducted on increasing the performance of models using ''attention,'' which can be described as a vector of the importance of weights. Attention identifies how strongly the input elements are associated with other elements and represents them by their sum to predict or estimate any input elements, such as pixel values in images or words in sentences. This algorithm has been applied in several areas, ranging from localization and image interpretation [17], [36] to sequence-based models [37], [38].

Attention is typically implemented with gating functions, such as softmax or sigmoid, or with sequential techniques [39]. Recent studies have found that it can be applied to tasks such as image captions [28], [40]. The authors in [41] introduced a powerful trunk-and-mask attachment mechanism using the hourglass module [16], and a high-capacity unit that was inserted into the deep residual networks between the intermediate stages, demonstrating excellent performance. The SE network (SE-net) [18] focuses on channel relationships, using modules that recalibrate the features of the channel, achieving state-of-the-art results in ImageNet recognition. In this paper, we propose a DNN with 3D point cloud data as the input. In contrast to the existing 3D deep learning models, our model achieves a state-of-the-art performance using a skip-connection network and attention-recalibration blocks; moreover, it learns 2-2.5 times faster than the other similar-level models.



(a) Multi-view representation (b) Voxel grid

(c) Point cloud (d) Triangular mesh

FIGURE 1. 3D Data Representation [28].

B. 3D DATA REPRESENTATION

Three-dimensional data can be represented using four methods, as depicted in Fig. 1: (a) multi-view, (b) voxel grid, (c) point cloud, or (4) triangle mesh.

The multi-view-based method (Fig. 1(a)) represents a 3D object as a set of images from various viewpoints. The multi-view method can reduce the dimensions by expressing 3D objects as a set of 2D images. However, it will not show 3D characteristics and requires multiple datasets for one object. A voxel-based method is a form of expression that converts an object into a voxel grid, as depicted in Fig. 1(b). Voxels are data representations suitable for naturally extending the processing methods in 2D to 3D.

However, because voxels have many sparse parts and are expressed in a grid unit form, it cannot effectively capture the details of 3D objects and incurs a high computational cost. As depicted in Fig. 1(c), the point cloud-based method is expressed in a 3D coordinate set. Point clouds are widely used for 3D objects and scenes, and many 3D point cloud datasets can be obtained today using 3D scanners, depth cameras, and LIDAR devices. A triangular mesh (Fig. 1(d)) is expressed as a collection of triangular faces approximating a geometric surface and can be viewed as a collection of 3D points sampled from a continuous geometric surface. The mesh-based method aims to represent a surface in a way that it can be easily rendered. The triangular mesh was initially created for computer graphics but is also useful for 3D vision.

This study incorporates 3D point cloud data because the point cloud object model is more realistic than multi-view, voxel, and mesh-based methods, and the associated input data processing has a lower computational cost. Classification and segmentation experiments were conducted using ModelNet40 and ShapeNet, which are 3D point cloud benchmark datasets.

C. GEOMETRIC DEEP LEARNING

Since the breakthrough of the convergence neural network in the 2D image field, as described in Section II-A, there have been many attempts to apply these methods to the geometry domain. However, in contrast to 2D images, geometric data over three dimensions typically lack a basic grid. Therefore, a network block applicable to convolution and pooling or grid structures is required.

A multi-view-based learning algorithm [43], [44] or voxel-based learning algorithms [19], [45]–[47] and a method of combining them [48] have been proposed to solve this problem. Voxel-based deep learning models require significant memory. More recently, in PointNet [9], [10] a CNN model was used with point cloud input, bringing significant research attention to geometric deepening-related algorithms [49] that use non-Euclidean data, such as graphics and manifolds. The authors in [10] considered regional information of input data by sampling and grouping them and then applying PointNet. Therefore, it could not consider the geometric information between a point and an adjacent point and had limitations in extracting local features of the point cloud. The authors in [50] suggested applying neural networks to graphs, and the authors in [51] developed this approach to apply a gated recurrent unit to the graphs. The authors in [52], [53] graphically generalized convolution using Laplacian eigenvectors. These methods have computational shortcomings that have been addressed using polynomial filters [54]–[56] and spectral filters [57], [58] to avoid Laplacian eigendecomposition and ensure localization.

A geodesic CNN (GCNN) [59] is a non-Euclidean deep learning algorithm that uses a spectral filter instead of a spatial filter. The GCNN is a CNN model applied to meshes that generalize the concept of the patch using local internal parameterization. The advantage of this spectral approach is that it is easy to configure for directional filters and offers superior generalization. The authors in [60] used anisotropic diffusion, and the authors in [61], [62] proposed a new local charting technique in a GCNN using a Gaussian mixture model. The authors [63], [64] performed an essential structural prediction of the correspondence between nonrigid shapes by integrating a differentiated functional map layer into a geometric deep learning model.

As described previously, studies on applying deep learning models to 3D objects have continued. In this study, a deep learning model was constructed using point cloud data as the input. We propose a model with minimal computational cost and state-of-the-art accuracy based on an efficient deep learning model that considers regional features between points.

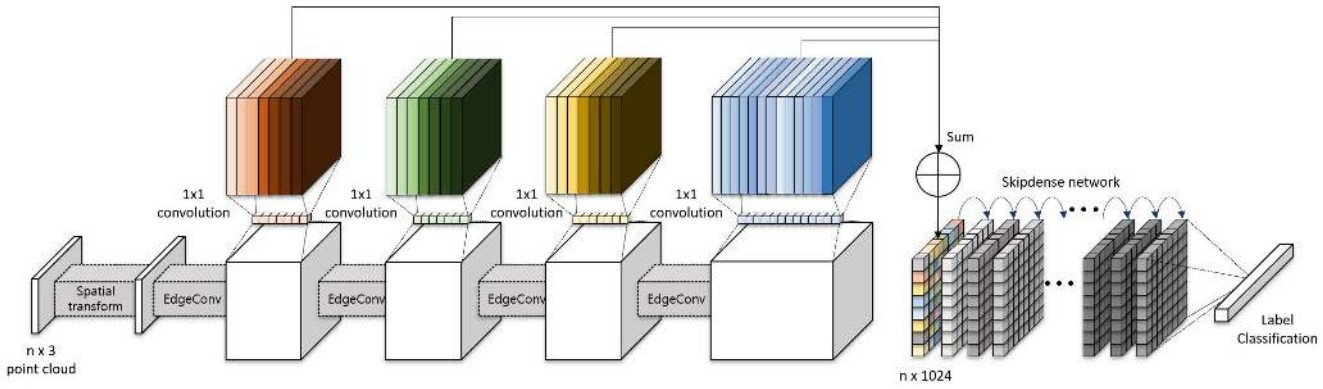
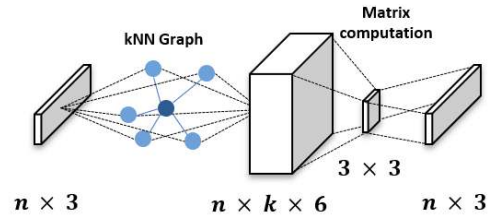
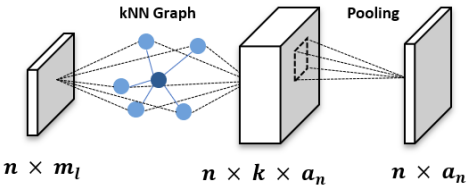


FIGURE 2. Our proposed classification model: The classification model receives n points as input and computes an edge feature map through a spatial transform block and an edge convolution block. The output edge feature map is recalibrated through the SE module, and the recalibrated feature maps are aggregated. The aggregated feature map finally outputs the classification score for the label through the skip-dense network.



(a) Spatial Transform Block



(b) Edge Convolution Block

FIGURE 3. Two Significant Blocks for Backbone Networks.

III. METHOD

This section describes our proposed model, which is significantly influenced by the DGCNN model [12]. Based on the edge convolution, the geometric features (or edge features) between the points were captured, and the edge features were learned. The DGCNN model is constructed based on a multilayer perceptron (MLP), whereas we built a deeper and faster network by adding our own skip-connection network and recalibration block.

A. CLASSIFICATION MODEL

1) PIPELINE MODEL

As depicted in the Fig. 2, the spatial transform block and edge convolution blocks are the major elements of the backbone model. The spatial transform block is designed to align the point cloud input to the canonical space by applying the estimated 3×3 matrix. To estimate this 3×3 matrix, a tensor connecting the coordinate difference between each point and k adjacent points is used (Fig. 3(a)). The coordinate difference between the k nearest neighbor and the coordinates of the point is concatenated. Therefore, as shown in Fig. 3(a), the size of the feature map after the k-NN graph is $n \times k \times (3+3) = n \times k \times 6$. The edge convolution block calculates the edge feature for each point and applies a pooling function to output the tensor of with an $n \times n$ shape. Here, n is the number of points entering as the inputted, and a_n is the size of the applied MLP (Fig. 3(b)).

The edge convolution block is described in detail as follows. Suppose that the F-dimensional point cloud dataset $X = \{p_1, p_2, \cdots, p_n\} \subseteq \mathbb{R}^F$ is inputted. For most 3D point cloud data, $F = 3$, and $p_i = (x_i, y_i, z_i)$. When information such as texture or color is added, F increases. Based on this X, we configured a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ comprising $\mathcal{V} = \{p_1, p_2, \cdots, p_n\}$ as a vertex set and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ representing an edge set. The edge set E is expressed as follows:

$$E = \{e_{ij} \mid e_{ij} = f_\Theta(p_i, p_j) for\ 1 \le i, j \le n\} \tag{1}$$

where f_Θ is a nonlinear function with $\mathbb{R}^F \times \mathbb{R}^F \to \mathbb{R}^{F'}$, and Θ is a learnable parameter. Based on this configured V and E, G is constructed as a k-nearest-neighbor graph and reflected in the edge convolution block. The function f_Θ that defines the edge features is expressed as follows:

$$f_\Theta(p_i, p_j) = \bar{f}_\Theta(p_i, p_j - p_i) \tag{2}$$

This asymmetric function combines the p_i-centered global shape structure and the $p_j - p_i$-centered local neighborhood. Finally, the edge feature of the lth channel through the MLP is expressed as follows:

$$\begin{aligned} e'_{ijl} &= \mathrm{ReLU}(\theta_l \cdot (p_j - p_i) + \phi_l \cdot p_i), \\ \Theta &= (\theta_1, \theta_2, \cdots, \theta_l, \phi_1, \phi_2, \cdots, \phi_l) \end{aligned} \tag{3}$$

After constructing the k-NN graph $\mathcal{G}$ for an n point set X, we perform an edge convolution process with $\mathcal{G}$ as input. In edge convolution, we apply a symmetric aggregation function g to the edge feature related to all the edges that are connected from each vertex. Through this process, the edge feature becomes permutation-invariant. The edge convolution result x_i' from the ith point x_i can be expressed as follows:

$$p'_i = g(f_\Theta(p_i, p_j)) for\ \forall j : (i, j) \in \mathcal{E} \tag{4}$$

TABLE 1. Comparison of typical order invariant methods in terms of the ModelNet40 testset classification accuracy as the metric. The experiments were conducted by the authors in [9].

	accuracy
MLP (unsorted input)	24.2
MLP (sorted input)	45.0
LSTM	78.5
Attention sum	83.0
Average pooling	83.8
Max pooling	**87.1**

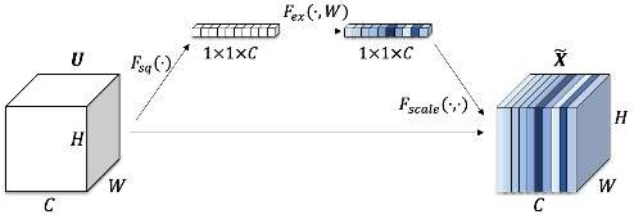


FIGURE 4. Squeeze and excitation model.

This symmetric function takes n vectors as input and outputs a new vector that is robust (or invariant) to the input order. Consequently, given an F-dimensional point cloud with n points, passing through the edge convolution block creates a point cloud with the same number of points in the F' dimension. Methods, such as attention, long short-term memory (LSTM), average pooling, and max pooling, can be used to select the order-invariant function g; we selected the max pooling function g based on Table.1. Therefore, the result of the edge convolution following (4) is as follows:

$$p'_{im} = \max_{j:(i,j)\in\mathcal{E}} e'_{ijl} \tag{5}$$

Because the edge feature function f is a symmetric function, it is invariant to permutation, and the feature aggregation function g, which is max-pooling in our model, is also invariant to the permutation. Therefore, p'_i, the result of (5), is also permutation-invariant to input p_j. Furthermore, the edge feature is preserved when each point is moved by T based on (6). For $\phi_l = 0$, the edge features are fully translation-invariant. In this case, the model utilizes only the features between the points (or edge features) and ignores the geometric information of each point. Therefore, for $\phi_l \neq 0$, by considering p_i and $p_j - p_i$ as input values simultaneously, the model can consider local region information while maintaining the original shape information.

$$\begin{aligned} e''_{ijl} &= \theta_l \cdot ((p_j + T) - (p_i + T)) + \phi_l \cdot (p_i + T) \\ &= \theta_l \cdot (p_j - p_i) + \phi_l \cdot (p_i + T) \\ &= e'_{ijl} + \phi_l \cdot (p_i + T) \end{aligned} \tag{6}$$

2) SE MODULE

In the CNN structure, each convolution filter learns the local feature of an image or feature map—a combination of information in the local receptive field. By passing these combinations through the active function, we deduce the nonlinear relationship, and using the same approach as pooling, reduce the large features so that they can be seen at once. Consequently, CNNs have been able to outperform humans in areas such as image classification because they can efficiently manage the relationship of the global receptive fields. The SE module models the dependency between the convolution features to further enhance the expressiveness of the existing CNN. The SE module consists of a squeeze operation that summarizes the complete information on each feature map and an excitation operation that scales the importance of each feature map. With the SE module, the model performance improvement is significant compared with the increase in the number of parameters, while ensuring that the model and computational complexities do not increase significantly.

The squeeze operation literally squeezes the features. Only import information is extracted from each channel. The concept of extracting core information is important in the subnetwork, where the local receptive field is very small. We use global average pooling (GAP), one of the most common methodologies for extracting core information. The GAP enables global spatial information to be compressed into a channel descriptor.

After squeezing the core information, the module recalibrates through the acquisition operation, which calculates channel-wise dependencies.

$F_{scale}(\cdot,\cdot)$ is a channel-wise multiplication, and $\tilde{X}$ is the feature map of H, W, C size before the squeeze operation. Eventually, the scale value after the excitation operation has a value between 0 and 1. Therefore, it is scaled based on the importance of the channels.

In this study, the SE operation is applied to each feature map through an edge convolution block and then combined to create a point cloud feature. Furthermore, a deeper feature map can be constructed by adding a channel-specific weighted SE operation output at each step. Through this process, high-dimensional point cloud data can be processed more efficiently, and a high learning speed and improvement in performance can be expected with negligible additional computation.

3) SKIP-DENSE NETWORK

We use the output of the backbone network described above as input to the skip-dense network [65] (Fig. 6(a)). A skip-dense network includes stacked fully connected layers with skip connections, expressed as follows:

$$\mathbf{I}_{l+1} = \mathbf{W} \cdot \max(0, BN_{\gamma,\beta}(\mathbf{I}_l)) + \mathbf{b} + \alpha \times \mathbf{I}_l \tag{7}$$

In (7), $\mathbf{I}_l$ is the skip-dense input of the lth layer, $BN_{\gamma,\beta}$ is the batch normalization, and γ and β are the parameters for batch normalization. The next step is the ReLU activation function and a fully connected layer. $\mathbf{W}$ and $\mathbf{b}$ are the parameters of the fully connected layer. α is the coefficient that adjusts the ratio of the skip connections. This pure skip-dense network adds depth to the model while increasing the performance at the cost of a significant increase in the number of parameters and computational complexity. Therefore, the SE module was

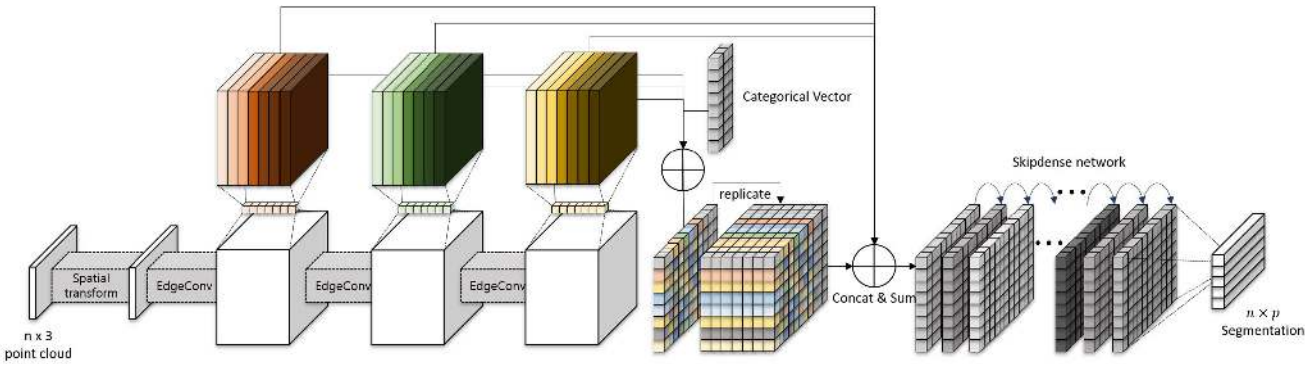


FIGURE 5. Our proposed segmentation model.

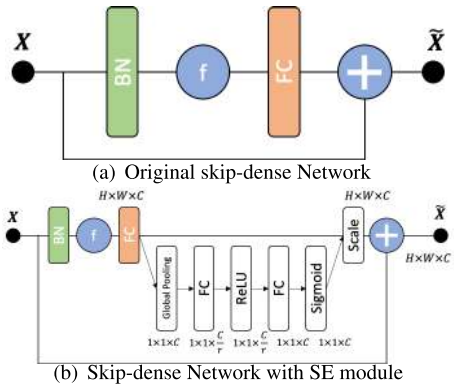


FIGURE 6. Two types of skip-dense Network.

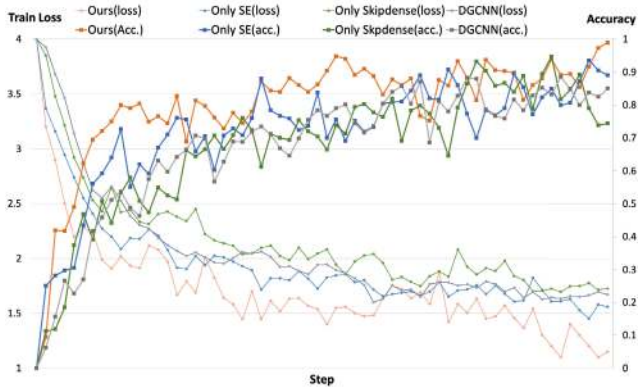


FIGURE 7. Training loss and training accuracy graph of the DGCNN, proposed model, proposed model with only the SE module, and proposed model with only the skip-dense network. The left axis represents the training loss, and the right axis represents the training accuracy.

applied to the skip-dense network to improve the learning speed and performance (Fig. 6(b)).

B. PART SEGMENTATION MODEL

Fig. 5 shows the segmentation model. The segmentation model is similar to the classification model but additionally considers a categorical vector. The difference between the segmentation and classification models is that the categorical vector is aggregated to the recalibrated feature map. By considering the label vector and bundling the point cloud feature and the segmentation label of the point into the same feature map, the effects of learning both local and global information occur simultaneously. Finally, the model predicts an $n \times p$ label for segmentation.

IV. EXPERIMENT

In this section, we introduce the classification and segmentation experiments of our proposed model and the corresponding results.

A. DATASET AND DATA AUGMENTATION

The classification models were tested on the ModelNet40 dataset [19], which includes 12311 CAD modes in 40 categories. This dataset was divided into 9843 validation sets and 2468 test sets. Furthermore, 1024 points were sampled uniformly for each model and resized to fit the unit sphere. The segmentation model was tested on the ShapeNet part dataset [20], which includes 16881 CAD modes in 16 categories, annotated with a total of 50 parts. Each point in this dataset is classified as a part category label. The model learns 2048 uniformly sampled points and consists of up to six part types.

Furthermore, we performed data augmentation to increase the generalization capability of the learning model. As with previous studies, the input point cloud is rotated, shifted, jittered, and scaled randomly, and noise is randomly added at each point. Through data augmentation, the model is trained robustly for rotation and translation.

B. EXPERIMENT IMPLEMENTATION

For the classification experiments, we optimized the model using an Adam optimizer with a learning rate of 0.001 and applied gradient clipping to suppress gradient explosion. We set the number of input points to 1024, batch size to 16, and the momentum of the batch normalization to 0.9. The k-NN graph was set to $k = 20$, the scaling factor of the skip-dense network was set to 0.1, and the dropout rate of the dropout layer was set to 0.5. For the loss function, a cross-entropy function was used, and overfitting was prevented by considering the L2-regularization loss. The segmentation experiment was performed by increasing the number of input points to 2048 and the k value to 30 with the above settings.

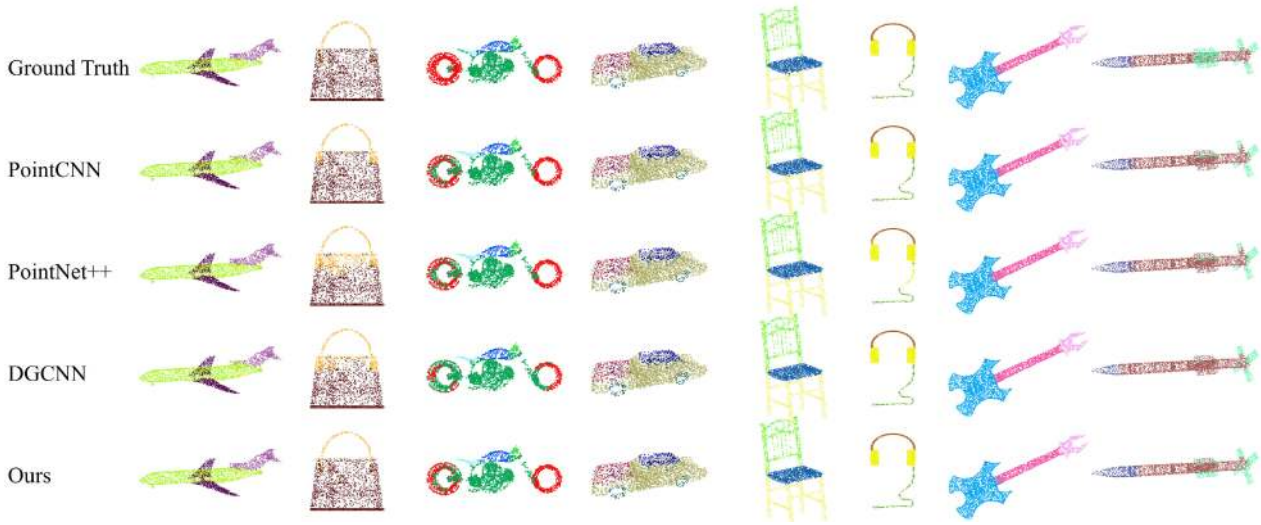


FIGURE 8. Comparison of segmentation results among PointCNN, Pointnet++, DGCNN, Ground Truth and proposed model.

TABLE 2. Classification results on ModelNet40.

Method	#Input Points	Mean Accuracy (%)	Overall Accuracy (%)
VoxNet [46]	1024	83.0	85.9
PointNet [9]	1024	86.0	89.2
PointNet++ [10]	5000	-	90.7
KD-Net [45]	1024	-	91.8
DGCNN [12]	1024	90.2	92.9
PointCNN [66]	1024	88.1	92.2
Ours	1024 points	**91.1**	**93.9**

TABLE 3. Comparison of models' complexity.

Method	Model size (MB)	#Parameters (M)	Computation Time (ms)	Accuracy (%)
PointNet [9]	40	3.48	16.6	89.2
PointNet++ [10]	12	1.48	163.2	90.7
DGCNN [12]	21	1.84	27.2	92.9
Ours	22	1.86	20.2	93.9

The models presented in our paper were implemented using Python and TensorFlow 1.14. The classification experiment was conducted using one 16 GB NVIDIA TESLA V100 GPU, and the segmentation was performed using two 16 GB NVIDIA TESLA V100 GPUs.

V. RESULT

This section introduces the results of experiments with the settings specified in Section IV.

A. CLASSIFICATION EXPERIMENTAL RESULT

First, we introduce the results of the classification model experiment on the ModelNet40 dataset.

1) PERFORMANCE COMPARISON WITH EXISTING MODELS

Table.2 presents a comparison between the results of the previous studies and those of our proposed model. The mean accuracy is the average value of the class-specific accuracy, and the overall accuracy is the ratio of the correct number of classifications to the overall object classification. Except for PointNet++, 1024 points were used as input values. Based on the results, the proposed model has the highest accuracy. Based on the results of the experiments conducted in the same settings, the proposed model exhibited a 1.3% higher accuracy than the previous state-of-the-art DGCNN model.

Table.3 lists the result of evaluating the complexity of the models by comparing the model size, number of parameters, and computation time with other classification networks. Our proposed model includes additional networks, thus increasing the model size and number of parameters. Nevertheless, through the SE module, our model was able to learn more efficiently, resulting in a faster computing time and higher accuracy.

TABLE 4. Effects of SE module and the skip-dense network. The train metric column means the first train step index when the train accuracy reached 90%. ○ indicates the model containing the module, and X indicates the opposite case.

SE module	Skip-dense Network	Mean Accuracy (%)	Overall Accuracy (%)	Train Metric (step)
X	X	90.2	92.9	5865
○	X	90.5	93.0	2487
X	○	90.7	93.4	5908
○	○	91.1	93.9	2390

2) EFFECTS OF APPLIED NETWORKS

Fig. 7 and Table.4 present the effect of SE modules and the skip-dense networks that we additionally applied in this study, each of which increased the classification performance. Moreover, the models with the SE module reduced the loss at a faster rate, and the training metric confirms that the models with the SE module can train 2 to 2.5 times faster than the other models. This demonstrates that the SE module learns more efficiently through the recalibration operation.

TABLE 5. Part segmentation results on ShapeNet part dataset. The values are mean Intersection over Union (mIoU) (%) on points.

	MEAN	AERO	BAG	CAP	CAR	CHAIR	EAR PHONE	GUITAR	KNIFE	LAMP	LAPTOP	MOTOR	MUG	PISTOL	ROCKET	SKATE BOARD	TABLE
# SHAPES		2690	76	55	898	3758	69	787	392	1547	451	202	184	283	66	152	5271
POINTNET	83.7	83.4	78.7	82.5	74.9	89.6	73.0	91.5	85.9	80.8	95.3	65.2	93.0	81.2	57.9	72.8	80.6
POINTNET++	85.1	82.4	79.0	87.7	77.3	**90.8**	71.8	91.0	85.9	83.7	95.3	71.6	94.1	81.3	58.7	76.4	82.6
KD-NET	82.3	80.1	74.6	74.3	70.3	88.6	73.5	90.2	87.2	81.0	94.9	57.4	86.7	78.1	51.8	69.9	80.3
PCNN	85.1	82.4	80.1	85.5	79.5	**90.8**	73.2	91.3	86.0	85.0	95.7	73.2	94.8	83.3	51.0	75.0	81.8
POINTCNN	**85.8**	84.1	**86.5**	86.0	**80.8**	90.6	**79.7**	**92.3**	88.4	**85.3**	96.1	**77.2**	**95.3**	**84.2**	**64.2**	**80.0**	83.0
DGCNN	85.2	84.0	83.4	86.7	77.8	90.6	74.7	91.2	87.5	82.8	95.7	66.3	94.9	81.1	63.5	74.5	82.6
OURS	85.7	**84.6**	82.3	**88.0**	79.2	**90.8**	71.5	91.3	**88.7**	**85.3**	**96.2**	68.0	94.7	80.5	58.9	74.7	**83.3**

TABLE 6. Comparison of results by changing the number of nearest neighbors (*k*) value.

k	Computation Time (ms)	Mean Accuracy (%)	Overall Accuracy (%)
10	14.0	89.7	91.9
20	20.2	91.1	93.9
30	26.2	91.0	93.4
40	32.5	90.4	92.7

3) EFFECTS OF NUMBER OF NEAREST NEIGHBORS *k*

We compared the performance of the model by conducting an experiment where the k value was varied. As k increases, the computation time increases approximately linearly. A similar performance is observed when $k = 30$ and $k = 20$, but decreases when $k = 40$. If the k value is too small or too large, the model does not correctly capture the geometric features around each point.

B. SEGMENTATION EXPERIMENTAL RESULT

This section presents the results of the part segmentation experiment on the ShapeNet dataset.

The intersection over union (IoU) was used as a performance metric for point cloud segmentation. The segmentation model predicts the part label of each point and finds the intersection and union of the actual and predicted labels. The IoU is the number of intersections divided by the number of unions. As depicted in Table.5, some classes exhibit a state-of-the-art segmentation performance, and the total IoU has (marginally) the second-highest performance after PointCNN. The greater the number of shapes, the higher the segmentation performance. Fig. 8 shows an example of the actual segmentation results. For a more realistic comparison of the results, the segmentation results of the poorly predicted classes are also included (such as the motorcycle in Fig. 8).

VI. CONCLUSION

In this study, we developed a faster DGCNN model to classify and segment point clouds. The overall architecture was optimized by adding a skip-dense network to the DGCNN to build an in-depth network and adding an SE module for an efficient learning. This model exhibited state-of-the-art performance on two benchmark datasets: ModelNet40 and ShapeNet.

In terms of the classification, our model exhibited a 1% to 1.7% higher accuracy than the previous models, and the learning speed was 2 to 2.5 times faster. As described in Section V-A2, the skip-dense network enabled the use of a high-performance model by building a deeper network, and the SE module increased the model efficiency, enabling a faster learning process using feature recalibration. Furthermore, for data with large dimensions, such as point clouds, the processing speed is high, so accelerating the learning is a beneficial research result. In the segmentation process, our model achieved the best IoU for 7 out of 16 categories and exhibited the highest overall performance.

In the future, we plan to conduct research on the classification and segmentation with real 3D data as input. For real-time point cloud deep learning research, we plan to apply a model compression method such as quantization. We will also apply the proposed model to higher-dimensional geometric data beyond 3D.

REFERENCES

[1] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 3357–3364.

[2] M. Liang, B. Yang, S. Wang, and R. Urtasun, "Deep continuous fusion for multi-sensor 3d object detection," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 641–656.

[3] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum PointNets for 3D object detection from RGB-D data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 918–927.

[4] S. Wang, S. Suo, W.-C. Ma, A. Pokrovsky, and R. Urtasun, "Deep parametric continuous convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2589–2597.

[5] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, "Towards 3D point cloud based object maps for household environments," *Robot. Auto. Syst.*, vol. 56, no. 11, pp. 927–941, Nov. 2008.

[6] K. Zhang, C. Xiong, W. Zhang, H. Liu, D. Lai, Y. Rong, and C. Fu, "Environmental features recognition for lower limb prostheses toward predictive walking," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 27, no. 3, pp. 465–476, Mar. 2019.

[7] K. Zhang, C. W. de Silva, and C. Fu, "Sensor fusion for predictive control of Human-Prosthesis-Environment dynamics in assistive walking: A survey," 2019, *arXiv:1903.07674*. [Online]. Available: http://arxiv.org/abs/1903.07674

[8] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An information-rich 3D model repository," 2015, *arXiv:1512.03012*. [Online]. Available: http://arxiv.org/abs/1512.03012

[9] 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.*, Jul. 2017, pp. 652–660.

[10] 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. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5099–5108.

[11] Y. Shen, C. Feng, Y. Yang, and D. Tian, "Neighbors do help: Deeply exploiting local structures of point clouds," vol. 1, no. 2, 2017, *arXiv:1712.06760*. [Online]. Available: http://arxiv.org/abs/1712.06760

[12] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, pp. 1–12, Nov. 2019.

[13] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, *arXiv:1502.03167*. [Online]. Available: http://arxiv.org/abs/1502.03167

[14] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1–9.

[15] S. Bell, C. L. Zitnick, K. Bala, and R. Girshick, "Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 2874–2883.

[16] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2016, pp. 483–499.

[17] M. Jaderberg, K. Simonyan, and A. Zisserman, "Spatial transformer networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2017–2025.

[18] J. Hu, L. Shen, and G. Sun, "Squeeze-and-Excitation networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7132–7141.

[19] 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. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1912–1920.

[20] L. Yi, V. G. Kim, D. Ceylan, I.-C. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, and L. Guibas, "A scalable active framework for region annotation in 3D shape collections," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 1–12, Nov. 2016.

[21] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556

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

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2016, pp. 630–645.

[24] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2377–2385.

[25] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng, "Dual path networks," in *Proc. Adv. Neural Inf. Process. Syst.*, ss2017, pp. 4467–4475.

[26] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 4700–4708.

[27] Y. Ioannou, D. Robertson, R. Cipolla, and A. Criminisi, "Deep roots: Improving CNN efficiency with hierarchical filter groups," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 1231–1240.

[28] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 1492–1500.

[29] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017.

[30] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 2818–2826.

[31] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2016, *arXiv:1611.01578*. [Online]. Available: http://arxiv.org/abs/1611.01578

[32] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8697–8710.

[33] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," 2014, *arXiv:1405.3866*. [Online]. Available: http://arxiv.org/abs/1405.3866

[34] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 1251–1258.

[35] M. Lin, Q. Chen, and S. Yan, "Network in network," 2013, *arXiv:1312.4400*. [Online]. Available: http://arxiv.org/abs/1312.4400

[36] C. Cao, X. Liu, Y. Yang, Y. Yu, J. Wang, Z. Wang, Y. Huang, L. Wang, C. Huang, W. Xu, D. Ramanan, and T. S. Huang, "Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 2956–2964.

[37] T. Bluche, "Joint line segmentation and transcription for end-to-end handwritten paragraph recognition," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 838–846.

[38] A. Miech, I. Laptev, and J. Sivic, "Learnable pooling with context gating for video classification," 2017, *arXiv:1706.06905*. [Online]. Available: http://arxiv.org/abs/1706.06905

[39] M. F. Stollenga, J. Masci, F. Gomez, and J. Schmidhuber, "Deep networks with internal selective attention through feedback connections," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3545–3553.

[40] L. Chen, H. Zhang, J. Xiao, L. Nie, J. Shao, W. Liu, and T.-S. Chua, "SCA-CNN: Spatial and channel-wise attention in convolutional networks for image captioning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 5659–5667.

[41] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, "Residual attention network for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 3156–3164.

[42] M. Garimella and P. Naidu. (Aug. 2018). *Beyond the Pixel Plane: Sensing and Learning in 3D*. [Online]. Available: https://thegradient.pub/beyond-the-pixel-plane-sensing-and-learning-in-3d/

[43] 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.*, Dec. 2015, pp. 945–953.

[44] L. Wei, Q. Huang, D. Ceylan, E. Vouga, and H. Li, "Dense human body correspondences using convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 1544–1553.

[45] 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.*, Oct. 2017, pp. 863–872.

[46] 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)*, Sep. 2015, pp. 922–928.

[47] M. Tatarchenko, A. Dosovitskiy, and T. Brox, "Octree generating networks: Efficient convolutional architectures for high-resolution 3D outputs," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 2088–2096.

[48] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view CNNs for object classification on 3D data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 5648–5656.

[49] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond Euclidean data," *IEEE Signal Process. Mag.*, vol. 34, no. 4, pp. 18–42, Jul. 2017.

[50] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009.

[51] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," 2015, *arXiv:1511.05493*. [Online]. Available: http://arxiv.org/abs/1511.05493

[52] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," 2013, *arXiv:1312.6203*. [Online]. Available: http://arxiv.org/abs/1312.6203

[53] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data,"2015, *arXiv:1506.05163*. [Online]. Available: https://arxiv.org/abs/1506.05163

[54] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*. [Online]. Available: http://arxiv.org/abs/1609.02907

[55] F. Monti, M. Bronstein, and X. Bresson, "Geometric matrix completion with recurrent multi-graph neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3697–3707.

[56] F. Monti, K. Otness, and M. M. Bronstein, "MotifNet: A motif-based graph convolutional network for directed graphs," in *Proc. IEEE Data Sci. Workshop (DSW)*, Jun. 2018, pp. 225–228.

[57] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3844–3852.

[58] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "CayleyNets: Graph convolutional neural networks with complex rational spectral filters," *IEEE Trans. Signal Process.*, vol. 67, no. 1, pp. 97–109, Jan. 2019.

[59] J. Masci, D. Boscaini, M. M. Bronstein, and P. Vandergheynst, "Geodesic convolutional neural networks on Riemannian manifolds," in *Proc. IEEE Int. Conf. Comput. Vis. Workshop*, Dec. 2015, pp. 37–45.
[60] D. Boscaini, J. Masci, E. Rodolà, and M. Bronstein, "Learning shape correspondence with anisotropic convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3189–3197.
[61] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model CNNs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 5115–5124.
[62] P. Veli ković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," 2017, *arXiv:1710.10903*. [Online]. Available: http://arxiv.org/abs/1710.10903
[63] O. Halimi, O. Litany, E. Rodolà, A. Bronstein, and R. Kimmel, "Self-supervised learning of dense shape correspondence," 2018, *arXiv:1812.02415*. [Online]. Available: http://arxiv.org/abs/1812.02415
[64] O. Litany, T. Remez, E. Rodolà, A. Bronstein, and M. Bronstein, "Deep functional maps: Structured prediction for dense shape correspondence," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 5659–5667.
[65] K. Kim, B. Seo, S.-H. Rhee, S. Lee, and S. S. Woo, "Deep learning for blast furnaces: Skip-dense layers deep learning model to predict the remaining time to close tap-holes for blast furnaces," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, 2019, pp. 2733–2741.
[66] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "PointCNN: Convolution on X-transformed points," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 820–830.

JINSEOK HONG was born in Suwon, Gyeonggi, South Korea, in 1992. He received the B.S. degree in industrial engineering from Korea University, South Korea, in 2015, where he is currently pursuing the integrated Ph.D. degree with the Department of Industrial Engineering.

Since 2018, he has been with the Artificial Intelligence Research Institute. His research interests include deep learning on vision processing and designing and applying deep learning architectures on real industrial problems and optimization.

KEEYOUNG KIM was born in Anyang, Gyeonggi, South Korea, in 1980. He received the B.S. degree in computer science from Seoul National University, South Korea, in 2004, and the M.S. degree in computer science from Seoul National University, in 2006. He is currently pursuing the Ph.D. degree with the Department of Computer Science, State University of New York, South Korea.

From 2006 to 2010, he was an Engineer with Samsung Electronics. From 2010 to 2016, he was a Researcher with the CEWIT Korea Research Institute. From 2016 to 2019, he was also a Senior Researcher with the Artificial Intelligence Research Institute. Since 2019, he has been with Ingenio AI. His research interests include machine learning algorithms, such as genetic algorithms and deep learning, and diverse application fields, such as prediction and control in smart factory, recognition for CAD and 3D scanning data, adversarial attack for CNN, and medical imaging.

HONGCHUL LEE received the B.S. degree in industrial engineering from Korea University, in 1983, the M.S. degree in industrial engineering from The University of Texas at Arlington, in 1988, and the Ph.D. degree in industrial engineering from Texas A&M University, in 1993. He is currently a Professor with the Department of Industrial Systems and Information Engineering, Korea University. His research interests include system engineering, system simulations, and artificial intelligence.