IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 8 November 2022, accepted 23 November 2022, date of publication 1 December 2022,
date of current version 8 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3226211

SURVEY

Deep Learning Based Point Cloud Processing Techniques

ABDURRAHMAN HAZER[1] AND REMZI YILDIRIM[2]
[1]Graduate School of Natural and Applied Sciences, Ankara Yıldırım Beyazıt University, 06760 Ankara, Turkey
[2]Department of Computer Engineering, Ankara Yıldırım Beyazıt University, 06760 Ankara, Turkey

Corresponding author: Abdurrahman Hazer (195101405@ybu.edu.tr)

ABSTRACT In this study, deep learning techniques and algorithms used in point cloud processing have been analysed. Methods, technical properties and algorithms developed for 3D Object Classification and Segmentation, 3D object detection and tracking and 3D scene flow of point cloud data have been also analysed. 3D point cloud sensing techniques have been grouped as Multi-view, Volumetric approach and raw point cloud processing and mathematical models of them have been analysed. In 3D Object Classification and Segmentation, algorithms are given by analysing it in different categories as Convolutional Neural Network (CNN) based, Graph based, Hierarchical Data Structure-Based Methods and Others. 3D object detection and tracking, Segmentation-based, Frustum-based, Discretization-based analysed as point based and other methods. In each section, deep learning algorithms are compared with respect to applicability to real-time processing, amount of points, and relevance to large-scale or small-scale areas. In the last section, comparisons of point cloud processing methods are made and their advantages and disadvantages are given in the form of a table.

INDEX TERMS Deep learning, point cloud processing, 3D image processing, 3D computer vision, 3D object classification and segmentation.

I. INTRODUCTION

Today, with the development of sensor technology, 2D and 3D imaging techniques are constantly developing. Especially in recent years, the use of 3D sensors such as RGB-D, 3D scanner and Light Detection and Ranging (LiDAR) has been increasing due to the development of 3D sensors and their availability at affordable prices [1], [2]. While the spread of 3D sensors has led to an increase in databases containing 3D data, new research areas are also emerging in the field of computer vision for data processing. 3D image processing has a wide range of uses, from robotics to autonomous vehicles, from medical to defense industry, and from augmented reality to remote sensing [3], [4], [5], [6], [7], [8].

Compared to 2D images, 3D images enable machines which has vision function, to perceive their surroundings better with their depth and rich geometric details. Since 2D cameras project our 3D world to a two-dimensional plane, complex algorithms are needed for the machines to see and perceive the surroundings correctly. These algorithms often cause applications to run slowly or increase the possibility of error. Another disadvantage of 2D images is that they have poor contrast compared to when they are obtained from very light or very dark environments. The weak contrast makes it difficult to perceive the geometric details of the objects, especially the corners or edge information. Since height (z-axis) is added in 3D images, corners, edges or different geometric features of objects can be obtained faster and more accurately. Thanks to their rich features, 3D images are also preferred for many applications such as robotics, autonomous vehicles or smart defense systems. 3D images can be presented in different formats such as point cloud, meshes, volumetric and depth images. Images in point cloud format preserve the geometric properties of the objects since they are not subjected to any decomposition process. Among 3D images, point cloud is a very preferred data type in many computer vision application areas such as autonomous driving, robotics

The associate editor coordinating the review of this manuscript and approving it for publication was Manuel Rosa-Zurera.

and defense industry due to the rich information it contains about objects [9], [10], [11], [12].

Deep learning algorithms, which have been frequently mentioned recently and continue to develop, have shown impressive success in many different computer vision tasks such as object recognition, segmentation, and optical flow using 2D images [13]. However, deep learning algorithms developed for 2D images cannot be directly applied to the 3D point clouds due to being unstructured and irregular. Since the number of public databases for 3D images is increasing day by day, deep learning algorithm studies suitable for 3D space have gained speed recently. The objectives of the methods developed for detecting and understanding images in the form of 3D point clouds are: processing large amounts of data, correctly classifying or segmenting irregular point clouds, object detection and tracking, 3D image reconstruction and scene flow. Many literature reviews have been published for 3D image processing tasks [14], [15], [16], [17], [18], [19], [20].

In this study, a large-scale comparison was made by considering the recently published point cloud perception methods and algorithms with their general equations. In addition, unlike other literature reviews, general mathematical models of the methods and flow diagrams of algorithms are given collectively. Thus, contributing to the ability of readers to analyze the past studies better and to develop new algorithms is aimed. In this study, the methods used in the processing of small and large-scale point cloud data developed for different purposes and the systems suitable for real-time work have also been analyzed.

One of the most important considerations for deep learning algorithms is the minimum amount of data to be used for training. As the amount and variety of data to be trained increases, the variety, and quality of the developed algorithms also increases. Similarly, one of the biggest factors in the recent increase in studies on 3D deep learning is the current increase in the number and variety of data sets created in this field. In addition to the increase in 3D datasets, there is a need to develop deep learning algorithms applicable to real-time systems by effectively processing data in terms of speed and accuracy. In this study, the most up-to-date algorithms developed recently have been analyzed by considering their general equations and models.

Although sensors that can record 3D point cloud data such as LiDAR and RGB-D (Kinect [21], Tango [22], Structure Sensor [23], Intel RealSense [24]) are available and cost-effective, the amount of data collected is increasing. LiDAR sensors are widely used due to their highly accurate measurement capability [2]. The working principle of LiDAR is based on the back reflection of laser beams. After the scattered laser pulses hit the objects, the reflected rays are collected by a receiver detector. The arrival time of the rays reflected from the objects to the receiver is calculated by the receiver, and the necessary information is recorded. Finally, point cloud data is created. Among 3D image sensors, the biggest advantage of LiDAR is the acquisition of sensitive data regardless of day or night.

LiDAR is widely used in the defense industry [6] and remote sensing [2] as well as having an important place in autonomous driving applications.

The working principle of Kinect cameras, which is another 3D imaging sensor, has similar features to LiDAR. Kinect sensors consist of 3 main components: IR transmitter, RGB camera, and IR receiver. There is also a sensor component to digitize the received data. Kinect uses diffraction grating, light coding, and time of flight (ToF) to create a depth image. In other words, optical measurement methods based on delay-time are used. The light codes sent by the IR transmitter are collected by the IR receiver and the RGB camera by hitting all parts of the object's surface. Since the collected beams are scattered from different points of the object, they reach the sensor at different times (ToF). A depth image of the object is created by calculating the delays of the beams when they reach the sensor. 3D data quality differs depending on the sensor and system used. The data obtained from the sensors have different representation types. However, research has focused on point cloud data for three main reasons. The first reason is that the point cloud is the most likely representation type closest to the raw data. Since point cloud does not require any quantization, it provides the most accurate data in sensing the physical world without loss. The second reason is that the point cloud is an easy-to-learn and simple notation type. Compared to a complex representation such as mesh, which has a complex structure due to its polygonal connections and volume, a point cloud is simply a collection of points. The third reason is that computational complexity is avoided as no transformation is needed in the point cloud processing. In this case, the obtained data is ready for direct usage.

Databases used in deep learning algorithms developed for 3D computer vision tasks are classified according to whether they contain single or multiple objects, are obtained from indoor or outdoor space and consist of synthetic or real images [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50].

Databases in which objects are recorded one by one are used for object recognition, classification and to extract the motion characteristics of object parts, which is a new field [27], [28], [29], [30], [31]. These databases are created by modeling objects in the computer environment or scanning real objects. 3D synthetic databases created with the help of computer aided design (CAD) software used for 2D and 3D modeling in engineering, architecture and design are widely used in the field of object recognition and classification [25]. Another software used to create synthetic data is Daz Studio [26]. A new database used for motion prediction of object parts was created by scanning real objects instead of synthetic objects.

Databases that contain multiple objects in a single image frame are often used for object segmentation and motion flow

tasks [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50]. Multi-object databases are obtained by scanning indoor [32], [33], [34], [35], [36], [37], [38], [39], [40], [41] or outdoor spaces [42], [43], [44], [45], [46], [47], [48], [49], [50] using 3D sensors such as LiDAR or Kinect.

Databases created with an object in each image are often used for 3D object classification tasks. ModelNet and ScanObjectNN are the most commonly used for classification among single object-based databases. ModelNet [27] has no real-world problems such as noise that may occur in the real environment due to the computer-generated. The absence of any noise in the data can create a weakness in the tests of the developed applications in terms of accuracy. Unlike ModelNet [27], ScanObjectNN [30] is more suitable for practical use as it is created by scanning the real world and provides more accurate results in the testing of developed applications. Although ShapeNet [28] is mostly used for point cloud generation and completion tasks, experiments have been made in the semantic segmentation task. Shape2Motion [29] has been developed in line with a new type of application recently, the task of segmenting and extracting the characteristics of object movements. Databases created in such a way that there are many objects in each image are used for tasks such as 3D segmentation, scene flow, object detection, and tracking rather than classification. While SemanticKITTI [42], Semantic3D [46], and nuScenes [49] datasets are extensively used for semantic segmentation, S3DIS [33] and ScanNet [36] are frequently used for instance segmentation task. KITTI, nuScenes and ScanNet datasets are generally used for object detection and tracking tasks. The semantic3D dataset, which consists of approximately 4 billion points in total, is used less than other methods in applications despite the intense data it contains. S3DIS, which is generally used for instance segmentation, contains approximately 215 million points, 86 times larger than ScanNet with an average of 2.5 million points. Although nuScenes is approximately 7 times larger than the KITTI dataset, it is the most widely used KITTI database among the studies published as open access.

II. DEEP LEARNING BACKGROUND FOR 3D POINT CLOUD PROCESSING

In this section, deep learning algorithms and learning fundamentals developed for the sensing of 3D images are given. Deep learning algorithms are basically based on two items. The first is to have an algorithm that can extract certain attributes from the data. The second is to learn a mathematical model that can obtain meaningful features from data that is assumed to have the same distribution as the input data, but not the same [51]. Deep learning is actually seen as a large neural network. A single-layer neural networks consisting of an input and an output layer are insufficient for solving very complex tasks. In order to overcome complex problems, multilayer perceptrons (MLP) consisting of many neural networks are used. An MLP consists of an input layer, at least one hidden layer, and an output layer.

When the mathematical model of the operation performed at a node of a single-layer neural network is applied to an input, output y,

$$y = \delta\left(\sum_{n} x_n w_n + b\right) \tag{1}$$

is obtained [51]. In equation (1), δ () denotes activation function, x denotes the input, n is the total number of inputs, w is the weight and b is the bias. Data entering a single-layer neural network are first multiplied by the corresponding weights. Although the weight matrices are determined randomly at the beginning, they are iteratively updated through the training process. The ranges of input data multiplied by the weights are changed by adding the bias which is the linear component. Then, the output data is obtained by applying the non-linear activation function to the data. An iteration in the training process consists of two stages. The first stage is forward propagation while the second stage is backpropagation. Equation (1) describes forward propagation for a single node. After the forward propagation process is completed for all nodes, cost calculation is made using the cost/loss function as the last step. Different loss functions such as mean square error (MSE), binary cross-entropy (BCE), and Categorical Cross Entropy (CCE) are used according to different application areas such as regression, binary classification, and multiclass classification tasks [13]. Although there are more loss functions in the literature, the most used ones are given in this study. Learning algorithms aim to reach the most accurate result by minimizing the cost function in the training process of the data. In order to complete an iteration after forward propagation, the training process of the algorithm continues with backpropagation. In the backpropagation stage, the weight and bias parameters are updated in order to minimize the cost function with optimization algorithms such as gradient descent. These updates are provided by the gradient of the cost function feeding back to the network. Backpropagation is actually the backward derivative of the steps performed in forward propagation. When the weight and bias values updated in each iteration reach a certain desired level, the learning process of the algorithm is completed. One of the most critical stages in the learning process is the activation functions. The activation function helps the classification process by allowing the output to take values in intervals such as (0,1), $(-1, 1)$, $[0, \infty)$ or $(-\infty, \infty)$ according to its properties. Since we live and perceive in a non-linear world, a linear learning structure causes problems in learning nonlinear data. In order to learn nonlinear complex data (image, audio and video, etc.), activation functions are needed that will give learning algorithms a non-linear feature. The most commonly used activation functions are known as Sigmoid, Tanh, ReLU and Leaky ReLU [52].

The sigmoid function $\sigma(x)$ is defined as

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2}$$

where x denotes the input value. When the input value is very high in the equation (for example, it increases towards infinity), the output of the function approaches 1 as the value of e^{-x} approaches zero. On the contrary, when the x value is very low, the e^{-x} value approaches infinity and the function output approaches 0.

The mathematical model of the hyperbolic tangent, $\tanh(x)$,

$$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{3}$$

is defined. While as the input value approaches infinity, the output value becomes 1, as the input value increases in the negative direction, the output value becomes -1 for the tanh function. However, when the input value is 0, the output value will be 0, so there is a third value for tanh.

The mathematical model of the ReLU function is

$$\text{ReLU}(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases} \tag{4}$$

In the ReLU function, the output value is 1 for input values greater than zero, while the output value is 0 for input values less than zero. The mathematical model of Leaky-ReLU is

$$\text{Leaky-ReLU}(x) = \begin{cases} 0.01x & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases} \tag{5}$$

The difference of the Leaky-ReLU function from the ReLU is that the output value is 0.01 times the input instead of 0 for input values less than zero. Different versions of ReLU such as Parametric-ReLU and Bipolar-ReLU are also available in the literature. There are many more activation functions such as SoftMax, softplus and square nonlinearity (SQNL). However, here are the most commonly used ones. Convolutional neural networks (CNN), a sub-branch of deep learning, are used in the development of deep learning algorithms, especially for big data such as images and videos. In a classical CNN architecture, there are convolution, non-linearity, pooling, flattening, and fully connected (FC) layers [53].

For a 2D image matrix, the convolution layer is mathematically expressed in two ways as continuous convolution and discrete convolution. Accordingly, between image matrix, $u(x,y)$, and filter matrix, $v(x,y)$, where (x,y) denotes spatial coordinates, continuous convolution operation $u(x,y)^* v(x,y)$ is

$$u(x,y)^* v(x,y) = \int_{t_1=-\infty}^{\infty} \int_{t_2=-\infty}^{\infty} u(t_1,t_2) \cdot v(x-t_1, y-t_2)\, dt_1 dt_2 \tag{6}$$

where (t_1, t_2) is the coordinates on the spatial plane. Similarly, discrete convolution, $u[x,y] * v[x,y]$, is

$$u[x,y]^* v[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} u[n_1,n_2] \cdot v[x-n_1, y-n_2] \tag{7}$$

where $[n_1, n_2]$ is the coordinates on the spatial plane. In the Convolution layer, features such as curves and edges of the image are extracted with the help of a filter. If a feature extraction by applying a 3×3 matrix size filter (convolution kernel) on a 2D image piece with a 5×5 matrix size is considered, a convolution kernel with a 3×3 matrix size is first applied to the upper left corner of the image. After multiplying the elements of the filter with the image pixel values, they are added together to form the first element of the 3×3 feature matrix. Then, the feature matrix is obtained thanks to the kernel that completes the navigation on the image by shifting one step at a time. The stride of the filter can be changed according to the needs of the algorithm. Thus, the kernel can also be shifted over the image by taking two steps instead of one. Since the shifting filter by taking two steps will make the feature matrix even smaller, the feature matrix obtained by applying zero-padding around the image is prevented from shrinking. With zero-padding, zeros are added around the matrix so that the feature matrix to be obtained at the output contains more information.

The non-linearity layer usually comes right after the Convolution layer. The non-linearity layer is the layer where activation functions are used. Nonlinear data can be learned better by using activation functions. Although the activation function to be used varies according to the purpose of designing the neural networks, the ReLU function is generally used in the intermediate layers, while SoftMax is preferred in the output layer of the classification task.

After providing the nonlinear features to the neural networks, a pooling layer is added for subsampling. The pooling layer reduces the computational burden by reducing the number of parameters used in the neural network. Although there are many different pooling methods, the most commonly used are max-pooling and average-pooling methods. In the max-pooling process, the maximum value in the filter is taken by shifting the filter on the matrix. In the average-pooling process, the average of the values in the filter is taken. The purpose of both methods is to reduce the computational burden by reducing the size of the matrix.

In the flattening layer, the data to enter the FC layer is prepared. Since the FC layer expects the data to be processed in a single line array, the flattening layer prepares the received data by serializing it for the FC layer.

The FC layer, which consists of neural networks, is the most critical and last layer where learning is carried out in CNN. The FC layer is designed specifically for the application according to different purposes such as detection, tracking, classification and segmentation of objects.

Sampling, grouping, and aggregation functions are the three key components used in deep learning algorithms to process 3D point cloud data. Since the point cloud type contains a large amount of data, sampling is needed first in order to avoid computational burden instead of processing at all points. Although many sampling methods such as random sampling (RS) [54], furthest point sampling (FPS) [55] and voxel sampling (VS) [56] are used as sampling functions, the most frequently used sampling method is the FPS. According to the FPS, one sample point is selected at a time and the sample points should be located at the farthest point from each other. For example, after the first sample point in an area of the image is selected according to the FPS, each point to be sampled will be at the farthest distance from the previous point. Thus, sample data with a normal distribution without overlapping is obtained. In fact, the center points of the local regions in the sampling layer are obtained with the sampling function.

Usually, just after the sampling stage, local regions are created by choosing the nearest neighborhoods to the center points in the grouping layer. Different methods such as k-nearest neighboring (K-NN) and ball query are used to find the nearest neighbors. The K-NN method, which is one of the most frequently used and oldest methods, determines k nearest neighbors according to distance formulas such as Euclidean, Manhattan, Minkowski and Hamming [57]. The reference point is classified according to the majority of its k nearest neighbors.

The aggregation layer, which is the last stage, aims to get rid of unnecessary computation by using pooling algorithms as in CNN. One of the two most common methods used for pooling is max-pooling and the other is the average pooling method. The details of different algorithms and structures related to deep learning are given in the section where the methods are explained.

In this section, topics such as the basic deep learning concept, neural network structure, multilayer perceptron, convolutional neural network logic and learning mechanism form the basis of deep learning methods used to process point cloud data. Convolution-based learning techniques are widely used in the processing of 3D images as well as being critical in the processing of 2D images. However, CNN-based networks need transformation functions to process point cloud since CNN architectures developed for 2D images cannot be directly applied to irregular point cloud data. How the CNN structure evolves into 3D images is given in the methods using CNN in the next section. Another technique as important as CNN is MLP. MLP, which provides machine learning by training the weight and bias parameters, is generally used to perform the classification task. Deep learning algorithms can be designed by combining many MLPs. MLP techniques are frequently used for the task of classifying point cloud data. Forward propagation and back propagation concepts that provide the learning function in a neural network or MLP also form the basis of deep learning and even machine learning algorithms. The selection of the loss functions to be used during backpropagation is an important step to ensure correct learning. Since each function used for deep learning has different advantages, the functions to be selected according to the purpose of the developed algorithm vary. Although the necessary structures and concepts for deep learning are given, different deep learning structures such as Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Restricted Boltzmann Machine (RBM), Deep Belief Network (DBN) and Generative Adversarial Networks (GAN) is also included in the literature. However, here are some commonly used structures that can be helpful in point cloud processing. Different deep learning architectures such as RNN and GAN are included in the methods they are used in the next section.

III. POINT CLOUD PROCESSING

Object classification and segmentation tasks have an important place in many different application areas such as robotics, autonomous driving and augmented reality. As classification and segmentation for 3D point cloud data is still one of the challenging tasks, many methods have been developed for these tasks. The deep learning algorithms, approaches and techniques used for the methods developed for classifying and segmenting 3D data are given in this section.

A. MULTI-VIEW BASED APPROACH

Since 3D point cloud data is unstructured and unordered, CNN algorithms developed for regular 2D images are useless for processing point cloud data [9]. The point cloud (raw data) is transformed into a regular structure by pre-processing so that the algorithms that already exist for 2D images can be used for 3D point cloud data. The basis of the multi-view-based approach is based on a similar transformation. In order to use the CNN algorithms developed to percept 2D images with regular structure, 3D point cloud data is first converted into 2D images with different viewing angles. The transformation is achieved by projecting the 3D point cloud data onto two dimensions from different angles. Thus, 2D CNN algorithms are now applicable for 2D images from different angles. Multi-view-based algorithms are used for many different computer vision tasks. The most common methods developed for classification and segmentation are reviewed under this title.

1) MULTI-VIEW CONVOLUTIONAL NEURAL NETWORKS (MVCNN)

The first of the multi-view-based approaches for 3D object recognition and classification tasks is Multi-view Convolutional Neural Networks (MVCNN) [58]. After applying CNN separately to the 2D images converted at different angles, the features are collected in a single global descriptor. While the data learning process is completed in eight layers in total, including five CNN layers and three FC layers, the classification process is performed with the SoftMax activation function used at the end. The seventh layer, which is used as the descriptor of the image, and the previous layer are

connected to each other with the ReLU activation function. In the last stage, data loss occurs in the MVCNN method, since most of the small values in the image are eliminated with max-pooling, which is used to collect the features in a single global descriptor. Since the MVCNN method is based on the processing of 2D images, it has advantages as well as disadvantages. The fact that there are many convolution methods developed on 2D images provides different alternatives to MVCNN that it can use. Data loss caused by using 2D instead of 3D negatively affects the accuracy of classification and segmentation tasks.

2) GIFT

GPU acceleration and Inverted File Twice (GIFT) method has been developed as a projection-based three-dimensional shape search engine for real-time object recognition task. Typically, shape search engines include projection generation, view feature extraction, multiple view matching, and reordering components [59]. Projection-based algorithms, which run slow especially due to the computational burden of the first three components, are generally not suitable for real-time applications. However, the GIFT method has become suitable for real-time object recognition with the help of GPU acceleration by extracting features from the viewpoint without doing any pooling. Multiple images consisting of a minimum of 25 projections for each object can be obtained quickly with the help of the GPU by placing the center of gravity of a 3D object at the starting point of the spherical coordinate system during the projection creation stage. The feature extraction stage, which consists of 5 CNN and 3 FC layers, is also accelerated by the GPU. Unlike MVCNN, GIFT does not make unnecessary calculations by focusing only on pairs that correspond to each other instead of pairs of objects that do not correspond to each other in the multi-view matching stage. Assuming X and Y are two 3D objects, $\{X_i\}_{i=1}^{k}$ and $\{Y_j\}_{j=1}^{k}$ represent the properties of the k-projected images. For the matching of X and Y, Hausdorff distance similarity (HDS) is used as a similarity measure. $HDS\,(X, Y)$, is

$$HDS\,(X, Y) = \sum_{i=1}^{K} \max_{j} \langle X_i, Y_j \rangle \tag{8}$$

where $\langle ., . \rangle$ is the dot product of two vectors, max is the operator that takes the maximum of the values, and K represents the total pair of objects. As can be seen from Equation (8), GIFT only considers the most similar pairs. Finally, the Aggregated Contextual Activation (ACA) contextual similarity measure, which has the same purpose as diffusion for reordering, has been developed to enable the GIFT method to work in real time. Although compatibility with real-time systems provides an advantage for GIFT, one of the disadvantages of GIFT is that it only searches between pairs of images that match each other, thus missing useful information from other non-matching images.

3) MULTI-VIEW HARMONIZED BILINEAR NETWORK (MHBN)

Developed for object recognition and classification tasks, Multi-view Harmonized Bilinear Network (MHBN) aims to create a global identifier by collecting local convolution features with bilinear pooling [60]. Unlike MVCNN and GIFT, MHBN uses the similarity of patches to patches for multi-view matching. Thanks to the patches-to-patches similarity feature, MHBN avoids the computational complexity problem in MVCNN, since only the similarity between the useful local features of the images is taken into account. In addition, MHBN avoids the problem of information loss experienced by GIFT, since it does not pay attention to whether there is a correspondence between the image pairs. If X and Y again represent two 3D objects, the sets of local convolutional properties a and b for both objects can be defined as $\vartheta_X = \{a_1, \ldots, a_K\}$ and $\vartheta_Y = \{b_1, \ldots, b_K\}$, respectively. In this case, for the MHBN method, the generalized polynomial set-to-set similarity (PSSn), $\mathrm{PSS}_n\,(\vartheta_X, \vartheta_Y)$, is

$$\mathrm{PSS}_n\,(\vartheta_X, \vartheta_Y) = \left(\sum_{a \in \vartheta_X} \sum_{b \in \vartheta_Y} \langle a, b \rangle^n \right)^{1/n} \tag{9}$$

where a and b denote local convolutional properties. $\langle a, b \rangle^2$, obtained after subtracting $1/n$ from Equation (9) and assigning $n = 2$, can be written as $\langle a, b \rangle^2 = \left\langle vec\left(aa^T\right), vec\left(bb^T\right) \right\rangle$ using bilinear pooling. In this case, by rearranging equation (9), $\mathrm{PSS}_n\,(\vartheta_X, \vartheta_Y)$,

$$\mathrm{PSS}_n\,(\vartheta_X, \vartheta_Y) = \left\langle vec\left(\sum_{a \in \vartheta_X} aa^T \right), vec\left(\sum_{b \in \vartheta_Y} bb^T \right) \right\rangle \tag{10}$$

is obtained where $\sum_{a \in \vartheta_X} aa^T$ represents the bilinear pooled properties of object X and $\sum_{b \in \vartheta_Y} bb^T$ represents the bilinear pooled properties of object Y. The difference of MHBN from other studies using bilinear pooling is that it can use the PSS feature even more effectively. When converting 3D images to 2D images from different angles, harmonizing process is applied to the bilinear features in order to obtain more distinctive views.

Each of the 2D images projected in the MHBN enters the convolution layer to obtain local features $\vartheta = \{a_i\}_{i=1}^{K}$. The local features then enter the Harmonized Bilinear Pooling layer, whose mathematical models are given below:

1. In the first layer, early sqrt, when root normalization is applied to local features a_i, $\bar{a}_i$,

$$\bar{a}_i = sign\,(a_i) \odot \sqrt{|a_i|} \tag{11}$$

where $sign$ is sign function, $\odot$ is element-wise multiplication, $|.|$ is absolute value operator and $\sqrt{.}$ is square root operation, is obtained.

2. Size of the local features resulting from the normalization process decreases from D to d by convolving with

a kernel size of 1 × 1 × d × D in the convolution layer. When $\bar{a}_i$ enters the sub-convolution layer, $\hat{a}_i$,

$$\hat{a}_i = \mathrm{W}\bar{a}_i + c \tag{12}$$

where W and c are the parameters to be initiated by the Principle Component Analysis (PCA), is obtained.

3. After the convolution layer, the set $\hat{\mathrm{A}} = \{\hat{a}_1, \ldots, \hat{a}_k\}$ containing local features that are reduced in size is subjected to the bilinear pooling process. As a result of the process, the bilinear pooled matrix P is

$$\mathrm{P} = \hat{\mathrm{A}}\hat{\mathrm{A}}^{\mathrm{T}} \tag{13}$$

4. The P matrix is decomposed by the singular value decomposition (SVD) in the harmonizing sublayer as follows:

$$\mathrm{P} \rightarrow \mathrm{U}\Lambda\mathrm{U}^{\mathrm{T}} \tag{14}$$

where U and Λ denote the invertible matrix and the diagonal matrix, respectively. After the harmonizing layer is performed, the output H is

$$\mathrm{H} \leftarrow \mathrm{U}h(\Lambda)\,\mathrm{U}^{\mathrm{T}} \tag{15}$$

where $h(\Lambda)$ denotes the matrix containing harmonized singular values and $h(\Lambda)$ is

$$h(\Lambda)_{m,n} = \begin{cases} \dfrac{\gamma_m^{\varphi_m} - 1}{\varphi_m}, & m = n \\ 0, & m \neq n \end{cases} \tag{16}$$

where m and n are the row-column numbers, $\{\gamma_i\}_{i=1}^{d}$ is the singular values of the input bilinear pooled matrix with the reduced d dimension, and $\{\varphi_i\}_{i=1}^{d}$ is the parameter used by the harmonizing layer for training.

5. The 2D matrix is converted to a 1D vector aiming that the matrix size harmonized with the late-sqrt sublayer does not increase further. The late-sqrt transformation produces output, s,

$$s = sign\,(vec\,(H)) \odot \sqrt{vec\,(|H|)} \tag{17}$$

6. In the last step, ℓ_2 norm sublayer, the output s of the fifth step is subjected to ℓ_2 normalization and $\hat{s}$ is

$$\hat{s} = s \big/ \|s\|_2 \tag{18}$$

Back propagation during the training of the algorithm is obtained by taking the derivatives of all the above steps. When the object recognition comparison of MHBN and MVCNN over ModelNet40 is made, it has been determined that MHBN is superior. The advantage of MHBN is that MHBN can run faster than MVCNN as well as use useful information that GIFT cannot. Unfortunately, data loss caused by processing 2D images also negatively affects MHBN.

4) MULTI-VIEW SPHERICAL PROJECTIONS

The Multi-view Spherical Projections (MVSP) method, which uses spherical projection for 3D object classification, has been developed with a total of two-stage projection processes, one is depth-based projection and the other is image-based projection [61]. The depth calculation is first calculated for the vertices of the semi-regular quad mesh. Thus, depth values $(\cos(\varphi_i)\cos(\theta_j), \cos(\varphi_i)\sin(\theta_j), \sin(\varphi_i))$, are

$$\begin{gathered} \left(\cos(\varphi_i)\cos\left(\theta_j\right), \cos(\varphi_i)\sin\left(\theta_j\right), \sin(\varphi_i)\right) \\ \varphi_i = \frac{180^\circ \cdot i}{x}, \theta_j = \frac{360^\circ \cdot j}{y}, \begin{matrix} 0 \le i \le x-1 \\ 0 \le j \le y-1 \end{matrix} \end{gathered} \tag{19}$$

where (φ, θ) represents the coordinates of the point on the spherical plane, defined. The depth of other points on the sphere is calculated by linear interpolation. Image-based projections are then used. While data loss is minimized during projection with depth information, 2D CNN structures can be used easily thanks to 2D projection. The purpose of the method is to first project 3D images into a spherical domain and then generate neural networks to classify these projections. Each projection forms a cylindrical strip on the sphere surface. After applying CNN separately to each of the cylindrical strips formed on the vertical and horizontal axes, 3D object classification has been realized by combining them. Benchmark was made using ShapeNetCore and ModelNet40 datasets.

B. VOLUMETRIC BASED APPROACH

Irregular point cloud data can be converted to 2D data as well as 3D grid data by voxelizing. The voxelization process is the conversion of point cloud data into a volumetric structure by quantizing a certain size [62]. A voxelization can be done at different densities and resolutions. As the resolution increases, the appearance of the voxel grid begins to resemble the raw data, while the occupancy rate of each voxel decreases. Therefore, high resolution voxel shapes are inefficient due to occupying unnecessary memory space. The biggest advantages of voxel-based approaches are that each voxel has a coordinate and regular structure, allowing 3D deep learning algorithms to be applied easily.

1) VoxNet

VoxNet, developed for real-time object recognition, uses volumetric occupancy grid and 3D CNN together as a solution to the problem of processing large amounts of point cloud data [63]. The data entering VoxNet is first converted to 3D binary occupancy grid and then feature vectors are extracted by passing through fully connected layers in 3D CNN. When a 3D point data enters the VoxNet system, first of all, each point with (x, y, z) coordinates is converted into discrete voxel coordinates (i, j, k). Let the $\left\{z^d\right\}_{d=1}^{D}$ coordinates be a series of distance measurements that pass through $\left(z^d = 0\right)$ or hit $\left(z^d = 1\right)$ a given voxel where d and D denote current iteration and total number of iterations, respectively. Three different

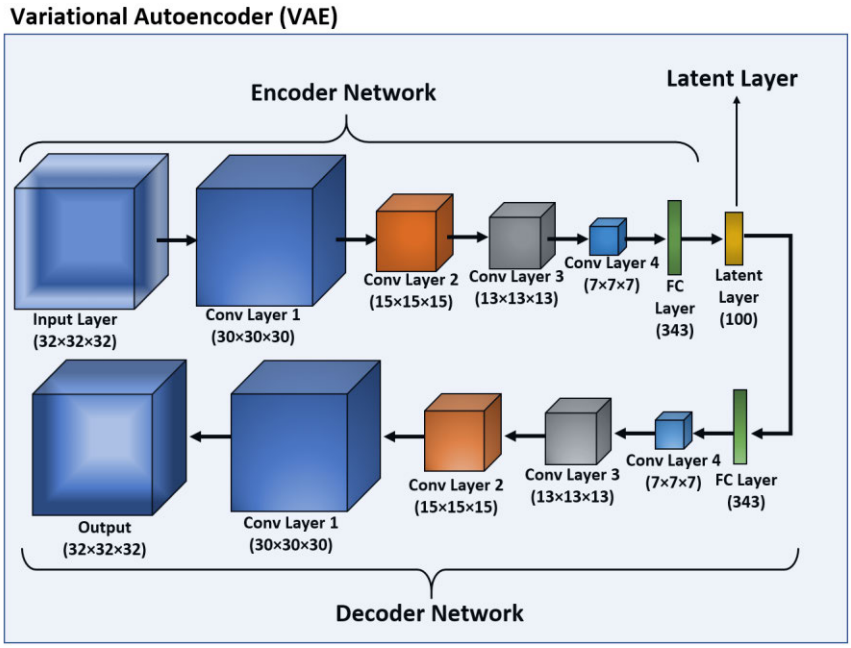

FIGURE 1. Voxception-ResNet flow diagram.

occupancy grid models can be used: binary occupancy grid, density grid and hit grid. For example, according to the binary occupancy grid model, each voxel $\left\{\ell_{ijk}^{d}\right\}_{d=1}^{D}$,

$$\ell_{ijk}^{d} = \ell_{ijk}^{d-1} + z^{d}\ell_{occ} + \left(1 - z^{d}\right)\ell_{free} \tag{20}$$

where ℓ_{occ} and ℓ_{free} represent the log odds ratios of full and empty cells if measurements hit or miss voxel cells, respectively, is updated. VoxNet has been tested with databases containing LiDAR, RGB-D and CAD based images. VoxNet has a very realistic representation as it presents 3D objects by dividing them into cubes. The coordinates of the objects are clearly determined, just like in 2D images, thanks to the realistic representation and division into cubes. In this case, the classification process is successfully completed by applying 3D convolution. However, voxel-based methods generally have a disadvantage, as complex objects are very difficult to voxel. Naturally, the VoxNet also has this disadvantage.

2) VOXCEPTION-ResNet

Voxception-ResNet (VRN), based on ResNet, can train neural networks up to a total of 45 layers by combining ResNet Bottleneck Block and standard ResNet block in a single initial layer [64]. VRN uses the voxel-based Variational Autoencoder (VAE) model given in figure 1. VAE encoder network consists of three main components: encoder, latent and decoder network. While the encoder network consists of 4 convolution +1 FC layers, the decoder network consists of the gradients of the structure used in the encoder network. The latent stage between the encoder and decoder networks consists of 1 latent layer. In the convolution layers at the encoder network stage, $3 \times 3 \times 3$ filters are shifted by 2 steps. The exponential linear unit (ELU) function is used in all layers except the output layer, where the sigmoid function is used as the activation function. The loss function used for the encoder network includes KL deviation, L2 weight rearrangement, and reconstruction error and it is a customized version of Binary Cross-Entropy (BCE). The standard BCE model L is

$$L = -x\log(s) - (1-x)\log(1-s) \tag{21}$$

where x represents the target value in $\{0, 1\}$ and s represents the network output value in each output element (0, 1). For the VRN method, the training of the data has been improved by updating the BCE in two steps. First, the range with the target value is moved to the range $\{-1, 2\}$, while the output value is moved to the range [0.1, 1). By changing the intervals in this way, the magnitude of the loss gradient along the output (s) increases, while the probability of vanishing gradients reduces. Second, by adding a hyperparameter λ to the BCE function, the priority of false positives versus false negatives is weighted. The new BCE model with hyperparameter added, L is

$$L = -\lambda x\log(s) - (1-\lambda)(1-x)\log(1-s) \tag{22}$$

where λ (hyperparameter) is a regularizer. The most significant advantage of VRN is to improve the accuracy percentage in classification tasks as it can train data in 45 layers. A state of art performance has been achieved in object classification with VRN. VRN outperformed VoxNet, MVCNN and FusionNet in classification accuracy in benchmark tests on ModelNet40 and ModelNet10. VRN accuracy rates have also been surpassed with the VRN ensemble version. Although

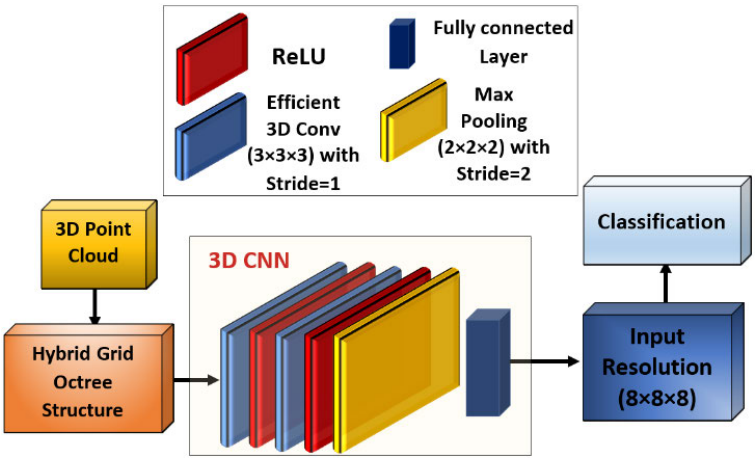

FIGURE 2. OctNet framework for 3D point cloud classification.

VRN has a significant advantage over its peers due to its deep structure, it also has a disadvantage since complex objects are difficult to voxel.

3) 3D ShapeNets

The 3D ShapeNets method has been developed in order to percept scenes converted to a 3D voxel grid by using Convolutional Deep Belief Network (CDBN) [27]. Deep Belief Network (DBN) has been adapted to 3D images with 3D ShapeNets as it is an effective neural network that is frequently used on 2D images. In 3D ShapeNet, the size of a 3D voxel input is first reduced by passing through three convolutional layers. The pooling method, which is down sample, used for a typical CNN is not used for 3D ShapeNets as it is unknown what uncertainty it may cause when reconstructing the shapes. After the convolutional layers, the data entering an FC layer is subjected to 1 layer containing multi-labelling and 4000 hidden units. Benchmark tests of 3D ShapeNets developed for object recognition were also made using ModelNet. 3D ShapeNets also have the advantages and disadvantages of voxel-based methods. In addition, 3D ShapeNets have smaller accuracy values compared to the VRN and VoxNet methods.

4) OCTREE-BASED METHODS

Octree based methods that split 3D voxel inputs into octants by using the octree have been developed in order to overcome the problems such as voxel-based methods occupying too much memory and computational cost [65], [66]. In a 3D cube, an area is divided into 8 cells at a time with the octree method, allowing 3D data to be processed in a hierarchical manner. The first of the octree-based methods developed to process 3D point cloud data is OctNet [65]. Since displaying high-resolution 3D point cloud data with a single octree increases the computational burden of the algorithm, OctNet, taking advantage of the sparsity of the input image, divides the 3D data hierarchically by using a set of hybrid grid octrees instead of an ordinary octree. Hybrid grid octree limits the octree depth of a 3D data using shadow octrees. In Figure 2, the 3D object classification structure with the OctNet method is given as block diagrams. After encoding a 3D voxel with an Octree structure using a bit string, The feature vectors of each voxel are indexed on this bit string with a simple arithmetic operation. Finally, the octants are subjected to classification with 3D CNN. The 3D CNN used for classification consists of 4 layers in total, 2 effective convolution layers, 1 max pooling and 1 FC layer. Although normally, the computational and cost burden is high in the convolution layer, OctNet reduces the cost 76.83% as a result of filtering only the corners, surface and edges of the 3D data thanks to the effective convolution layer [65]. In the network, the layers are connected to each other with the ReLU function. The cost function used when the input data reduced to 83 resolutions with 3D CNN is subjected to the classification process is the cross-entropy loss (CEL) function, which is also frequently used for multiple object classification tasks. CEL, L_{CE},

$$L_{CE} = -\sum_{i=1}^{k} x_i \log(s_i) \tag{23}$$

where k represents the total number of classes, x_i and s_i represent the i th correct class and the probability of the i th predicted class being true, respectively, is calculated. ModelNet10 dataset has been used for 3D classification. For the 3D point data segmentation task, the U-shaped network block diagrams designed for OctNet are given in Figure 3.

In the coding part of the network, after the 3D point data is converted as octree, it reaches the decoder network by passing through 2 convolution layers and 2 max pooling layers. In the decoder stage, the segmentation task is completed by passing the data with reduced resolution through 2 unpooling and 2 convolution layers. In the Unpooling layer, the resolutions of the data which has reduced size before are upsampled by applying nearest neighbor interpolation. While OctNet allows working with high resolution data,

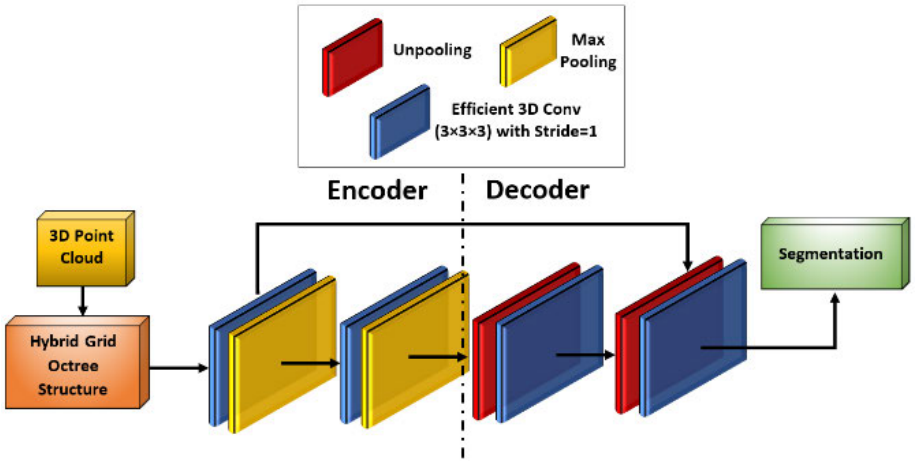

FIGURE 3. OctNet framework for 3D point cloud segmentation.

it minimizes the computational burden thanks to hybrid grid structure. Although the hybrid grid structure puts OctNet in an advantageous position due to reducing the computational burden, the same structure also puts OctNet at a disadvantage since it loses the memory efficient feature provided by octree.

Octree-based Convolutional Neural Network (O-CNN) is another method developed using Octree [66]. In O-CNN, input data is created by taking the average normal vectors of the 3D model sampled at the thinnest leaf octane. Then, object classification is made by applying 3D-CNN on these octanes. O-CNN consists of octree data entering a series of convolution and pooling layers. While the ReLU activation function is used between the layers, the cross-entropy loss function is used in the training of the data. O-CNN's benchmark tests were performed on the ModelNet40 dataset. O-CNN has an advantage over its peers in terms of computation speed and compact storage space. Although octree structures have been developed to address the problems experienced by the voxel approach, it is not an effective viable approach as the complexity of the object increases, making it more difficult to effectively split the object into octants.

5) PointGrid

Although voxel presentation is widely used in algorithms, different solutions have been investigated due to reasons such as the unnecessary space it occupies in memory and high computational costs [67]. PointGrid, which takes a hybrid approach by combining voxel and point representation types is one of these solution methods. PointGrid has reduced information loss by sampling a fixed number of points into each voxel and has improved the extraction of geometric details of objects with 3D convolution. In Figure 4, networks developed using PointGrid for object classification and segmentation tasks are given. After the point cloud data is converted as a point grid, classification is performed with 10 convolution, 4 maxpooling and 2 FC layers in total. In the classification network, the ReLU activation function is used between the layers except the last FC layer where softmax activation is used. Benchmark tests were performed on the ModelNet40 dataset for object classification, while the ShapeNet dataset was used for segmentation. In terms of operating speed, PointGrid is slower than methods that work directly on point cloud data, which will be explored in the next section.

C. RAW POINT CLOUD PROCESSING APPROACH

In both multi-view and voxel-based approaches, point cloud data is first transformed, and then the data that has been converted into a regular structure is processed. Data transformation causes information loss. When a comparison is carried out between multi-view and voxel-based approaches, data loss in multi-view methods is higher than in voxel transformation, since it changes from three dimensions to two dimensions. However, in voxel-based methods, in addition to information loss, unnecessary occupy in the memory caused by empty voxels and excess computational burden are major disadvantages. For these reasons, methods have begun to be developed that can directly process 3D raw point cloud without the need for transformation. PointNet, which is considered to be the first in this field, and subsequent studies are analyzed under this title. The algorithms have been examined in 5 different categories according to their types: pointwise MLP, convolution-based, graph-based, hierarchical data structure-based and others. Pointwise MLP approaches are based on the processing of each point in the point cloud data separately. PointNet, the first pointwise MLP approach, was developed at the University of Stanford.

1) PointNet and PointNet++

PointNet entered the literature as the first method that can process 3D point cloud data without the need for any transformation using the MLP structure [68]. Let point cloud set be $P_c \in \mathbb{R}^{N\times d}$ where N is the total number of points, $c = \{1, 2, .., N\}$ are points, $\mathbb{R}$ and d denote the set of real number and dimension of feature, respectively. Although d

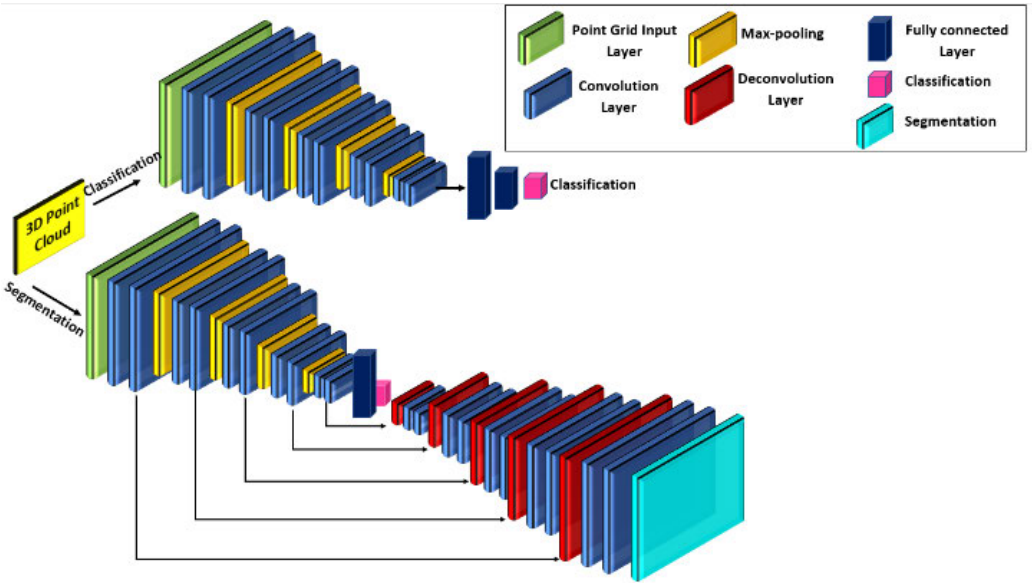

FIGURE 4. PointGrid networks developed for object classification and segmentation tasks.

is generally accepted as 3 since the point P_c is a vector defined as (x, y, z) in the coordinate plane, the value of d can increase due to possibilities of adding extra features. The point cloud inputs P_c are first multiplied by an affine transform matrix estimated by a mini neural network called T-net in the input transform block, providing geometric invariance against rotation changes. The properties of each point in the point cloud data are learned by the system independently by applying several MLP layers to the tensor obtained in the first stage. When it comes to the last stage, the global feature vector is extracted with max-pooling. Thus, the features of the important points have been extracted and precautions have also been taken against data loss. In an unordered point cloud data set $c_i \in \mathbb{R}^d$, where $\{c_1, c_2, \ldots, c_n\}$ denotes points, let $\hbar$ is a continuous function that maps any point cloud set ψ to a vector ($\psi \rightarrow \mathbb{R}$). In this case, mathematical model of PointNet can be defined with the continuous function, $\hbar(c_1, c_2, \ldots, c_n)$,

$$\hbar(c_1, c_2, \ldots, c_n) = \chi\left(\underset{i=1,\ldots,n}{\text{MAX}} \{\lambda(c_i)\}\right) \tag{24}$$

where MAX represents the take maximum operator, χ and λ usually represent multilayer perceptron (MLP) networks. λ is used to spatially transform points, while χ can approximate any continuous function that varies according to permutations of points. As a result of the comparison made in [68], while the accuracy rate of VoxNet decreases almost linearly as the data loss increases, the accuracy rate of PointNet decreases only when the data loss reaches 80%. PointNet can perform most popular 3D computer vision tasks such as object classification, part segmentation and semantic segmentation. PointNet also forms the basis of many subsequent point cloud processing techniques. In addition to being simple and effective, PointNet also has advantages such as immutability against geometric rotation, robustness against data corruption and loss.

The disadvantage of PointNet is that it cannot extract local relationships between points as it learns the points independently and extracts global features. For this reason, the hierarchical PoinNet++, which is a higher version, has been developed [69]. Unlike PointNet, which includes a single max-pooling layer, PointNet++ essentially includes a set of feature abstraction levels consisting of 3 basic layers: sampling, grouping, and PointNet. Just like in CNN's computer vision tasks, hierarchical local features are gradually extracted by the combination of many abstraction levels. The entire network structure of PointNet++ developed for object classification and segmentation tasks is given in Figure 5. In the first step, local features are extracted using a series of abstraction levels. The first layer of abstraction is sampling using the furthest point sampling (FPS) method. Sampling, similar to pooling and convolution, is implemented to reduce the resolution when transition between layers. While the resolution of the point cloud used as input is $N \times (d + C)$, where N is the number of points, d is the number of features, whose value is 3 since features in the first stage are only 3D coordinates (x, y, z), and C denotes the feature domain (RGB, normal, etc.), the resolution of point cloud after sampling is $N' \times \left(d + C'\right)$ where N' and C' denote the sampled point and the new feature vector, respectively. After sampling, the point cloud data that enters the grouping process with the ball query method has a $N' \times K \times (d + C)$ where K represents the number of points in the neighborhood of the center points, resolution at the output. Ball query is another method that can be used for grouping. Compared to K-NN, ball query method is generally preferred for local feature extraction, since the ball query method guarantees a constant region scale of the local neighborhood. Finally, in the PointNet layer,

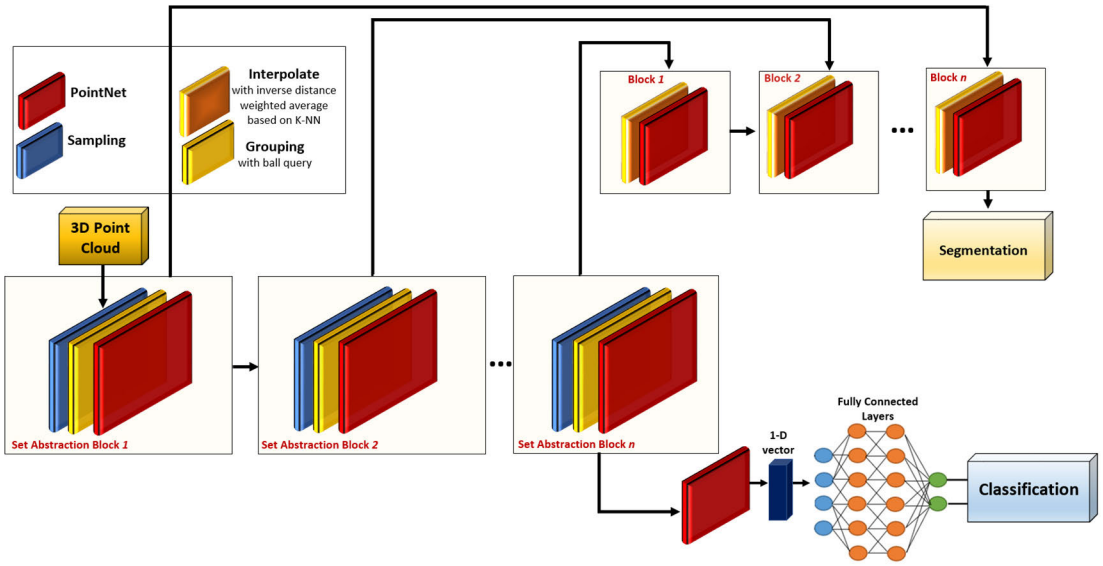

FIGURE 5. Overall framework of PointNet++ method.

local feature vectors are generated by summarizing each local region into neighborhoods which are central points. A level of abstraction is completed with PoinNet that is the last stage of the abstraction level. The first stage of PoinNet++, the hierarchical layer, is completed by using successive set of abstraction. As can be seen in Figure 5, after the first hierarchical layer is completed, the algorithm is completed by continuing in separate layers according to their task as classification or segmentation. The data whose resolution has been reduced for the segmentation task are increased to previous resolutions by using blocks containing interpolation and unit PoinNet algorithms. Inverse distance weighted average based on k-NN technique is used for interpolation. While the PointNet layer is used in the first step for the classification task, classification is carried out by applying a series of FC layers to the data reduced to one dimension in the next stages.

2) PointSIFT

The PointSIFT model was developed, inspired by the scale-invariant feature transform (SIFT), which was developed to detect and define local features in 2D images [70]. The PointSIFT module basically consists of a set of orientation encoding (OE) units that can encode data from 8 specific directions and contain 3 orientation encoding convolution (OEC) layers. The invariance of the image against scale changes is ensured due to the SIFT approach. In OE structure, firstly, 8-octant neighbor points closest to the center point from point cloud data are found by 8-neighborhood (S8N) search method. OEC applied to the nearest 8 points consists of 3 sub-convolutions applied to a cube of $2\times2\times2\times$d in (x, y, z) directions. If the feature vectors obtained with the 3-stage OEC layers are denoted as f, the feature vectors (f_x, f_{xy}, f_{xyz}) are

$$f_x = \lambda\left(conv\left(W_x, f\right)\right) \in \mathbb{R}_{1\times2\times2\times d}$$

$$f_{xy} = \lambda\left(conv\left(W_y, f_x\right)\right) \in \mathbb{R}_{1\times1\times2\times d}$$
$$f_{xyz} = \lambda\left(conv\left(W_z, f_{xy}\right)\right) \in \mathbb{R}_{1\times1\times1\times d} \quad (25)$$

where λ represents the ReLU activation function and (W_x, W_y, W_z) represents the weight of the convolution operator. After the orientation features are obtained with OEC, the features f_{xyz} can be used in point-based data processing algorithms. Therefore, PointSIFT actually improves the performance of the methods in terms of both speed and segmentation accuracy by integrating existing point cloud sensing algorithms. Experimental results were obtained by integrating PointNet++ and PointSIFT. Figure 6 shows the network designed by integrating PointSIFT into PointNet++ for the semantic segmentation task. The set abstraction (SA) and feature propagation (FP) layers used in PoinNet++ are used in the encoding and decoding stages, respectively. Benchmark tests of PointSIFT, which is a suitable method for semantic segmentation, were made using S3DIS and ScanNet data sets. Tests show that PointSIFT has accuracy rate bigger than PointNet and PoinNet++. PointSIFT increases the classification accuracy by collecting all the information in different directions without loss thanks to the OE module.

3) POINT NEIGHBOURHOODS-BASED SEGMENTATION

Engelmann et al. developed a method based on neighborhoods to perform the semantic segmentation task in point cloud data [71]. The whole structure of the method is given in figure 7. In the first step, the features of the data are transformed using a feature network, which consists of a series of feature blocks, each of which is a network similar to a simplified PointNet. Then, two different grouping techniques are applied to the data, using k-means clustering and k-NN to extract features in neighborhoods. The first grouping technique is the feature space neighborhood extraction module

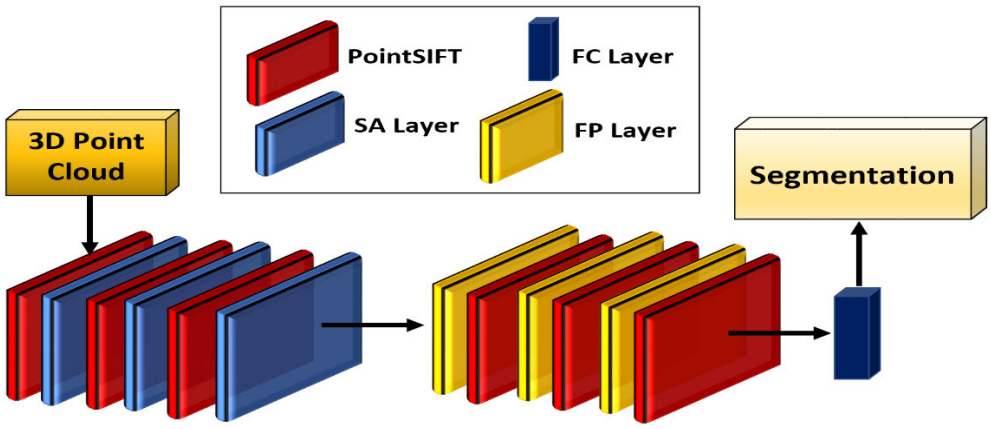

FIGURE 6. PointSIFT segmentation block diagram by integrating PointNet++.

Nh_F, which consists of k-NN, MLP and max-pooling blocks. The second grouping mechanism is the world space neighborhood extraction module Nh_W using k-means clustering. Three Nh_F modules, one Nh_W module, two MLPs and one max-pooling operation are used for semantic segmentation. Two special loss functions, centroid loss ℓ_c and pairwise loss ℓ_p have been added to the method in order to further detail the properties of the points learned with the developed method. As a result of benchmark tests for segmentation tasks using S3DIS and ScanNet datasets, the point neighborhoods-based segmentation method gives better results than PointNet++ in terms of accuracy.

4) MOMENT NETWORK (MomentNet)

Moment network (MomentNet) is another classification and part segmentation method based on PointNet [72]. Figure 8 shows the MomentNet structure for the classification task. MomentNet, which is structurally very similar to PointNet, only differs from PointNet in that point cloud data transformed with spatial transform at the input feeds a simple 2nd-degree polynomial layer. If a point cloud set is defined by $P = (p_1, p_2, \ldots, p_n)$, where P and n denote the points and the total number of points, respectively, the location of the second point in the 3D coordinate plane can be given as $P_k = (x_k, y_k, z_k)^T$. In this case, the 2nd- order polynomial layer is defined as $\left(x, y, z, x^2, y^2, z^2, xy, xz, yz\right)$. Global features of the data coming out of the second-order polynomial layer are extracted by passing it MLP layers with the max-pooling process. Finally, the data entering the fully connected (FC) layer is subjected to classification and segmentation processes. While the MLP and FC layers are connected to each other with ReLU activation functions, the activation function used in the last classification step is the softmax function. As a result of the tests performed on ModelNet40 for the classification task, MomentNet passed PoinNet++ in terms of accuracy, while as a result of the tests performed on ShapeNet for the segmentation task, MomentNet lagged behind some methods in terms of accuracy in the classification of objects such as cars, chairs, and bags. While MomentNet is better and superior to its predecessors in terms of memory and computational complexity, it is less advanced in terms of accuracy.

5) POINT ATTENTION TRANSFORMERS (PATs)

Point Attention Transformers (PATs), developed in accordance with Point cloud classification and segmentation tasks, use three basic components: Absolute and Relative Position Embedding (ARPE) module, Group Shuffle Attention (GSA) and Gumbel Subset Sampling (GSS) [73]. In Figure 9, the entire network structure for classification and semantic segmentation is given. In the first step, the positions of point cloud data relative to their neighbors by entering the data into the ARPE module that use the PointNet mathematical model is calculated. Thus, the function, $ARPE\ (c_k)$ is

$$\begin{aligned} ARPE\ (c_k) &= \chi\left(\max\left\{\lambda\left(c'\right)|c' \in C'_k\right\}\right) \\ s.t.C'_k &= \left\{\left(c_k, c_j - c_k\right)|j \neq k\right\} \end{aligned} \tag{26}$$

where $C = \{c_1, c_2, .., c_k, \ldots, c_N\}$ denotes the point cloud set, C'_k is the set of positions for the k point, the max is maxima operator, χ and λ denote the MLP functions. After the position detection, the relationship between the points is defined using the GSA layer and the GSA layer is

$$\begin{aligned} GSA\ (C) &= GN\ (\gamma\ \{GA\ (C)\} + C) \\ s.t.GA &= concat\left\{Att_\sigma\ (C_i, C_i)\,|C_i = C^{(i)}W_i\right\}_{i=1,..,s.} \end{aligned} \tag{27}$$

where GN denotes group normalization, GA denotes group attention, γ is channel mixing function (reshape, transpose, flatten etc.), C denotes input data, Att is attention layer, σ denote ELU activation function, W is weight and s is total group number. Attention function, $Att_\sigma\ (q, C)$ is

$$Att_\sigma\ (q, C) = \hbar\,(q, C)\,\sigma\ (C) \tag{28}$$

where q and $\hbar$ denote query and SoftMax functions, respectively. GSA is also a structure that provides permutation invariant and differentiable. Output data of the GSA

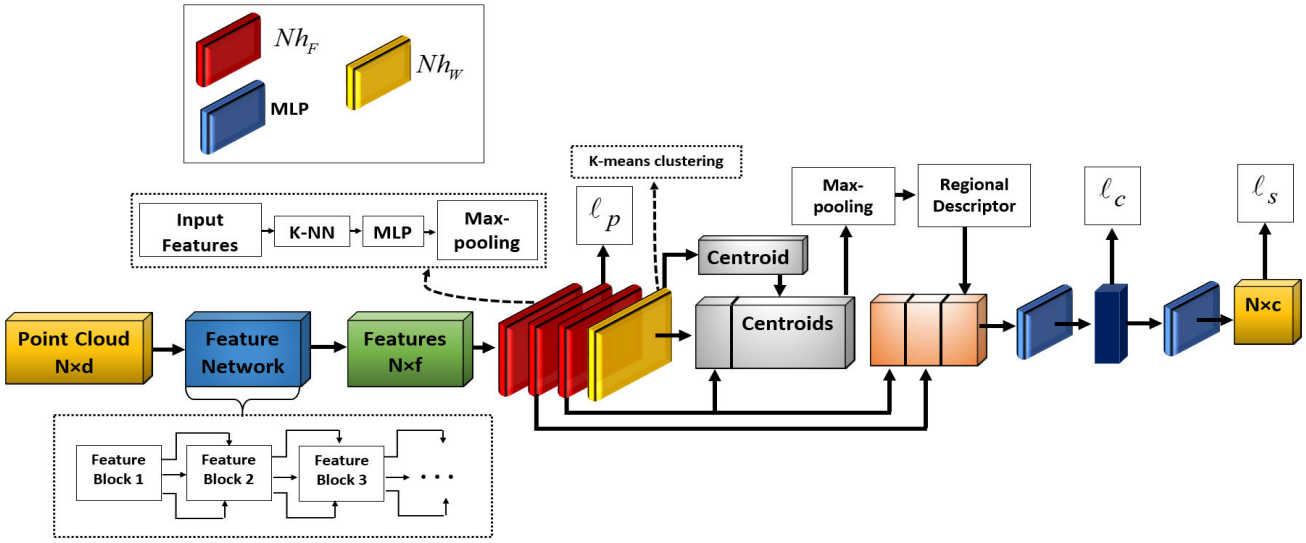

FIGURE 7. PointNet neighbourhoods-based 3D semantic segmentation.

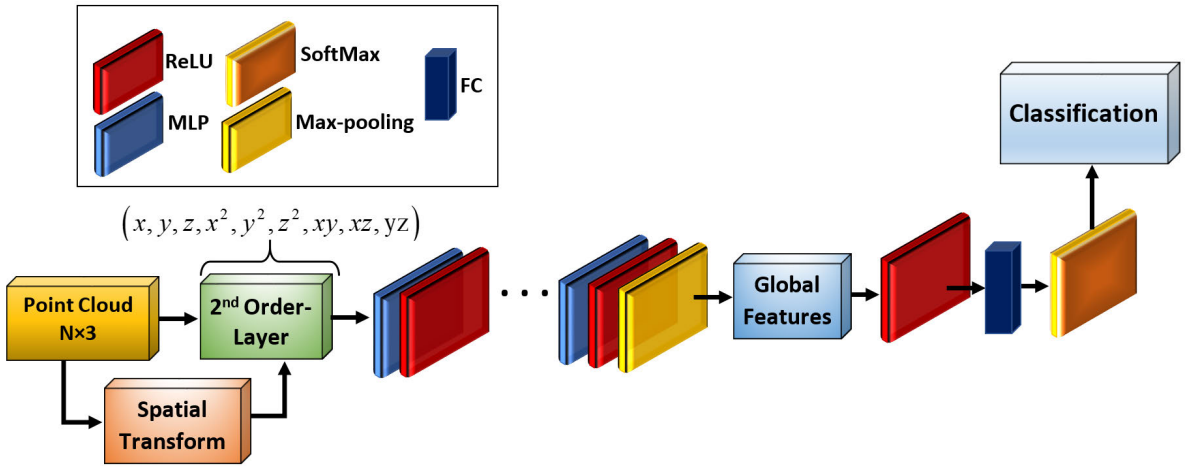

FIGURE 8. Moment Network for 3D shape classification.

layer is then divided into two branches according to the segmentation and classification tasks. Gumbel Subset Sampling (GSS) technique has been developed instead of Furthest Point Sampling (FPS), which is frequently used in point cloud processing as downsampling. GSS function, $GSS\ (C_i)$ is

$$GSS\ (C_i) = GS\left(WC_i^T\right).C_i$$
$$s.t.GS = \text{softmax}\left(\left(\log\left(d\right) + x\right)/\text{-}\lambda\right) \quad (29)$$

where GS denotes the gumble softmax function, W is the learnable weight parameter, d is the input data, x is the gumbel noise, and $\text{-}\lambda$ is the annealing temperature. GSS provides the extraction of hierarchical features. GSA and GSA-Downsampling layers are applied n times sequentially, and then the high-dimensional properties of the data are learned using MLP layers. Although PATs method performs poorly in terms of accuracy compared to other methods in the tests performed with ModelNet and S3DIS data sets, it showed the best performance in the gesture data set recorded with the event camera which has dynamic vision sensor (DVS128).

6) LOCAL SPATIAL AWARE (LSA) NETWORK

L. Z. Chen, X. Y. Li et al. determined that fine-grained details were missed in point cloud data since the spatial distribution of the points was not paid much attention in previous point cloud processing methods, and they developed the Local Spatial Aware Network (LSANet) method by considering the spatial distribution to carry out classification and segmentation tasks [74]. LSANet, which uses the PointNet++ structure in its core, mainly uses spatial feature extractor (SFE) and LSA components. The structure of LSANet for classification and segmentation tasks is given in figure 10. First of all, the characteristics of the point itself and the local region in which it is located are extracted by applying SFE only on the spatial

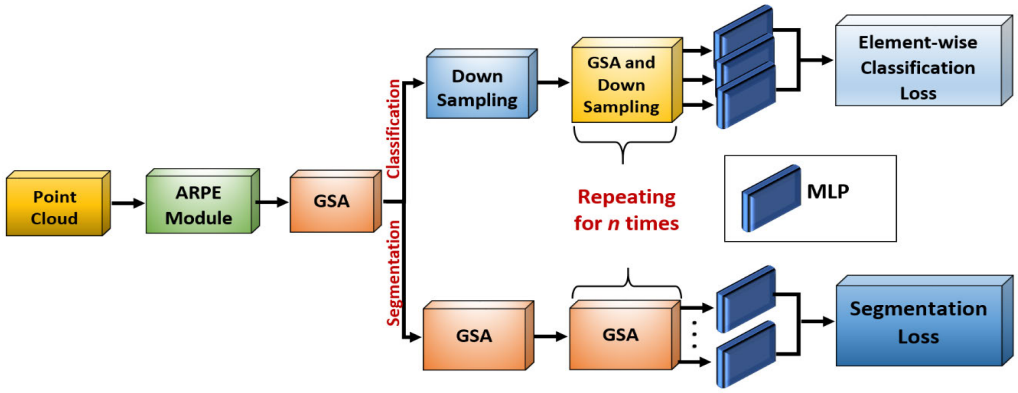

FIGURE 9. Overall framework of PATs.

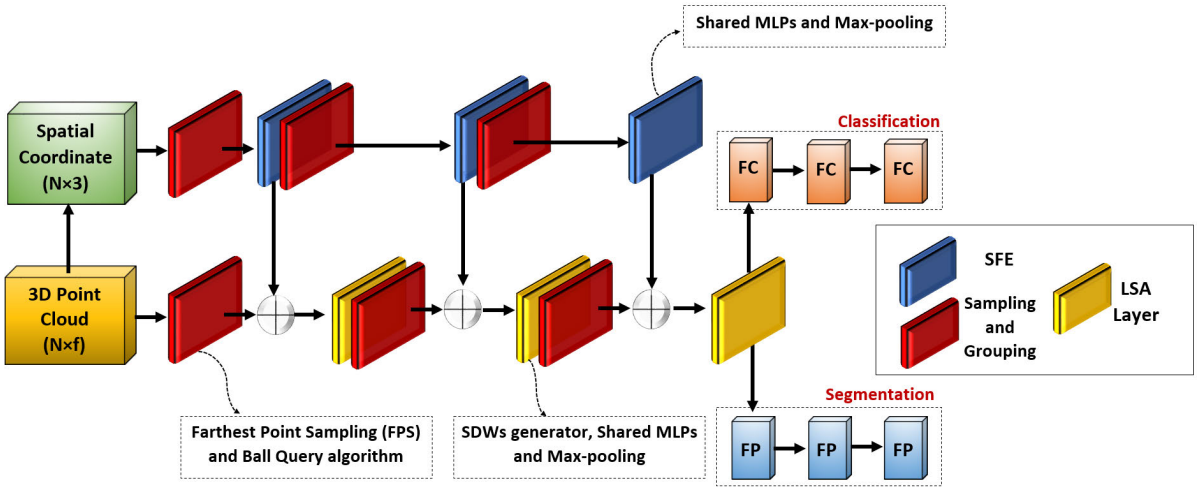

FIGURE 10. Overall framework of LSANet.

coordinates of the point cloud data. The spatial feature of the j-th point (X_j) from N points in a local region, SFP_j is

$$SFP_j = W_0 X_j \tag{30}$$

where $SFP_j \in \mathbb{R}^{64}$ represents the spatial property of the point and $W_0 \in \mathbb{R}^{64\times 3}$ represents the weight parameter. The spatial feature of local region (SFLR) is

$$SFLR = \frac{1}{N}\sum_{j=1}^{N} W_1 X_j \tag{31}$$

where N represents the total number of points in the local area. W_1 is applied to each point as a constant to ensure permutation invariance. As seen in Figure 10, in the sampling and grouping layer, sampling is done with Farthest Point Sampling (FPS) as in PointNet++, while grouping is done with ball query algorithms. The point cloud data passing through the sampling and grouping layer is combined with the features extracted only from the coordinate information to go to the LSA layer. LSA layer can learn to create Spatial Distribution Weights-(SDWs) of points in local regions of point cloud. Thus, while the geometric properties of the points can be learned more accurately, the properties of fine-grained details can be extracted. SDWs, c_j^k, are

$$\begin{aligned} c_j^k &= \sigma\left(W_p^{k-1}\left(c_j^{k-1}\right)\right) \\ s.t. c_j^{k-1} &= c_j^1 = \sigma\left(W_p^1\left(P_j\right)\right) for \quad k = 2 \end{aligned} \tag{32}$$

where σ is sigmoid function, k denotes total number of points and P_j denotes point cloud set $P_j = \{p_1, .., p_j., p_k | j = 1, .., k\}$. As a result of the benchmark tests of the method with ModelNet40, ShapeNet, ScanNet and S3DIS datasets, the accuracy performance of LSANet is better than other methods except for a CNN-based method. LSANet is not applicable for real-time systems like many other methods despite its high accuracy rate, since the running speed of the algorithm is low.

7) PointWeb

PointWeb, which is based on PointNet++, has been developed to better determine the properties of points by

connecting all points in a local area [75]. An adaptive feature adjustment (AFA) module has been added to the method in order to understand the interaction of the points with each other. In the PointWeb structure, which uses the same PointNet++ structure given in Figure 5, the AFA module is added just after the grouping stage. In a point cloud data, if $\psi = \{\chi_1, \chi_2, .., \chi_K\}$ is the feature set in a particular local region (L), let the enhanced features using the AFA module be defined with χ'_m. Thus, enhanced feature, χ'_m is

$$\chi'_m = \chi_m + \sum_{n=1,(n\neq m)}^{K} \lambda_n^{(m)} . (\chi_n - \chi_m), \tag{33}$$

where

$$\lambda_n^{(m)} = \begin{cases} -\tau_{imp}(\chi_m, \chi_n) & \text{for } n \neq m \\ 1 + \tau_{imp}(\chi_m, \chi_m) & \text{for } n = m \end{cases} \tag{34}$$

In equation, the variable $\tau_{imp} = MLP(f(\chi_m, \chi_n))$ where MLP and f represent multi-layer perceptron and coupling function, respectively. After the point cloud data entering PointWeb is sampled with Farthest Point Sampling (FPS), the neighborhoods of the points in their local regions are determined with K-NN. Then, a fully connected network is established between the points by using the AFA module and these points are arrived at the last max-pooling stage passing through several MLP layers. The difference between PointWeb, which uses the PointNet++ structure, and PointNet++, which uses only max-pooling to extract features in each local region, is that PointWeb creates a fully connected point network between the points in the local regions. As a result of benchmark tests using S3DIS and ModelNet40 datasets, PointWeb, which is suitable for classification and segmentation tasks, gives better results than other methods in terms of accuracy performance in some objects, but worse in some objects. Although PointWeb improves classification performance by improving feature extraction, it is not suitable for real-time applications in terms of runtime.

8) STRUCTURAL RELATION NETWORK (SRN)

J. Lu et al., who consider what the local features extracted from the point cloud data are obtained independently from each other a shortcoming, have also developed the Structural Relation Network (SRN) module at about the same time as PointWeb [76]. SRN, which is a plug and play module, can be directly applied to methods that process point cloud data. If the geometric and spatial relationships between the points in two different local regions m and n are denoted by g and ℓ, respectively, learned relational features with the SRN module is denoted as χ_m. Thus, χ_m is

$$\chi_m = \hbar\left(\sum_{\forall n} \nrightarrow \left(\psi_g(g_m, g_n), \psi_\ell(\ell_m, \ell_n)\right)\right) \tag{35}$$

where $\hbar$ denotes the function that provides a single feature to obtain the relational features in the regions, $\nrightarrow$ is the function that combines geometric and local features, ψ_g is the function that extract geometric features and ψ_ℓ is the function that extract local features. SRN has been tested by applying it to PointNet++ [76]. Point cloud data is first subjected to sampling and grouping operations, and then to operations at the PointNet layer. Then, the global feature of the object is extracted from the geometric and spatial features of the data in the SRN layer, and this process is repeated once again. Finally, classification or segmentation is performed. While ModelNet40 and ScanNet were used for classification in benchmark tests, ShapeNet dataset was used for segmentation.

9) JUSTLOOKUP

The JUSTLOOKUP method has adapted the lookup table logic to the detection of point cloud data in order to accelerate the slow-running deep learning algorithms [77]. JUSTLOOKUP has given the workload to memory rather than network by recording the data trained with PointNet in look-up tables. Since point cloud data are processed only according to their location (x, y, z) in the coordinate plane in PointNet, look-up tables were created using only three columns with the developed method. Thus, rapid classification or segmentation is possible. According to the benchmark tests made with ModelNet and ShapeNet, the JUSTLOOKUP method, which works in only 1.5 ms and obtains results, works 32 times faster than PointNet. However, JUSTLOOKUP lags behind other methods in terms of accuracy.

10) ShellNet

Zhang et al. developed ShellNet using ShellConv to better understand point cloud data, train data faster, and establish more effective structures that are not complex [78]. ShellConv, works by analyzing statistics obtained from data in concentric spherical shells. The properties of the points extracted using ShellConv, $P(x)^k$, is

$$P(x)^k = \sum_{\zeta \in \delta_x^k} w_\zeta^k P(\zeta)^{k-1} \tag{36}$$

where x denotes point cloud data, k denotes layer, ζ denotes shell, $P(\zeta)$ shell properties and δ_x point cloud cluster. In the first stage, the point cloud data were randomly sampled and each sample was scattered within the spherical shells. Each shell is subjected to max-pooling process and obtained new shell properties $P(\zeta)$ is

$$P(\zeta) = \max\left(\{P(y) : y \in \delta_\zeta\}\right) \tag{37}$$

where max is the max-pooling operator, y is the neighbor points, $P(y)$ is the properties of the neighborhoods, and δ_ζ is the shell set. Output feature is obtained by applying 1D convolution process from inside to outside, respectively, to the rings (shell) where the features extracted by the max-pooling process are located. Figure 11 gives the configuration of the ShellNet structure for classification and segmentation tasks. Benchmark tests of the ShellNet method were performed

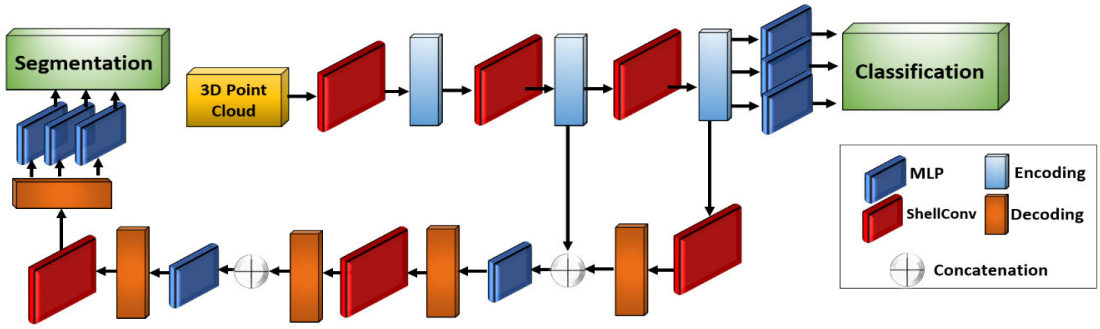

FIGURE 11. Overall framework of ShellNet.

using ModelNet40 for classification, while ShapeNet, ScanNet, S3DIS and Semantic3D were used for segmentation processes. ShellNet uses memory effectively while minimizing the training time of data compared to previous methods.

11) RandLA-Net

Qingyong H. et al. have found that algorithms developed with sampling methods which has high computation and memory costs are generally suitable for working with small data sets [79]. In order to deal with big data, RandLA-net method has been developed, which avoids heavy computation by using random sampling method that is a cost-effective function. RandLA-Net includes two basic modules, random point sampling and local feature aggregation (LFA). Point cloud data is first sampled by using random sampling method with computational complexity of O(1). Since the computational cost is a fixed value, the memory and computational burden of the random sampling method is also fixed when working with big data. However, an LFA module, which will enable more detailed extraction of geometric features, has been developed in RandLA-Net since random selection of points with random sampling method is a problem. The LFA module, on the other hand, consists of three sub-modules: Local Spatial Encoding (LocSE), Attentive Pooling (AP) and Dilated Residual Block (DRB). In order to increase the features to be extracted from the sampled point cloud data, the points are first entered into the LocSE module. In LocSE, firstly, the neighbourhood of the j th point is determined by using k-NN. Then, relative point position encoding (RPPE) is applied to the point in the center of the neighbouring point set $\left\{x_j^1, \ldots, x_j^m, \ldots .x_j^K\right\}$ for each point. Encoded relative point position τ_j^m is

$$\tau_j^m = MLP\left(x_j \oplus x_j^m \oplus \left(x_j - x_j^m\right) \oplus \left\|x_j - x_j^m\right\|\right) \quad (38)$$

where $MLP()$ represents multi-layer perceptron, $\oplus$ represents the concatenation operator, $\|.\|$ represents the Euclidean distance calculation operator between the neighbourhoods and the center point, x_j and x_j^m represent the position information of the point and the central point (x, y, z) adjacent to this point. In point feature enhancement, which is the last step of LocSE, the coded position information τ_j^m and the features p_j^m of the point are sent to the attentive pooling module by combining each other. In order to learn the unique attention score in each feature of the points $\left\{p_j^1, \ldots, p_j^m, \ldots .p_j^K\right\}$, the $\hbar$ function, which consists of shared MLPs and SoftMax activation function, is used. The unique attention score ζ_j^m is

$$\zeta_j^m = \hbar\left(p_j^m, W\right) \quad (39)$$

where $\hbar$ represents the function composed of shared MLPs, and W represents the learnable weight parameter for MLPs. The weighted sum of the features $\tilde{p}_j$ is

$$\tilde{p}_j = \sum_{m=1}^{K}\left(p_j^m \cdot \zeta_j^m\right) \quad (40)$$

In the dilated residual block, which is the last step of the LFA module, several LocSE and attentive pooling layers are combined to increase the receptive field. 1 million points can be processed approximately 200 times faster than existing methods with RandLA-Net. Benchmark tests have been done on large datasets such as Semantic3D and KITTI. As a result, the method surpasses most state-of-art approaches for large datasets. In addition, the ability to process millions of points in a short time makes RandLA-Net a candidate for real-time applications.

12) PointASNL

PointASNL method developed in 2020, basically consists of two main modules, adaptive sampling (AS) module that regulates the properties and locations of sampled data with Farthest Point Sampling (FPS), and local-non-local (L-NL) module that can extract both the local properties of the sampled points and their relationship with distant points [80]. The general structure of the method is given in figure 12. In a set abstraction level, AS module is applied to the point cloud data sampled by FPS. AS consists of four stages. If the neighbourhoods of the sampled points with k-NN are defined as $\psi\left(p_j\right) = \left\{p_j^1, \ldots, p_j^m, \ldots., p_j^K\right\}$, where m denotes any number of the set $\{1, \ldots, m, \ldots .K\}$, the properties of corresponding neighbourhoods are defined as

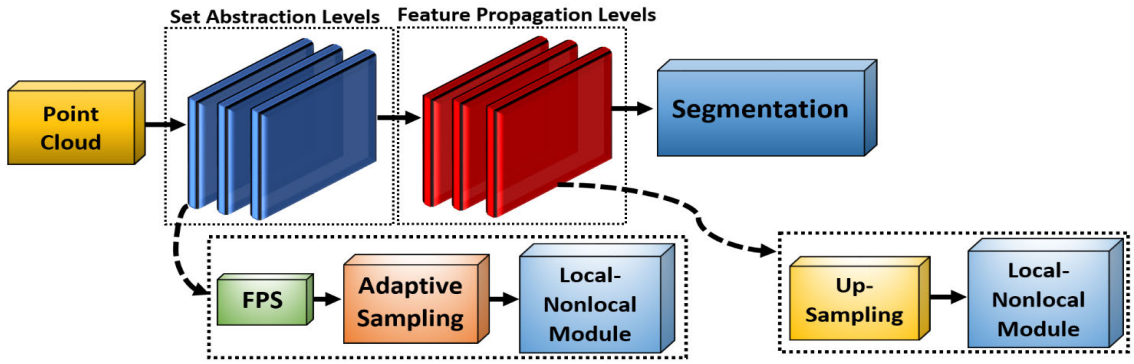

FIGURE 12. PointASNL structure.

$\lambda\left(s_j\right) = \left\{s_j^1, \ldots, s_j^m, \ldots., s_j^K\right\}$. Thus, in the first stage, $\psi\left(p_j\right)$ and $\lambda\left(s_j\right)$ are determined. Then the property s_j^k of group member p_j^k is updated and updated s_j^k is

$$\begin{aligned} s_j^k &= \delta\left(\hbar\left(p_j^k, p_j^m\right) \dashv\!\lambda\left(p_j^m\right)\right) \\ s.t.\hbar\left(p_j^k, p_j^m\right) &= \zeta\left(\gamma\left(s_j^k\right)^T \varphi\left(s_j^m\right) \Big/ \sqrt{\tilde{d}}\right) \end{aligned} \tag{41}$$

where δ represents the summation, $\hbar$ represents the function of extraction high-level features among $\left(p_j^k, p_j^m\right)$ group members, $\dashv\!\lambda$ represents the function of changing the size of the group feature, ζ represents the SoftMax activation function, d represents the dimension in which the feature has been changed, γ and φ are the 1D convolution functions. After the features are updated, the point MLPs functions v_p and v_s are used together with the SoftMax activation function to calculate the density of each point in the group using the MLPs. Thus, the MLP outputs O_p and O_s are

$$\begin{aligned} O_p &= \left\{v_p\left(s_j^k\right)\right\}_{k=1}^K, \mathrm{W}_p = \zeta\left(O_p\right), \\ O_s &= \left\{v_s\left(s_j^k\right)\right\}_{k=1}^K, \mathrm{W}_s = \zeta\left(O_s\right), \end{aligned} \tag{42}$$

where W_p and W_s denote the normalized weights after the application of the SoftMax activation function. As the last step of the AS module, adaptive shifting is applied on the coordinate information of K nearest neighbourhoods of the points, Z and corresponding properties O, to obtain new coordinates z_j' and new features o_j'. Thus, z_j' and o_j',

$$\begin{aligned} z_j' &= W_p^T Z, Z = \left\{z_j^k\right\}_{k=1}^K, \\ o_j' &= W_s^T O, O = \left\{o_j^k\right\}_{k=1}^K. \end{aligned} \tag{43}$$

are obtained. In order to improve the learning process of the local and global properties of the points after the sampling process, the local-non local (L-NL) module consisting of point local (PL) and point nonlocal (PNL) cells is applied to the points. While PL cell can be different point cloud processing algorithm such as PointNet++, PNL is the structure that deals with correlation between points. The updated feature s_j^ℓ of the point located at the center of the local region is obtained by using PL. s_j^ℓ is

$$s_j^\ell = \delta\left(\mu\left(s_i\right), \forall p_i \in \psi\left(p_j\right)\right) \tag{44}$$

where μ represents the local feature transformation function. If PL is taken as PointNet++, let 0δ is max-pooling and $\hbar$ represents MLPs. After local features are detected, global features are determined by PNL cell. Accordingly, $PNL\left(p_j, N_k\right)$ is

$$PNL\left(p_j, N_k\right) = \delta\left(\hbar\left(p_j, p_m\right) \dashv\!\lambda\left(p_m\right)\right) \tag{45}$$

where N_k represents point cloud data. Global and local features are combined with a nonlinear convolution function. Performance tests of PointASNL have been performed with real data on large datasets such as semanticKITTI [81] and S3DIS. As a result, PointASNL is the resistant method against noise and occlusion of data.

13) RPNet

The RPNet method has been developed to learn both low-level relationships (geometric) and high-level relationships (semantic) between the points [82]. The method basically uses the group relation aggregator (GRA) module. Let u and $x\left(u\right)$ are the point cloud data and the corresponding coordinates, respectively. In this case, output of the GRA module $g\left(u_m\right)$ is

$$\begin{aligned} g\left(u_m\right) &= \dashv\!\lambda\left(\lambda\left(\left\{\hbar\left(u_m, u_{mn}\right), \forall u_{mn} \in G\left(u_m\right)\right\}\right)\right) \\ where\ \hbar\left(u_m, u_{mn}\right) &= \varphi\left(R\left(u_m, u_{mn}\right)\right) \otimes \vartheta\left(u_{mn}\right) \end{aligned} \tag{46}$$

In the equation, $\dashv\!\lambda$ is linear function, λ is aggregation function, $\hbar\left(.\right)$ is inner-group relation function, $G\left(.\right)$ is grouping operator, $\varphi\left(.\right)$ is mapping function, $R\left(.\right)$ is relation function, $\otimes$ and $\vartheta\left(.\right)$ represent cross-channel attention and other linear functions. The MLP layers used for the mapping function are connected to each other with the ReLU nonlinear function. The relation function $R\left(.\right)$ is

$$\begin{aligned} R\left(u_m, u_{mn}\right) &= \left[\mu\left(a\left(u_m, u_{mn}\right)\right), \beta\left(u_m, u_{mn}\right)\right] \\ s.t.a\left(u_m, u_{mn}\right) &= \left[\left\|x\left(u_m\right) - x\left(u_{mn}\right)\right\|, x\left(u_m\right)\right. \\ &\qquad \left. x\left(u_{mn}\right), x\left(u_m\right) - x\left(u_{mn}\right)\right] \end{aligned} \tag{47}$$

where $\mu(\cdot)$ represents mapping function, $a(\cdot,\cdot)$ represents geometric function and $\beta(\cdot,\cdot)$ represents semantic function. While the RPNet-W model was developed for classification, the RPNet-D model was developed for semantic segmentation. While ModelNet40 dataset is used for classification task, S3DIS and ScanNet datasets are used for semantic segmentation task. As a result of the benchmark, RPNet is resistant to rigid transformations and noise.

14) PointTransformer

Inspired by the success of Self Attention networks on 2D images, transformers and self-attention-based point transformer method have been developed for classifying, interpreting and part interpretation tasks of point cloud objects. The point transformer layer uses vector self-attention [83]. If $\Psi = \{\psi\}$ is the feature set of the point cloud data, the output features o_m are obtained using the point transformer layer on the point cloud data. Thus, o_m is

$$\begin{aligned} o_m &= \sum_{\psi_n \in \Psi(n)} \lambda\left(\mu\left(A+\chi\right)\right) \odot \left(\varphi\left(\psi_n\right)+\chi\right) \\ s.t\ A &= \vartheta\left(\psi_m\right) - \gamma\left(\psi_n\right) \end{aligned} \tag{48}$$

where λ is a normalization function such as softmax, μ is the mapping function (attention vector), χ is the position encoder MLP, $\odot$ is the hadamard product, and φ, ϑ and γ are linear functions that perform point feature transformations. The mapping function, μ, consists of two linear layers and one nonlinear (ReLU) layer. The position encoding function, χ, is

$$\chi = \phi\left(x_m - x_n\right) \tag{49}$$

where ϕ and (x_m, x_n) represent MLP function and the coordinates of m and n points, respectively. As a result of the tests performed on the S3DIS dataset, the method exhibits a state of art performance. In comparisons made in terms of semantic segmentation task, PointTransformer method gives better results than the previous methods. In addition, PointTransformer is an advanced method for understanding large-scale 3D point cloud data through position encoding.

15) SCF-Net

The SCF-Net method was developed using the SCF module, which can learn spatial contextual properties, to partition large-scale point cloud data [84]. The SCF module, which is developed with 3 main blocks: local polar representation (LPR), dual distance attentive pooling (DDAP), and global contextual feature (GCF), can also be integrated into different point cloud understanding methods. After the creation of the first local representation in the LPR block, the invariance of the local representation with respect to the z-axis is ensured by calculating the local directions. For point u_m, whose coordinates are given by $\left(\alpha_m^k, \beta_m^k, \theta_m^k\right)$, the local notation $\left(d_m^k, \varphi_m^k, \vartheta_m^k\right)$ is

$$d_m^k = \sqrt{\alpha_m^{k2} + \beta_m^{k2} + \theta_m^{k2}} \tag{50}$$

$$\varphi_m^k = \arctan\left(\beta_m^k \Big/ \alpha_m^k\right) \tag{51}$$

$$\vartheta_m^k = \arctan\left(\theta_m^k \Big/ \sqrt{\alpha_m^k + \beta_m^k}\right) \tag{52}$$

After the first local representation is formed, the local directions are determined by calculating the point at the center of mass among the neighbouring points in the local region. The direction of a point u_m to a neighbouring point u_m^j is expressed as the local direction. During calculation, φ_m^k and ϑ_m^k are

$$\varphi_m^{k'} = \varphi_m^k - a_m \tag{53}$$

$$\vartheta_m^{k'} = \vartheta_m^k - b_m \tag{54}$$

where a_m and b_m represent the relative angles of the point u_m^j. Another block that forms the basic structure of SCF like LPR is DDAP. The DDAP block, which has three inputs as geometric distance, point features and geometric pattern, enables learning local contextual features by using neighboring point features. Let $\ell_{(m,g)}^k$ and $\ell_{(m,f)}^k$ be the geometric distance and feature distance, respectively. Feature distance $\ell_{(m,f)}^k$ is

$$\ell_{(m,f)}^k = \text{mean}\left(\left|\psi_m - \psi_k\right|\right) \tag{55}$$

where $mean(\cdot)$ denotes mean function, $|\cdot|$ denotes the L1 norm, ψ_m denotes the feature vectors of the m*th* point and ψ_k denotes the feature vectors of the k*th* neighbor point. The double distance, ℓ_m^k,

$$\ell_m^k = \exp\left(-\ell_{(m,g)}^k\right) \oplus \lambda \exp\left(-\ell_{(m,f)}^k\right) \tag{56}$$

where $\oplus$ and λ represent the combining and editing operators, respectively, is calculated. Finally, the local contextual properties, $\Psi_{(m,L)}$, is

$$\begin{aligned} \Psi_{(m,L)} &= \sum_k^K \left(p_m^k \cdot \Psi_m\right) \\ \text{s.t. } p_m^k &= softmax\left(MLP\left(\ell_m^{k+}\right)\right) \end{aligned} \tag{57}$$

where p_m^k denotes the properties of the points, Ψ_m denotes the attentive pooling weight and ℓ_m^{k+} represents the combination of the double distance and the point property. In the GCF block (last block), the global properties $\Psi_{(m,G)}$ are

$$\begin{aligned} \Psi_{(m,G)} &= MLP\left(\left(\alpha_m, \beta_m, \theta_m\right) \oplus c_m\right) \\ \text{s.t. } c_m &= \mu_m \big/ \mu_g \end{aligned} \tag{58}$$

where c_m is the volume ratio, μ_m is the volume of the sphere bounding the point neighbourhood and μ_g is the volume of the sphere bounding the point cloud data. SCF-Net was developed by designing the SCF module, random sampling, up sampling, MLP and FC layers. Cross-entropy loss function is used while training data with SCF-Net. Tests were conducted on two large datasets, S3DIS and Semantic3D, for the segmentation task. As a result of the tests, there are many different scenarios where SCF-Net gives better results than the previous methods.

16) BAAF-Net

The bilateral augmentation and adaptive fusion network (BAAF-Net) has been developed to accurately segment point cloud data obtained from real environments using bilateral context and adaptive fusion modules [85]. On the one hand, BAAF-Net aims to eliminate the ambiguity between points close to each other by increasing the local contextual features between the points in the bilateral structure, on the other hand, to combine the points with different resolutions with adaptive fusion to make an accurate segmentation. The bilateral context module increases local contextual features by using geometric and semantic features together. Let $\chi = \{x\}$ and $\Psi = \{\psi\}$ be the geometric properties of the point cloud data and the semantic properties, respectively. In this case, the local geometric context of the point x_m in the center is defined as $\hbar(x_m) = [x_m; x_n - x_m]$ in the 3D space with respect to the point x_n in the calculated neighbourhood with k-NN, while the local semantic context in the feature space is defined as $\hbar(\psi_m) = [\psi_m; \psi_n - \psi_m]$ where ψ_n is the neighbor features. MLP is applied over the local semantic context $\hbar(\psi_m)$ to estimate the 3-DoF (Degree of Freedom) bilateral offsets for neighbouring points x_n and the MLP yields the output, $\tilde{x}_n$,

$$\tilde{x}_n = MLP\left(\hbar\left(\psi_m\right)\right) + x_n \tag{59}$$

Similarly, equation (59) is applied to the data in the feature space and obtained output, $\tilde{\psi}_n$ is

$$\begin{aligned} \tilde{\psi}_n &= MLP\left(\tilde{\hbar}\left(x_m\right)\right) + \psi_n \\ \text{s.t. } \tilde{\hbar}\left(x_m\right) &= \left[x_m; x_n - x_m; \tilde{x}_n\right] \end{aligned} \tag{60}$$

where $\tilde{\hbar}(x_m)$ represents augmented geometric context. The augmented local meaning context for the feature can be expressed with $\tilde{\hbar}(\psi_m) = \left[\psi_m; \psi_n - \psi_m; \tilde{\psi}_n\right]$. In this case, the local context $\hbar_m$ is obtained by combining geometric and semantic features. Therefore, $\hbar_m$ is

$$\hbar_m = concat\left(MLP\left(\tilde{\hbar}\left(x_m\right)\right), MLP\left(\tilde{\hbar}\left(\psi_m\right)\right)\right) \tag{61}$$

To approximate the geometric center of the shifted neighbouring points $\tilde{x}_n$ to the local centroid in 3D space, the l2 minimization function is applied to the points and the result of the minimization, $\ell(x_m)$, is

$$\ell\left(x_m\right) = \left\| \frac{1}{k} \sum\nolimits_{n=1}^{k} \tilde{x}_n - x_m \right\|_2 \tag{62}$$

Finally, the obtained information is collected in a single feature map φ

$$\varphi_m = concat\left(\max_k\left(\hbar_m\right), \underset{k,\not{\lambda}_m}{\text{mean}}\left(\hbar_m\right)\right) \tag{63}$$

where $\not{\lambda}_m$ represents the learnable weight for k adjacent points. After extracting the feature map, φ, a comprehensive feature map φ_{out} is obtained by summing all the features at various resolutions for each point. The semantic segmentation process is completed by entering the feature map φ_{out} into the FC layer. While the FPS sampling method is used in the implementation of the bilateral context module, the cross-entropy loss function is used in the backpropagation stage. As a result of tests with S3DIS, Semantic3D, and SemanticKITTI databases, the method has a state-of-the-art level. Another advantage of BAAF-Net is that it can process real point cloud data at different resolutions.

17) PointMLP

Recently, the PointMLP method has been developed, which can accurately and quickly extract local features by directly processing point cloud data with MLP networks [86]. Instead of using a slow-running customized feature extractor to extract features from point data, the method works fast by using pure MLP networks, but also has a higher percentage of accuracy than its predecessors. If the feature set of point cloud data is represented by $\Psi = \{\psi\}$, then PointMLP's key operation (basic mathematical model) $\hbar_m$ is

$$\hbar_m = \varphi_{pos}\left(\lambda\left(\varphi_{pre}\left(\psi_{m,n}\right), |n = 1, \ldots, N\right)\right) \tag{64}$$

where φ_{pos} is the MLP block used to extract the deep aggregation features, λ is the aggregation function (max-pooling etc.), φ_{pre} is the MLP block that can learn the weights of the features in the local region and $\psi_{m,n}$ is the n-th neighbor point feature of m-th sampled point. A geometric affine module (GAM) has been developed to adapt the algorithm to the geometric diversity between the local regions to which the points are connected. The properties of neighbouring points $\{\psi_{m,n}\}$ in local areas can be transformed by using GAM and transformed properties $\{\psi_{m,n}\}$ is

$$\begin{aligned} \{\psi_{m,n}\} &= u \odot \frac{\{\psi_{m,n}\} - \psi_m}{\gamma + \eta} + v \\ s.t.\ \gamma &= \sqrt{\frac{1}{s} \sum_{n}^{s} \left(\psi_{m,n} - \psi_m\right)^2} \end{aligned} \tag{65}$$

where u and v denote the learning parameters, while $\odot$ and η denote the Hadamard product and a very small value that provides numerical stability, respectively. Equations (64) and (65) give a single step of PointMLP. A deep learning structure is created by repeating these steps n times. The MLP layers φ_{pre} and φ_{pos} given in Equation (64) are interconnected by the ReLU activation function. As a result of tests using ModelNet40 and ScanObjectNN datasets, PointMLP gives the best performance for object classification. At the same time, PointMLP has a high working speed. Since 3D point cloud data is irregular, 3D convolution design is quite difficult and complex compared to convolution designed for 2D images. 3D convolution design can be examined in two categories, continuous and discrete.

18) POINTWISE CNN

The pointwise CNN method has been developed for semantic segmentation and recognition tasks of point cloud data using point wise convolution applied to each point [87]. Output of

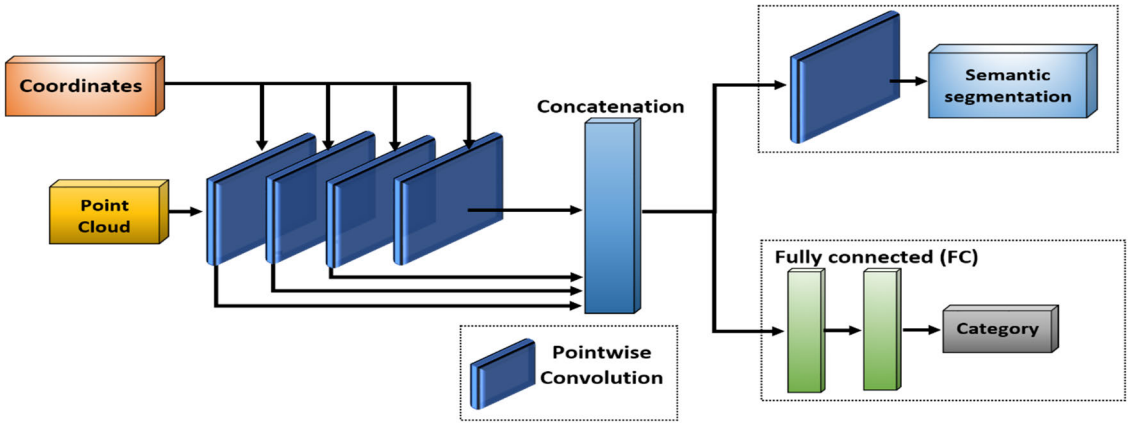

FIGURE 13. Pointwise CNN.

the Pointwise convolution v_m^u, is

$$v_m^u = \sum_s w_s \frac{1}{|\vartheta_m(s)|} \sum_{\rho_n \in \vartheta_m(s)} v_n^{u-1}, \tag{66}$$

where s is the number of iterations, w_s is the kernel weight in the s th subfield, $|.|$ is the operator that counts the points in the subfield, $\vartheta_m(s)$ is the s th subfield of the kernel in the center of the m point, ρ_n is the coordinates of the n point, v_m is the value at the m point, v_n is the value at the n point, $u-1$ and u denote the input and output layers, respectively. Gradients must be calculated according to the input data and kernel weights to make pointwise convolution trainable. The gradient equation of the m points located in the neighbourhood of the entry points n, $\partial\ell/\partial v_n^{u-1}$ is

$$\frac{\partial \ell}{\partial v_n^{u-1}} = \sum_{m \in \vartheta_m} \frac{\partial \ell}{\partial v_m^u} \frac{\partial v_m^u}{\partial v_n^{u-1}},$$
$$\text{where } \frac{\partial v_m^u}{\partial v_n^{u-1}} = \sum_s w_s \frac{1}{|\vartheta_m(s)|} \sum_{\rho_n \in \vartheta_m(s)} 1, \tag{67}$$

where ℓ represents the loss function. According to the chain rule, $\partial\ell/\partial v_m^u$ represents the gradient up to the known layer u during backpropagation. Similarly, the gradient at m points according to the kernel weights, $\partial\ell/\partial w_s$, is

$$\frac{\partial \ell}{\partial w_s} = \sum_m \frac{\partial \ell}{\partial v_m^u} \frac{\partial v_m^u}{\partial w_s},$$
$$\text{where } \frac{\partial v_m^u}{\partial w_s} = \frac{1}{|\vartheta_m(s)|} \sum_{\rho_n \in \vartheta_m(s)} \partial v_n^{u-1}, \tag{68}$$

Unlike the convolution used in the volumetric approach, pointwise convolution does not use pooling. One of the two advantages of not using pooling is that there is no need for up and down sampling, and the other is that it simplifies and speeds up the network design. The pointwise CNN structure designed for object recognition and semantic segmentation tasks is given in Figure 13. With the point wise convolution operator, the nearest neighbours of the central point are quickly queried, and then they are recorded in the kernel cells to perform the convolution operation with kernel weights. Comparison tests of the method were made with S3DIS, ModelNet40 and SceneNN datasets, and Pointwise CNN did not show a good performance compared to other methods. In addition, the training speed of the data was slow according to result of the tests.

19) PointCNN

PointCNN making it possible to apply typical convolution operators transforms the irregular point cloud data into latent and potential canonical order with X-conv transformation [88]. The x-conv operator converts the properties of the point cloud data Ψ_1 into Ψ_2. In the X-conv algorithm, in the steps using multi-layer perceptron (MLP), two FC layers are used, which are connected to each other by exponential linear unit (ELU) activation functions. Representative points are created with random subsampling for classification tasks, and farthest point sampling (FPS) for semantic segmentation tasks. The benchmark tests for the segmentation task are done with ShapeNet, S3DIS and ScanNet, while the classification is done with ModelNet40, ScanNet, TU-Berlin [89], Quick Draw [90], MNIST [91], CIFAR10 [92]. As a result of segmentation tests, PointCNN gives the best performance in terms of accuracy compared to previous methods. Similarly, as a result of classification tests for MNIST, Quick Draw and ModelNet40, PointCNN gives the best performance in terms of accuracy compared to previous methods.

20) FLEX-CONVOLUTION BASED NETWORK (FCBN)

Flex-convolution-based network (FCBN) has been developed to process data with high point density (approximately 7 million points) with low memory and hardware features at the same time [93]. The point cloud dataset, P, is

$$P = \{(x_j, \psi_j) \in X \times \Psi | j = 0, 1, \ldots, s-1\} \tag{69}$$

where x_j represents the coordinates of the point, $\psi_j$represents the feature matrix, X represents the coordinate set,

Ψ represents the feature set, and s represents the total number of points. Therefore, the feature matrix as a result of the convolution, $\psi'\left(v', x\right)$ is

$$\psi'\left(v', x\right) = \sum_{v \in V} \sum_{x' \in N_k(x_j)} \tilde{\kappa}\left(v, x, x'\right).\psi\left(v, x'\right) \tag{70}$$

where v represents the feature size of the point cloud (RGB information for this method), V is the feature size set, $\tilde{\kappa}$ is the kernel filter, k is the k nearest neighbourhood number and N is the neighbourhood set. In addition, the kernel filter in the equation can be defined as the learnable linear transformation function of the $\tilde{\kappa}$ Prewitt operator. For the sampling stage, a new sampling method, inverse density importance sub-sampling (IDISS) has been developed. Since the complexity of farthest point sampling (FPS) is $O\left(n^2\right)$, memory is overloaded. For this reason, the IDISS sampling method, which has a computational complexity of $O(n)$, has been developed, making it possible to process millions of points quickly. In the IDISS sampling method, inverse density δ can be approximated by summing the distances of all points in k neighbourhoods of a point. Therefore, δ is

$$\delta(x) = \sum_{x' \in N_k(x)} \left\| x - x' \right\| \tag{71}$$

The flex-convolution-based neural network structure designed for semantic segmentation is given in Figure 14. While FCBN was tested on the ModelNet40 dataset for the classification task, it was tested on the dataset consisting of approximately 215 million points, which was formed by scanning an area of 6000 square meters for the semantic segmentation task. Thus, one of the first methods that can process large data consisting of millions of points is the Flex-convolution-based neural network.

21) PCNN

Point convolutional neural network (PCNN) contains two main functions as extension and restriction in order to be able to segment and classify point cloud data by applying convolution [94]. Before the convolution process, the point cloud data is mapped to a volumetric structure with the extension function. After applying the standard Euclidean convolution to the volumetric data, it is sampled back to the point structure with the restriction function. The extension operator, φ_P, is

$$\begin{aligned} \varphi_P[\delta](p) &= \sum_m \delta_{mn} \not{\lambda}_m(p) \\ \text{s.t. } \not{\lambda}_m(p) &= c\vartheta_m \Psi\left(|p - p_m|\right) \end{aligned} \tag{72}$$

where δ and $\not{\lambda}$ are basis functions, c is a constant number, ϑ is the amount of shape area corresponding to p_m point and Ψ is Gaussian Radial Basis Function (GRBF). The restriction operatorϕ_P, is

$$\phi_P[\hbar] = \hbar_n(p_m), \tag{73}$$

$\hbar$ denotes the volumetric function. The max-pooling process is used for subsampling. Tests of the PCNN method were performed using ModelNet40 for classification and ShapeNet datasets for segmentation. The successful classification of point cloud data with few samples and the successful classification of data in different poses are the advantages of PCNN. In addition, CNN structures developed for images can be applied to PCNN, thanks to ability of the PCNN to transform an irregular structure such as a point cloud into a regular structure such as volumetric. However, since PCNN includes transformation operations, the computational cost and complexity are high.

22) SpiderCNN

The SpiderCNN method was developed by using the spider convolution units, which were developed by designing a filter consisting of the product of the step function and Taylor expansion that captures the geometric properties of the point cloud data [95]. SpiderCNN is designed for segmentation and classification tasks. Let $U = \{u_1, u_2, \ldots, u_n\}$ be the point cloud dataset, Ψ is the function defined on the point cloud data, f is the convolution filter defined in an area centered at the center radius r. Therefore, spider convolution, $\Psi * f(u)$ is

$$\Psi * f(u) = \sum_{v \in U, \|v-u\| \le r} \Psi(v) f(u - v), \tag{74}$$

where v and r represent the neighbourhoods of the points u and the radius, respectively. Convolution filter selection is made from the $\left\{f_\varphi\right\}$ parameter family. The filter design, which can better extract the geometric properties of the point cloud used when choosing from the filter family, $f_\varphi(x, y, z)$, is

$$f_\varphi(x, y, z) = f_{\varphi^{St}}^{Step}(x, y, z)\, f_{\varphi^{Ty}}^{Taylor}(x, y, z), \tag{75}$$

where $f_\varphi(x, y, z)$ is the designed filter and (x, y, z) represents the coordinate information. φ, on the other hand, consists of the combination of two vectors φ_j^{St} and φ_j^{Ty}. Among the filters, $f_{\varphi^{St}}^{Step}(x, y, z)$, is

$$f_{\varphi^{St}}^{Step}(x, y, z) = \varphi_j^{St} \text{ if } r_j \le \sqrt{x^2 + y^2 + z^2} < r_{j+1}, \tag{76}$$

while $f_{\varphi^{Ty}}^{Taylor}(x, y, z)$ is

$$\begin{aligned} f_{\varphi^{Ty}}^{Taylor}(x, y, z) = {} & \varphi_0^{Ty} \\ & + \varphi_1^{Ty} x + \varphi_2^{Ty} y + \varphi_3^{Ty} z \\ & + \varphi_4^{Ty} xy + \varphi_5^{Ty} yz + \varphi_6^{Ty} xz + \varphi_7^{Ty} x^2 \\ & + \varphi_8^{Ty} y^2 + \varphi_9^{Ty} z^2 + \varphi_{10}^{Ty} xy^2 + \varphi_{11}^{Ty} x^2 y \\ & + \varphi_{12}^{Ty} y^2 z + \varphi_{13}^{Ty} yz^2 + \varphi_{14}^{Ty} x^2 z + \varphi_{15}^{Ty} xz^2 \\ & + \varphi_{16}^{Ty} xyz + \varphi_{17}^{Ty} x^3 + \varphi_{18}^{Ty} y^3 + \varphi_{19}^{Ty} z^3. \end{aligned} \tag{77}$$

The SpiderCNN structure designed for classification and segmentation tasks is given in Figure 15. In the benchmark made with the SHREC15 dataset, which includes objects with more complex geometric properties than ModelNet40 for classification, spiderCNN not only outperforms its predecessors in

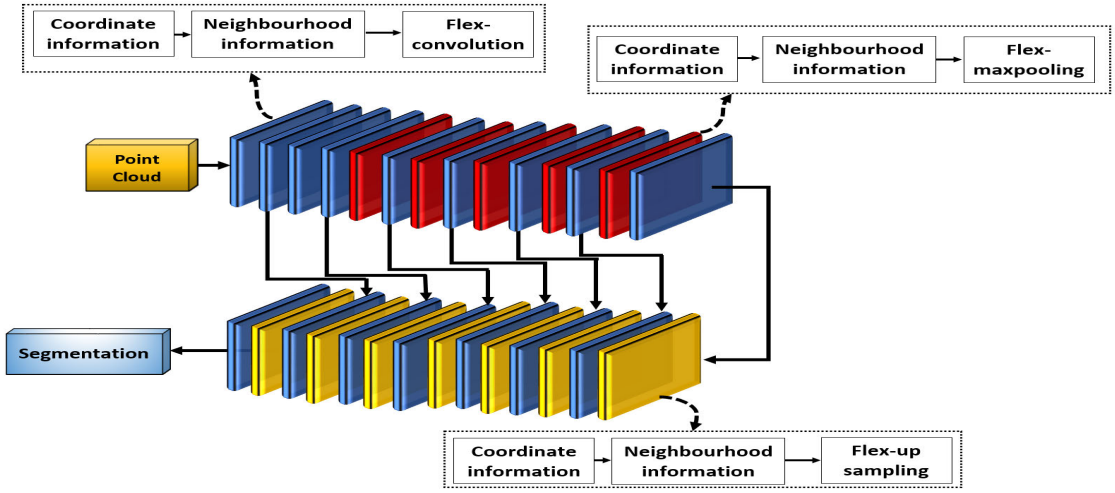

FIGURE 14. Flex-convolution based neural network for semantic segmentation.

terms of accuracy, but also gives good results even when the number of points is low. Even when the number of points is 32, the accuracy rate of SpiderCNN in terms of classification is 87.7%.

23) MONTE CARLO CONVOLUTION NEURAL NETWORK

The Monte Carlo Convolution Neural Network (MCCNN) method uses a Monte Carlo estimation-based convolution structure using a probability density function (PDF) [96]. Monte Carlo Convolution (MCC), a method developed for non-uniformly sampled points, is also applicable to different sampling densities and methods. For a point a, in the point cloud, the MCC $(u^* v)(a)$ is

$$(u * v)(a) \approx \frac{1}{|N(a)|} \sum_{i \in N(a)} \frac{u(q_i)\, v\left(\frac{a-q_i}{r}\right)}{p(q_i|a)} \tag{78}$$

where u represents the convolution function to be applied, v represents the convolution kernel, $N(a)$ represents the set of neighbourhoods of the sphere with radius r, and $p(q_i|a)$ represents the probability density function (PDF) at point q_i while point a is fixed. PDF, $p(q_i|a)$, is

$$p(q_i|a) \approx \frac{1}{|N(a)|\, \delta^3} \sum_{m \in N(a)} \left\{ \prod_{c=1}^{3} \hbar\left(\frac{q_{i,c} - q_{m,i}}{\delta}\right) \right\} \tag{79}$$

where δ denotes the bandwidth that adjusts the smoothness of the result obtained from the sample density function, $\hbar$ denotes the density prediction kernel, and c denotes one of the 3 dimensions. In addition, MCCN uses Poisson Disk Sampling (PDS), which is scalable instead of farthest point sampling and can preserve the sampling pattern while playing with the number of samples. The fact that PDS is scalable and can limit the number of samples makes it the reason for preference for MCCNN method. While MCCN performed at an average value in the benchmark for classification, it succeeded in getting ahead of many state-of-art methods in the benchmark for segmentation. Applicable for real-world tasks, MCNN has proven to be a method that can work well with different sampling densities, by significantly reducing the GPU (Graphical Processing Unit) memory required for training.

24) POINTCONV BASED NEURAL NETWORK

Another method, which can be considered as a different extension of the Monte Carlo approach, is the PointConv-based neural network structure [97]. PointConv aims to significantly enlarge the mesh by calculating weight functions effectively. PointConv considers convolution kernels as a non-linear function of local points made up of weights and density functions. While weight functions are learned with MLPs, density functions are learned with kernel density estimation. PointConv is mathematically defined as

$$\begin{aligned} \text{PointConv}(X, G, Q)_{xyz} = & \sum_{(\varphi_x, \varphi_y, \varphi_z) \in C} \delta \lambda \psi \\ & where\ \delta = X(\varphi_x, \varphi_y, \varphi_z),\ \lambda = G(\varphi_x, \varphi_y, \varphi_z) \\ & and\ \psi = Q(x+\varphi_x, y+\varphi_y, z+\varphi_z) \end{aligned} \tag{80}$$

where $(\varphi_x, \varphi_y, \varphi_z)$ is any coordinates in local region C, $X(\varphi_x, \varphi_y, \varphi_z)$ is inverse density in $(\varphi_x, \varphi_y, \varphi_z)$, $G(\varphi_x, \varphi_y, \varphi_z)$ is weight function, $Q(x+\varphi_x, y+\varphi_y, z+\varphi_z)$ is a feature of a point in local region C, and (x, y, z) denotes coordinate information in 3D plane. Not only the convolution operation, which reduces the resolution of the points, but also the deconvolution operation, which converts the points to their former resolution, is performed with PointConv. The PointConv has passed many methods in terms of accuracy in benchmarks for segmentation and classification.

25) Geo-CNN

The basis of the Geo-CNN method, which was developed as a method that can be used in point cloud data classification and detection tasks, is the generic convolution-like operation

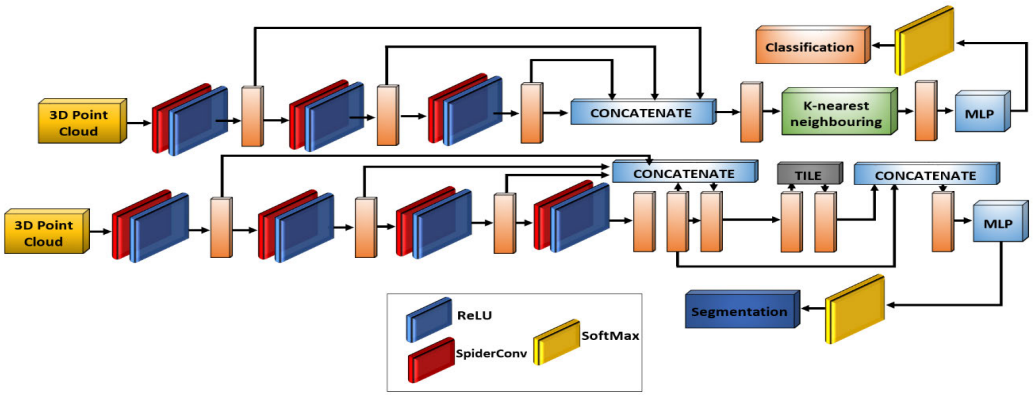

FIGURE 15. SpiderCNN structure designed for segmentation and classification.

(geo-conv) algorithm that models the geometric properties of the points taken from the local region [98]. The geometric relationship between the points is based on the extraction of the edge features between the center point and corresponding neighbours. If the point cloud data is expressed as a and the neighbouring points with b, the mathematical model of GeoConv applied to the neighbourhood of the point a yields the feature map, $F_{\vec{a}}^{k+1}$,

$$F_{\vec{a}}^{k+1} = W_m F_{\vec{a}}^{k} + \frac{\sum_{\vec{b}\in N(\vec{a},r)} \ell\left(\vec{a},\vec{b},r\right)\delta\left(\vec{a},\vec{b}\right)}{\sum_{\vec{b}\in N(\vec{a},r)} \ell\left(\vec{a},\vec{b},r\right)}$$
$$\text{s.t. } \ell\left(\vec{a},\vec{b},r\right) = \left(r - \left\|\vec{a}-\vec{b}\right\|\right)^2 \tag{81}$$

where $F_{\vec{a}}^{k+1}$ denotes the feature map resulting from GeoConv in the $k+1$ layer, W_m is the weight matrix used to extract features from the center point, $\ell\left(\vec{a},\vec{b},r\right)$ is the distance between point a and b, $\delta\left(\vec{a},\vec{b}\right)$ denotes the function that models the edge properties and $N\left(\vec{a},r\right)$ represents the neighbourhood set of a at radius r. In addition, after multiplying the weight matrix with the feature map of the point in order to detect edge features in all directions, the features are collected with the function $\delta\left(\vec{a},\vec{b}\right)$ is

$$\delta\left(\vec{a},\vec{b}\right) = \sum_{\vec{x}\in X_{\vec{b}}} \cos^2\left(\varphi_{\vec{ab},\vec{x}}\right) W_{\vec{x}} F_{\vec{a}}^{k} \tag{82}$$

where $\varphi_{\vec{ab},\vec{x}}$ denotes the angle between $\overrightarrow{ab}$ and $\vec{x}$, $\vec{x}$ represents the orthogonal basis and $\cos^2(.)$ is used to collect features. First of all, the edge features between the central point in the local region of an object and the points in its neighbourhood are extracted using vector decomposition on three orthogonal bases. Then, these orthogonal foundations form the edge feature by gathering under a single feature according to their angles. Thus, learning the geometric properties between points is provided. Tests for GeoCNN were conducted on KITTI and ModelNet40. In classification tests with ModelNet40, GeoCNN surpasses other methods except its predecessor VRN Ensemble in terms of accuracy.

26) LP-3DCNN

Kumawat et al. developed the Local Phase in 3D CNN (LP-3DCNN) method, which includes a Rectified Local Phase Volume (ReLPV) block, in order to reduce the computational and memory space costs of 3D CNN structures and to improve the learning properties [99]. The ReLPV block obtains phase information by calculating the 3D Short Time Fourier Transform (STFT) at fixed low-frequency points in a local region of the object. ReLPV block structure is given in Figure 16. In the first layer of ReLPV, a reduced size feature map $u\,(p)$ where p represents the location, is obtained at the output by applying classical 3D convolution to the feature map extracted from the point cloud data. In the second layer, local phase information is obtained by using STFT with $u\,(p)$ from the previous layer. What makes the local phase important is that the edge and contour information in the images are more prominent in the phase. By calculating the 3D STFT on the neighbouring point set,$N\,(p)$, the local phase information, $U\,(f,p)$ is

$$U\,(f,p) = \sum_{q\in N(p)} u\,(p-q)\exp^{-j2\pi f^T q} \tag{83}$$

where f and q represent the frequency value and the point adjacent to p, respectively, while $j=\sqrt{-1}$. The LP-3DCNN method is able to learn complex values by applying the nonlinear activation function in the third layer to the local phase information. In the last layer, the feature map of the local phase is extracted again with a classical 3D convolution. The LP-3DNN structure is formed by using the ReLPV block many times with average-pooling. In the ModelNet benchmark for classification comparison, apart from a volumetric-based approach, the LP-3DNN is superior to other methods compared. The LP-3DCNN achieves a state of art level by using only 11% of the parameters used by current state of art methods. In addition, the method is at a state of art level in

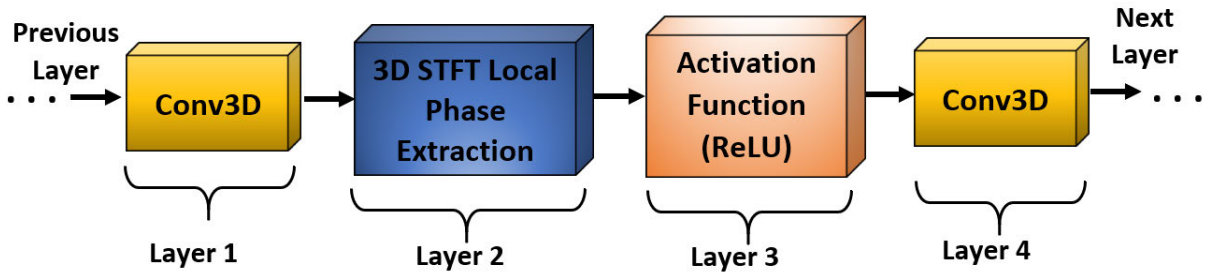

FIGURE 16. ReLPV system block structure.

terms of a different application type on the UCF-101 split-1 action recognition dataset.

27) ANNULARLY CONVOLUTIONAL NEURAL NETWORKS

Annularly Convolutional Neural Networks (A-CNN) has been developed using the annular convolution method in order to better detect the geometric properties of each point in local neighbourhood [100]. Ring structures, constrained based k-NN search, neighbour sorting, annular convolution, and pooling on rings are essential components for A-CNN. By using ring structures, A-CNN solves the problem of poor computational performance in the PointNet++ method, as there are overlaps between regions of different scales due to overlapping of neighbouring points. The points within each ring are processed separately by dividing point cloud data into regular or enlarged rings. Thus, since it will not be possible to see a point in different regions twice, the ring structure contributes to the finding of neighbouring points with k-NN. In addition, there is no need to increase the convolution parameters in large regions, since larger rings of the same kernel size are also convolved. In the stage of finding the nearest points, unlike the ball query algorithm used in PointNet++, finding the nearest neighbour points is guaranteed by using constrained based k-NN search and Euclidean metric in A-CNN. The neighbour sorting stage is divided into two sub-layers as reflection and sorting. For reflection and sorting, it is necessary to make a normal estimation on the point first and to calculate the eigenvector and eigenvalue values of the covariance matrix X in order to find the normal estimation. The Covariance matrix, X, is

$$X = \frac{1}{\kappa}\sum_{n=1}^{\kappa}(v_n - u_m)\,.\,(v_n - u_m)^T\,,$$
$$X.\phi_j = \varphi_j.\phi_j, j \in \{0, 1, 2\}\,, \tag{84}$$

where κ is the number of neighbouring points, v_n is points, u_m is neighbouring points, ϕ_j and φ_j denote the eigenvector of the covariance matrix and the j-th eigenvalue, respectively. Eigenvector ϕ_0 is the predicted normal vector η_m. Using the normal estimation, η_m, the orthogonal projection of neighbouring points, ψ_n, is

$$\psi_n = v_n - ((v_n - u_m)\,.\eta_m)\,.\eta_m, n \in \{1, \ldots, \kappa\} \tag{85}$$

In order to be able to sort after the reflection, firstly, the angle between the starting point s and the vector$(\psi_n - u_m)$, which is randomly selected from the point cloud data set,cos $\left(\theta_{\psi_n}\right)$, is

$$\cos\left(\theta_{\psi_n}\right) = \frac{s.\,(\psi_n - u_m)}{\|s\|\,\|\psi_n - u_m\|} \tag{86}$$

After calculating the angle values, the semicircle of the selected point is decided by using signature, $sign_{\psi_n}$,

$$sign_{\psi_n} = (s \times (\psi_n - u_m))\,.\eta_m \tag{87}$$

In equation (87), if $sign_{\psi_n} \geq 0$ then $\theta_{\psi_n} \in [0°, 180°]$ else if $sign_{\psi_n} < 0$ then $\theta_{\psi_n} \in (180°, 360°)$. According to the recalculated cosine values, λ_{ψ_n} is

$$\lambda_{\psi_n} = \begin{cases} -\cos\left(\theta_{\psi_n}\right) - 2 & sign_{\psi_n} < 0 \\ \cos\left(\theta_{\psi_n}\right) & sign_{\psi_n} \geq 0 \end{cases} \tag{88}$$

The points are sorted in descending order of λ_{ψ_n}. Geometric properties are obtained by applying annular convolution and max-pooling operations on the listed points, respectively. Just before the Constrained based k-NN algorithm, farthest point sampling (FPS) is used. When FPS, constrained based k-NN, annular convolution and max-pooling are applied sequentially, a set of abstraction layers is obtained. While the classification task is completed by using a set of abstraction layers and ReLU activation function together, the same layers used in classification for segmentation are used in the encoder part, and the interpolation method based on the inverse squared Euclidean distance-weighted average of the three nearest neighbours is used in the decoder part. In benchmarks made with standard data sets, A-CNN surpasses many methods in the comparison of different parameters.

28) KERNEL POINT CONVOLUTION

Thomas et al. have developed a Kernel Point Convolution (KPConv)-based network that performs point classification and segmentation tasks with a PCNN-like approach [101]. The convolution weights of KPConv are determined according to the distances of the kernel points that can be used in flexible numbers in Euclidean space. KPConv, which can be applied to points of different densities with subsampling, also has an adaptive structure. While deformable KPConv is used for complex tasks, rigid KPConv can be used for

simpler tasks. If the point cloud data set is expressed with $U = \{u_1, \ldots., u_n\}$, the corresponding feature set is expressed with $V = \{v_1, \ldots., v_n\}$. Before defining the kernel function κ in the KPConv algorithm, remember that the standard point convolution,$V(u) * \kappa(u)$, is

$$V(u) * \kappa(u) = \sum_{u_j \in N_u} \kappa\left(u_j - u\right) v_j \tag{89}$$

where N_u denotes represents the set of neighbours of the point u and let $x_j = \left(u_j - u\right)$ for the sake of simplicity. In this case, the kernel function defined on $\kappa\left(x_j\right)$ is

$$\kappa\left(x_j\right) = \sum_{s<S} \delta\left(x_j, \tilde{u}_s\right) W_s$$
$$\text{s.t. } \delta\left(x_j, \tilde{u}_s\right) = \max\left(0, 1 - \frac{\left\|x_j - \tilde{u}_s\right\|}{\lambda}\right) \tag{90}$$

where δ is linear correlation, $\tilde{u}_s$ is kernel points, W_s and λ kernel weights and impact distance of kernel points, respectively. Instead of rigid convolution described above, deformable convolution is applied for data with dense point cloud. Thus, deformable convolution $V(u) * \kappa(u)$ is

$$V(u)^* \kappa(u) = \sum_{u_j \in N_u} \kappa_d\left(u - u_j, \Delta(u)\right) v_j$$
$$\text{s.t. } \kappa_d\left(x_j, \Delta(u)\right) = \sum_{s<S} \delta\left(x_j, \tilde{u}_s + \Delta_s(u)\right) W_s \tag{91}$$

where κ_d and $\Delta(u)$ represent the deformable convolution function and the shift S produced for each convolution position, respectively. In addition, while the fitting loss function is used to regulate the distance between the kernel points and the nearest neighbouring input points, the repulsive loss function is used as a regulator for all unmatched kernel points. 5-layer neural network with KPConv in each layer is designed for the classification task. Except for the last layer where SoftMax activation function is used, each layer is connected to each other with Leaky ReLU. Average pooling is used just before classification with SoftMax. While the encoding part of the segmentation task is the same as the classification, deconvolution is performed with the closest upsampling method in the decoder part. Benchmark of the method was made with 4 datasets: Scannet, Sem3D, S3DIS and PL3D [102]. KPConv can be used in different applications in the CNN structure as well as giving good results in the field of classification and segmentation.

29) RELATION-SHAPE CONVOLUTIONAL NEURAL NETWORK

Liu et al. developed the Relation-Shape Convolutional Neural Network (RS-CNN) method, which can extract the geometric topology of local points for classification and segmentation tasks using Relation-Shape convolution (RS-Conv) [103]. RS-Conv is a convolution that can learn a high-level relationship based on predefined geometric priorities using several MLP layers. As a result of RS-Conv, feature vector of local point subset, $\psi_{X_{sub}}$, is

$$\psi_{X_{sub}} = \delta\left(G\left(\left\{\hbar\left(u_{mn}\right).\psi_{p_n}\right\}\right)\right) \tag{92}$$

where δ is activation function (ReLU), G is summing function (max-pooling), $\hbar$ is the mapping function (shared MLPs) that summarizes the high-level relationship between two points, u_{mn} is the low-level relationship vector between the p_m and p_n points, ψ is feature vector, ψ_{p_n} is feature vector of neighbouring points p_n around the central point p_m, and X_{sub} is local point subset. In Figure 17, RS-CNN structures designed for classification and segmentation tasks are given. RS-CNN, which exhibits state of art performance with the benchmarks, gives more accurate results compared to other methods even in cases where the point density is very low. In addition, the method has proven robust against rigid transformations.

30) SPHERICAL FRACTAL CONVOLUTIONAL NEURAL NETWORKS

Rao et al. developed a spherical symmetry-based Spherical Fractal Convolutional Neural Networks (SFCNN) method to recognize point cloud data [104]. Point cloud data with irregular structure are first projected into regular icosahedral lattices with aligned spherical coordinates. The method of discretization to spheres performed with equal-area sampling in SFCNN is significantly different from predecessor similar methods that used discretization using the equiangular sampling algorithm. High level model capacity is required to learn the invariance feature since inconsistencies occur for points with different rotations in equiangular sampling. Contrary to equiangular sampling, equal-area sampling allows for equal discretization even in different rotations, so its invariant feature can be learned more easily. Let $P = \{q_1, q_2, \ldots\ldots, q_S\}$ be the point cloud data set and $q_j = \left(x_j, y_j, z_j\right)$ be the coordinates of each point q_j. Let the spherical lattice (undirected graph) to which the points will be projected is expressed as $G = (V, E)$, where $V = \{v_1, \ldots, v_S\}$ and E denote a vertex set with S elements and set of corresponding edges, respectively. Thus, the points P are projected to the features $\left\{\psi_j | j = 1, 2, .., S\right\}$ on the spherical lattice with the developed method. Also, each feature ψ_j is associated with a unique vertex, v_j. Unlike methods that directly implement K-NN, the speed of k-NN in the SFCNN method can be accelerated by algorithms such as kd-tree, since SFCNN makes the points spherically discretized. Thus, processing dense point is possible for SFCNN. The convolution graph s applied to the points projected on the sphere ψ_j^{n+1} is

$$\psi_j^{n+1} = \lambda\left(\max_i\left(\lambda\left(\delta\left(\psi_j^n, \psi_i^n\right)\right)\right)\right), \tag{93}$$

where λ is the classical convolution operation, δ is the concatenation operator, n is the number of layers, max is the max-pooling operation, ψ_j^n and ψ_i^n are the feature vector of the vertices and corresponding neighbours to which the points are projected, respectively. The general structure of SFCNN for classification and segmentation tasks is given in Figure 18. SFCNN lags behind many methods in comparisons in terms of accuracy.

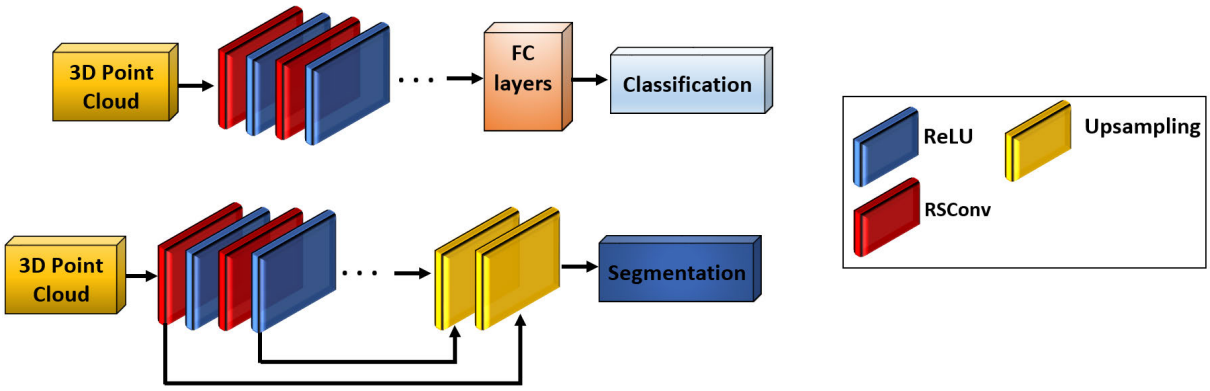

FIGURE 17. Overall frameworks of RS-CNN designed for classification and segmentation tasks.

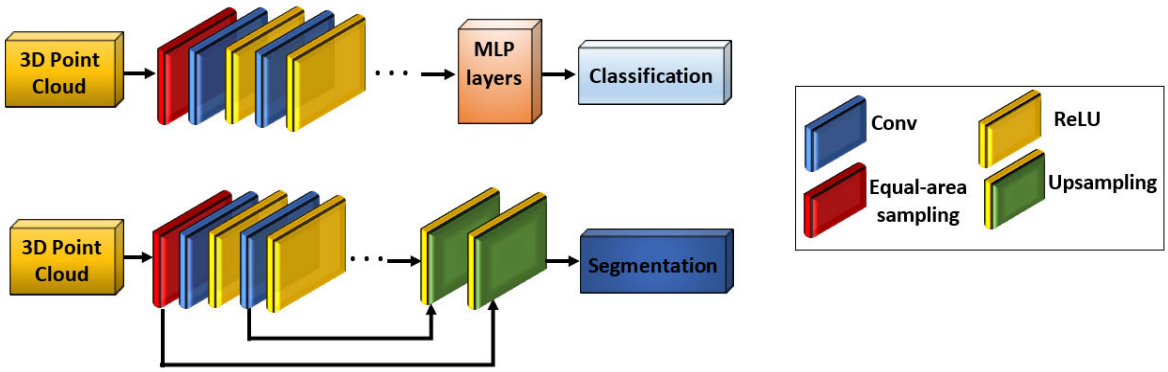

FIGURE 18. Overall frameworks of SFCNN.

31) INTERPOLATED CONVOLUTIONAL NETWORKS

Mao et al. developed the InterpConv-based Interpolated Convolutional Neural Networks (InterpCNN) method for point cloud data classification and segmentation tasks [105]. In InterpConv, discrete convolutional weights and continuous distances are used together. Thus, spatially discrete kernel weights w and interpolation function q are required. Interpolation can be done as a trilinear interpolation or a Gaussian interpolation function. The properties of irregular points are taught to InterpCNN by interpolating point features to kernel-weight coordinates with InterpConv. While classification tests are performed on ModelNet40, part segmentation operations are performed with ShapeNet Parts dataset. As a result of the tests, InterpCNN performs better in terms of accuracy against the previous methods.

32) DensePoint

The DensePoint method was developed using Single Layer Perceptrons (SLP) to learn input points contextually [106]. The DensePoint forward propagation algorithm is given in Figure 19. While the non-linear activation function δ given in the algorithm is ReLU, the aggregation function α is the max-pooling operator. The convolution used in the algorithm is done with SLP functions. Each stage is processed by taking the information of the previous step with the convolution used in the method. Thus, the recognition process can be done more accurately for the points that are made into contextual data. Tests are done with ModelNet for classification and ShapeNet for segmentation. Additionally, DensePoint has been tested for shape completion and normal estimation tasks. As a result of the comparison of the method, which was found to be resistant to noise, with other methods, the DensePoint method reached state-of-the-art performance.

33) CONTINUOUS CONVOLUTIONS

Boulch developed the ConvPoint method using the continuous convolution structure as an alternative to the methods generally developed using discrete convolution [107]. The basis of the method is the logic of continuous discrete kernels. One of the biggest advantages of the method is that it can be applied to different point sizes, while the other is that it can be applied easily just like in 2D convolution applications. As a result of the ConvPoint applied to point cloud data, μ, is

$$\mu = b + \frac{1}{|U|} \sum_{n=1}^{|U|} \sum_{m=1}^{|\kappa|} w_m u_n \delta\left(\{v_n - x\}\right) \tag{94}$$

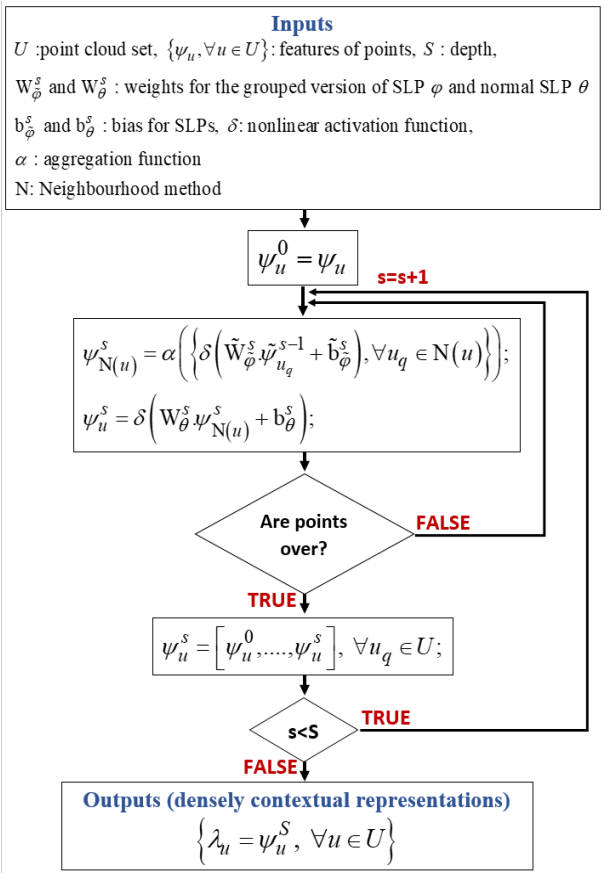

FIGURE 19. DensePoint forward pass algorithm.

where x represents the kernel points, v_n is the spatial coordinate of the nth point, δ is the geometrical weighting function, u_n is the nth point, w_m is the kernel weight, U is the point cloud set, κ is the kernel set, b and μ are the feature map obtained as bias and output, respectively. While MLP is used as the geometrical weighting function, the ReLU activation function is used when connecting the convolution layers to each other. The last layer of neural networks designed with ConvPoint is the fully connected (FC) layer. ConvPoint has been tested on large datasets such as NPM3D [102] and Semantic8 [46] for Shape classification, part segmentation and semantic segmentation tasks. ConvPoint is the state of the art in terms of accuracy and gives acceptable results even when the density of points is low.

34) CurveNet

The CurveNet model was developed in 2021 using the Curve Intervention Convolution (CIC) module, which consists of curve grouping and curve aggregation functions for classification and part segmentation tasks [108]. Learning models are developed for classification or segmentation tasks by subjecting the interconnected point arrays called curves in CIC layers to the grouping process with curve grouping function and then to the aggregation process with the curve aggregation function. CurveNet has been designed by arranging CIC modules like ResNet. If the point cloud set is expressed as $X = \{x\}$ and the point feature set $\Psi = \{\psi\}$, a curve with the length n, which is formed by the ordering of the point features, is defined as $e = \{u_1, \ldots, u_n | u \in \Psi\}$. The isomorphic graph formed by the points can be defined with $G = \{\Psi, B\}$ where B represents the k-NN calculated connection between each curve defined on the points. A parameter $\hbar(u)$ that can be learned with MLP is used to select the points on the curve. $\hbar(u)$, is

$$\hbar(u) = \sum_{1}^{k} (\text{gumbel\_soft max}(a) \cdot N_u)$$
$$\text{s.t. } a = \left\{\text{MLP}\left(s_{u^i} | u^i \in N_u\right)\right\} \tag{95}$$

where N_u represents the set of neighbouring points, and s_{u^i} and gumbel_soft max () represent the state descriptor of whose neighbourhoods are found by KNN and scoring function, respectively. The direction in which the curve will continue is determined by choosing the point with the highest score among the neighbouring points with $\hbar(u)$. After the curves are grouped, they are collected with curve aggregation to provide more variety of intra-channel features and to provide a better representation for classification. While the blocks in CurveNet are connected to each other with the Leaky-Relu function, ReLU is used in the segmentation and classification block. While ModelNet40 is used for classification tests, ShapeNetPart is used for object part segmentation tests. In addition to giving better results than previous methods in comparisons made in terms of accuracy, CurveNet also provides a plus advantage by using memory effectively.

35) FG-Net

The FG-Net method, which can rapidly process and understand large-scale LiDAR point cloud data, has been developed [109]. FG-Net basically consists of noise filtering, FG-conv and global feature extractor modules. FG-conv, which was developed to model the local features of the point cloud and obtain feature relations, includes three main modules: pointwise correlated features mining (PFM), geometric convolution modeling (GCM) and attentional aggregation (AG). By applying PFM to the point cloud data u_m, whose noise is reduced by the developed filter, the point-related features between the neighboring points u_n and u_m are extracted. The similarity score between points s_n is

$$s_n = \frac{u_n^T u_m}{\|u_n\| \, \|u_m\|} \tag{96}$$

where $\|.\|$ represents l2 norm. The similarity score vector $v_n \in \mathbb{R}^N, n = 1, 2, \ldots, N$ is generated by calculating s_n for each neighbouring point. Since the similarity score s_n cannot be associated with any task such as classification and segmentation, a new similarity score φ_n, can be adapted to deep learning training and tasks by using the attentional technique. Therefore, φ_n is

$$\varphi_n = \delta(g_1 v_n), \varphi_n \in \mathbb{R}^N \tag{97}$$

where g_1 represents the weight matrix to be learned and $\delta(\cdot)$ represents the SoftMax function. The enhanced feature matrix is obtained by multiplying each element of the similarity score φ_n with each element u_n of the neighbouring point cloud. Then, the convolution stage is started with GCM. The correlation function is used so that the kernel function can learn the local geometric properties well while applying the convolution. The correlation function $\chi(a_m, u_n, u_m)$ is

$$\chi(a_m, u_n, u_m) = \frac{1}{\|\kappa\|} \exp\left(-\frac{\|a_m - \Delta\vartheta_n\|^2}{b\gamma^2}\right) \quad (98)$$

where κ denotes the number of kernel points, a_m denotes the relative coordinates ($a_m = A_m - A_0$) of the kernel points, γ denotes the parameter that determines the domain of the kernel points, $\Delta\vartheta_n$ and b denote the difference between neighbouring points ($\Delta\vartheta_n = \vartheta_n - \vartheta_m$) and a fixed value, respectively. Using the correlation function, the kernel function $\hbar(u_n, u_m)$ is

$$\hbar(u_n, u_m) = \sum_{i=1}^{\kappa} \chi(a_m, u_n, u_m)\phi \quad (99)$$

where ϕ represents the weight matrix for the MLP layers. Thus, in the GCM stage, local geometric features are correctly detected with convolutional kernels. In the last stage, a deep learning structure is established by weighting the feature matrix with AR. The weighted feature matrix ψ^w is

$$\psi^w = \delta\left(g_2\psi^f\right) \quad (100)$$

where g_2 represents the attentive score for aggregation, while ψ^f represents the feature matrix. The deep learning structure is built with pyramid-based residual learning. Experimental studies 8 big data models including ModelNet40 [27], ShapeNet-Part [28], PartNet [110], S3DIS [33], NPM3D [102], Semantic3D [46], Semantic-KITTI [81] and ScanNet [36] made using the base. FG-Net has proven to be state-of-the-art in the segmentation task, as well as having a great advantage over counterparts in terms of speed in terms of processing and understanding big data.

Since graph-based approaches are widely used in image processing problems, similarly, graph-based approaches have been used in 3D point data processing. In graph-based networks (GBN), each point in the point cloud data is considered a vertex of a graph, and the vertices form edges by establishing a direct connection with their neighbouring points. In point cloud processing methods developed with GBN, feature learning can occur in two types, spatial domain, and spectral domain.

36) EDGE-CONDITIONED CONVOLUTION BASED NETWORK

Simonovsky et al. developed the Edge-Conditioned Convolution-based Network (ECC) method, which is a graph-based deep learning structure that is suitable for all graphs but has been specially tested for the point cloud classification task [111]. The basis of ECC is the convolution and pooling operations created in the spatial domain. In the ECC method, the points form the vertices of the graphs and the direct connection between the vertices creates the edges. Let $G = (V, E)$ be a graph where V and E denote vertices and edges, respectively. In this case, by applying the ECC to the graph obtained feature matrix $\Psi^s(m)$ is

$$\Psi^s(m) = \frac{1}{|N(m)|} \sum_{n \in N(m)} \hbar^s_{nm} \Psi^{s-1}(n) + b^s$$
$$\text{s.t. } \hbar^s_{nm} = \lambda^s\left(\varphi(n, m); w^s\right) \quad (101)$$

where Ψ is the function that assigns a label to each vertex V, φ is the function that assigns a label to each edge E, s is the number of layers, N (m) is the set of neighbors, λ is the filter-generating network, w is the kernel weight, b is the bias and $\hbar$ is the edge specific wight matrix. Filter weights are trained on edge labels in graphs created from the point cloud. In each point sample, the edge information changes dynamically. While the convolution layers are connected to each other with the ReLU activation function, the aggregation process is performed with max-pooling. Cross-entropy is used for the cost function of the method. Neural networks designed with ECC end with an FC layer. In the benchmark made with the Sydney Urban dataset for the classification task, ECC outperformed the volumetric-based methods in particular.

37) KERNEL CORRELATION BASED NETWORK

The KCNet method was developed to extract local geometric features of point data in detail using kernel correlation (KC) and graph pooling (GP) [112]. The points where geometric features can be learned in local regions of point cloud form a kernel. Feature extraction is performed by calculating the relationship between the kernel and the graphically nearest neighboring points. KC function $\kappa(k, p_i)$ is

$$\kappa(k, p_i) = \frac{1}{|N(i)|} \sum_{s=1}^{S} \sum_{j \in N(i)} \lambda_d\left(k_s, p_j - p_i\right)$$
$$\text{s.t. } \lambda_d(\phi, \varphi) = \exp\left(-\frac{\|\phi - \varphi\|^2}{2d^2}\right) \quad (102)$$

where k is the point set kernel, p_i is the point data, p_j is one of the nearest neighboring points, $N(i)$ is the set of neighboring points, S is the number of learnable points, λ_d is the kernel function, d is the kernel width, $\|.\|$ is the operator that calculates the Euclidean distance between two points, ϕ and φ are the two input parameters (point data). In KCNet, Gaussian kernel is chosen as the kernel function. After KC, which is used to extract the local geometric structure only in the front part of the KCNet neural network, the local feature structure is obtained using graph max-pooling. In each layer, the ReLU activation function is used without batch normalization (BN). In the neural network designed for classification and segmentation, the remaining layers are the same as PointNet except for the structures mentioned above. As a result of the tests conducted on KCNet for classification and segmentation tasks, KCNet lagged behind many methods in terms of accuracy despite its state-of-the-art performance.

38) DYNAMIC GRAPH CNN

Wang et al. developed the Dynamic Graph CNN (DGCNN) method, which includes EdgeConv and dynamically generates graphs at each layer, for point cloud classification and segmentation tasks [113]. EdgeConv is the layer that applies convolution to the edges formed by constructing a local neighborhood graph in the local regions of the point cloud and also forms the core of the DGCNN. Let $G = (V, E)$ be a graph where V and E denote vertices and edges, respectively. If a point cloud set is expressed as $U = \{u_1, u_2, \ldots, u_s\}$, features of edges ε_{mn}, is

$$\varepsilon_{mn} = \psi_\delta (u_m, u_n) \tag{103}$$

where ψ is the nonlinear function, δ is the learnable parameter, u_m and u_n are the points at the center and in the neighbourhood, respectively. Output feature of EdgeConv, u'_m is

$$u'_m = \underset{n:(m,n)\in E}{\hbar} \psi_\delta (u_m, u_n) \tag{104}$$

$\hbar$ represents the output after any aggregation function. The method is dynamic since the graph $G^s = (V^s, E^s)$ changes at each layer. While ModelNet40 is used for classification tests of DGCNN, ShapeNet is used for segmentation tests. As a result of the classification tests, DGCNN is the state of art method.

Linked DGCNN (LDGCNN) method was developed to reduce the model size of DGCNN and improve performance [114]. The features of different dynamic graphs are hierarchically linked together to obtain edge vectors with LDGCNN. Compared to DGCNN, LDGCNN gives more accurate results in classification.

39) ADAPTIVE GRAPH CONVOLUTIONAL NEURAL NETWORKS

Each graph can be learned in accordance with its own purpose with the Adaptive Graph Convolutional Neural Networks (AGCNN) method, which was developed based on the fact that the required graph sizes and connections change accordingly when performing operations on real data [115]. A distance metric learning method is presented to make the graphs learning stage more effective. AGCNN derives its ability to adapt to changing graphs from the Spectral Graph Convolution Laplacian Learning (SGC-LL) layer, which can perform convolution in the spectral plane. The idea behind AGCNN is to learn the whole graph rather than edge features. The performance of the method is analyzed using the Sydney Urban Objects Dataset. While the robustness of AGCNN against noise and data loss is proved, it is seen that the method still gives acceptable results even if the receptive field used to calculate neighborhoods is reduced.

40) LOCAL SPECTRAL GRAPH CONVOLUTION

Another method developed in the spectral domain is the Local Spectral Graph Convolution based Network (LocalSpec GCN) [116]. Instead of processing the graph as a whole, LocalSpecGCN teaches the properties of the points in local regions and their nearest neighbors to the system with SpecGraphConv based on Fourier transform. In addition, instead of max-pooling, recursive clustering and pooling method is used to aggregate points according to their spectral coordinates. Thus, a more detailed feature extraction is provided to the method. Let $P = \{p_1, p_2, \ldots, p_s\}$ denote the point cloud and $G = (V, E)$ denote the graph where V (points, p) and E (weights, w) denote vertices and edges, respectively. The graph adjacency matrix W formed by the weights consists of the distance $W_{mn} = d(p_m, p_n)$ between two neighboring points measured by methods such as the Euclidean distance d. After the convolution process is completed, the recursive cluster pooling (RCP) algorithm is used, which is quite different from the pooling methods used in other methods. RCP is developed by using Fourier transform in the cluster part and max-pooling and average pooling methods in the recursive part. ModelNet40, McGill Shape Benchmark, ShapeNet and ScanNet Indoor Scene Dataset are used for testing the classification and segmentation tasks. LocalSpec GCN outperforms the other methods in the benchmarks for classification and segmentation. One of the most popular advantages of LocalSpec GCN is that it performs all operations end-to-end on the fly without any offline computation. This makes SpecGraphConv a good candidate for practical use.

41) REGULARIZED GRAPH CNN

The convolution process in the Regularized Graph CNN (RGCNN) method developed using the spectral graph theory is based on the Chebyshev polynomial approach [117]. While creating the graph in RGCNN, all points are connected to another point. Then, the graphics in each layer are updated with the Laplacian matrix. Let $G = (V, E, A)$ is an undirected graph where V denotes the vertices, E denotes edges connected to vertices and A denotes the weighted neighborhood matrix consisting of weight values $\phi_{m,n}$ connected to (m, n) edge. Normally the graph signal represents the points at the vertices of a graph, whereas the graph signal represents the feature vector ψ_m belonging to the m -th point for RGCNN. Weight $\phi_{m,n}$ is

$$\phi_{m,n} = \exp\left(-\alpha \left\|\psi_m - \psi_n\right\|_2^2\right) \tag{105}$$

where α denotes scalar parameter. The convolution between the two graph signals p and q, $p^* q$

$$p^* q = V\left(V^T p\right) \odot \left(V^T q\right), \tag{106}$$

where V and $\odot$ represent orthonormal eigenvectors and Hadamard multiplication, respectively. As a result of the spectral filtering of the graph signal p by f_φ, q is

$$q = V f_\varphi (S) V^T p \tag{107}$$

where S denotes the diagonal matrix. Due to the high computational complexity $O\left(n^3\right)$ of spectral filtering, it is aimed to approach spectral filtering by using truncated Chebyshev

polynomials. After the K localized filter is applied to p, output q is

$$q = f_\varphi(\Psi)\, p = \sum_{k=0}^{K-1} \varphi_k C_k(\Psi) p, \quad \text{s.t. } \Psi = VSV^T \tag{108}$$

where Ψ, φ_k and C_k represent the graph Laplacian matrix, the k-th Chebyshev coefficient, and the k-degree Chebyshev polynomial, respectively. ReLU activation function is used for convolution. Graph-signal smoothness prior (GSSP) has been added to the cross-entropy loss function to be more effective for segmentation as a loss function. The new loss function with GSSP added, $L\left(q^0, q'\right)$, is

$$L\left(q^0, q'\right) = -\sum_{m=1}^{s} q_m^0 \log\left(q'_m\right) + \delta \sum_{x=0}^{2} q_x^T \Psi q_x \tag{109}$$

where q^0 is the output score, q' is the ground truth label, δ and q_x are the penalty parameter for smoothing and the feature map of the x-*th* layer, respectively.The RGCNN method performs well in processing and understanding point cloud with noise and different point densities, while reducing computational complexity.

42) PyramNet

Zhiheng et al. developed an easy-to-use and effective method, PyramNet, which includes two main modules at its core, the Graph Embedding Module (GEM) and the Pyramid Attention Network (PAN) [118]. While GEM is used to generate graphs from point data, PAN is used to extract geometric properties of each point. In order to better perceive the geometric features, while the covariance matrix is based on the GEM instead of the Euclidean distance, all layers are connected to each other with the ReLU activation function. While global average pooling is used for PAN, max-pooling function is used in classification phase. Segmentation and classification tests were performed on ShapeNet, S3DIS and ModelNet40. As a result of the tests, PyramNet was found to be inferior to many methods in terms of accuracy.

43) DYNAMIC POINTS AGGLOMERATION BASED NETWORK

Liu et al. have created a network that will minimize the computational and memory burden by combining sampling, grouping and pooling operations in a single matrix with the help of the Dynamic Points Agglomeration Module (DPAM) which is developed throughout the method [119]. Although DPAM, which was developed for point classification and segmentation operations, is integrated into the PointNet structure, it is a module that can be integrated into different methods. Let the graph convolution network (GCN), P_ℓ is

$$P_\ell = AP_{\ell-1}W_\ell \tag{110}$$

where P, A and W represent the feature, neighborhood matrix and weight of the point, respectively. In this case, the mathematical model of the learning stage of DPAM agglomeration matrix $\alpha^{(\ell)}$ is applied to feature matrix $\Psi^{(\ell)}$ at layer ℓ and the feature matrix of point agglomerations $\Psi_k^{(\ell)}$ is

$$\Psi_k^{(\ell)} = \alpha^{(\ell)T}\Psi^{(\ell)}, \quad \text{s.t. } \alpha^{(\ell)} = \delta\left(GCN\left(A^{(\ell)}, \Psi^{(\ell)}\right)\right) \tag{111}$$

where δ represents and the SoftMax function. The SoftMax loss function is used during the training of the neural networks. Although the developed method using DPAM shows a state-of-the-art performance in the benchmark result, it lags behind many other methods in terms of accuracy during the examination of different methods. In addition, DPAM based method has the best inference time among the benchmarked methods except PointNet. In addition, the parameter sharing scheme reduces the computational complexity and memory usage of the method due to DPAM.

44) GLOBAL CONTEXT REASONING APPROXIMATION

Point Global Context Reasoning (PointGCR) module has been developed, which can increase the performance of the integrated method by capturing the integrity feature of point data [120]. PointGCR uses ChannelGraph, an undirected graph representation, to extract global contextual information. In order to process large amounts of data quickly, a random sampling method is used in the subsampling stage. The PointGCR module, which is a plug-and-play method for segmentation, was found to improve performance as a result of tests conducted by integrating it into different methods. In the tests performed by adding PointGCR to methods such as PointNet++ and PointConv, the segmentation accuracy rate increased by 1.2% on average.

45) GRID-GCN

Xu et al. developed the Grid-GCN method using their Coverage-Aware Grid Query (CAGQ) instead of time-consuming data structuring methods such as Farthest Point Sampling (FPS) and neighbor points querying, which are often used to process point cloud data [121]. The Grid-GCN structure consists of GridConv. The GridConv layer contains CAGQ for data structuring and Grid Context Aggregation (GCA) module for convolution. For CAGQ to work, the input space is first voxelized $\left(\vartheta_x, \vartheta_y, \vartheta_z\right)$. Each point is then mapped to a voxel index $V(\phi, \varphi, \theta)$ is

$$V(\phi, \varphi, \theta) = \text{floor}\left(\frac{x}{\vartheta_x}, \frac{y}{\vartheta_y}, \frac{z}{\vartheta_z}\right). \tag{112}$$

Voxels contain full and empty ones. Let all occluded voxels are represented by Θ_ϑ and sampled S center voxels are represented by Θ_χ. The voxels in the neighborhood of each center voxel ϑ_m are denoted by $\nrightarrow(\vartheta_m)$ and the points within the neighboring voxels are called context points. Different methods are used when selecting K nodes from context pointswith CAGQ. First, the center voxels sampling framework method, which includes random voxel sampling (RVS) and Coverage-Aware Sampling (CAS) methods, was developed to select the center sample points. Grid-GCN is a method that is more

robust to instability of point density, since each occluded voxel will have the same probability of being selected with RVS. In order for the selection of the central voxel set Θ_χ to cover the fullest space, CAS searches for the optimum result by navigating all selection combinations with the help of iteration. While randomly selected voxels from the central voxels Θ_χ with RVS are called incumbents, the voxels that will be selected as a result of each iteration using CAS from the unselected voxels are called challengers. Thus, each time an incumbent ϑ_{ic} is replaced by a challenger ϑ_{ch}. The coverage gains $\hbar_G$ and coverage loss $\hbar_L$ are

$$\hbar_G = \sum_{\upsilon \in \nrightarrow(\upsilon_{ch})} \delta\left(A_\upsilon\right) - \alpha.\frac{A_\upsilon}{\gamma} \tag{113}$$

$$\hbar_L = \sum_{\upsilon \in \nrightarrow(\upsilon_{ic})} \delta\left(A_\upsilon - 1\right) \tag{114}$$

where

$$\delta\left(a\right) = \begin{cases} 1, & \text{if } a = 0 \\ 0, & \text{otherwise} \end{cases} \tag{115}$$

where υ_{ch} is the coverage penalty parameter for the challenger, υ_{ic} is the coverage penalty parameter for the incumbent, δ is the Dirac delta function, P_υ is the number of incumbents covering voxel υ, α and γ are the regularization parameter and the number of neighbours of a voxel, respectively. If $\hbar_G > \hbar_L$, the incumbent is replaced by the challenger. Once the center voxels are sampled, it is time to select K nodes from the context points. First, K nodes are randomly selected from the context points by cube query (CQ). Then k-NN is applied to the context points to get the closest K points. Since the k-NN used here only works on context points, it is much faster than k-NN applied to all points. The GCA module is used for Convolution. The new feature matrix obtained as a result of GCA, $\tilde{\psi}_m$, is

$$\tilde{\psi}_m = G\left(\varepsilon\left(u_j, \psi_j\right)^* \mu\left(\psi_j\right)\right) \tag{116}$$

where G is the aggregation function (grid context pooling), ε is the edge attention function, u_j is the coordinate information of the node, ψ_j and μ are the node feature vector used as input and the multi-layer perceptron (MLP), respectively. The edge attention function used in Grid-GCN method is designed differently from its predecessors to more accurately extract the semantic relationship between nodes. The benchmarks for classification are based on ModelNet40, while the benchmarks for segmentation are based on ScanNet and S3DIS. Grid-GCN, which takes the advantages of both volumetric and raw point processing, shows state of the art performance in terms of accuracy, while tests with ScanNet, which contains 81920 points, have shown that it works 50x faster than existing methods.

In Hierarchical Data Structure-Based Methods, point features are learned hierarchically from branches to root in a tree structure such as octree and KD-tree [122]. KD-Net [123], 3DContextNet [124] and SO-Net [125] methods are analysed under this heading.

46) KD-Net

Klokov and Lempitsky developed the KD-Net method for extracting features from point data by indexing them into multiple Kd-tree structures [123]. The features of points are learned according to their local and global features by using MLP at each level. After the features of the nodes up to the node with no leaves are calculated with MLP, they are subjected to max-pooling. Thus, the feature of the node without a leaf is extracted. For the classification task, this process is repeated up to the root node. The non-linear activation function used between layers is ReLU. ModelNet and MNIST datasets are used for the classification task, while ShapeNet dataset is used for the segmentation task. The results of the classification accuracy tests show that KD-Net outperforms many methods.

47) 3DContextNet

Zeng and Gevers developed the 3DContextNet method, which utilizes the K-d tree to describe the features of point data both locally and globally [124]. First, the point cloud data is indexed with a K-d tree to prepare it for the feature extraction stage. At each level of the indexed K-d tree, the local and global features of the points are obtained by MLP layers and pooling operations. In the last stage, aggregation is applied to perform classification or segmentation tasks. After the points are transformed into a k-d tree structure, they enter the Adaptive Feature Recalibration (AFR) stage where local features are learned and the Non-local Responses (NLR) stage where global features are learned. The local features $\tilde{\psi}_m$ obtained by AFR, is defined as

$$\tilde{\psi}_m = \sigma\left(\alpha\left(\Psi\right)\right).\psi_m \text{ s.t.} \Psi = \{\psi_1, \ldots, \psi_s\} \tag{117}$$

where σ is the sigmoid activation function, α is the aggregation function, Ψ is the feature set of the points in the local region, s and ψ_m are the total number of points and the feature vector of the m th point, respectively. As a result of NLR stage, the global features g_m,

$$g_m = \frac{1}{N\left(p\right)} \sum_{\forall n} \lambda_G\left(p_m, p_n\right) \hbar\left(p_n\right)$$
$$\text{s.t. } \lambda_G\left(p_m, p_n\right) = e^{\varphi(p_m)^T \theta(p_n)} \tag{118}$$

where $N\left(p\right)$ is the normalization parameter, p_m is the point at m, p_n is the point at n, λ_G is the Gaussian function embedded between points m and n, $\hbar$, φ and θ denote the MLP functions, is obtained. Finally, aggregation is applied to perform classification or segmentation tasks. For classification, segmentation and part segmentation tasks, tests were performed using ModelNet40, S3DIS and ShapeNet datasets, respectively. According to the results, 3DContextNet outperforms many methods for classification and object segmentation tasks, while it outperforms many methods in the semantic segmentation task using S3DIS.

48) SELF-ORGANIZING NETWORK

The Self-Organizing Network (SO-Net) method provides hierarchical feature extraction by modelling points into nodes with the help of the Self-Organizing Map (SOM) module [125]. SO-Net uses the k-NN algorithm to model the points. While the features of points are learned through FC layers, the features of nodes are extracted with channel-wise max-pooling. A global feature is also extracted from the features of the nodes. SO-Net shows a state-of-the-art performance in the classification task with ModelNet40 data, as well as providing fast data training and being a method that is resistant to point corruption. ShapeNetPart database is used for the part segmentation task. As a result of benchmarks, SO-Net, which is similar to PointNet, gives more effective results compared to PointNet++. Thus, SO-Net proved to be a superior method in terms of accuracy, fast in terms of time and robust against harsh environmental factors. However, subsequent methods have been developed that provide better results in terms of accuracy. The examination of different methods, which do not fall into the above-mentioned classes but have given successful results, is also carried out.

49) POINT-VIEW NETWORK

Gao et al. developed the Point-View Network (PV-Net) method for object classification using point cloud and multi-view representations [126]. High-level global features of objects obtained using multi-view and features obtained from point cloud data are combined using embedding network and attention fusion blocks. According to the attention fusion block (AFB), first, local features $\psi(u)$ and multi-view features $\psi(v)$ of the point cloud data extracted using edge-conv, where u denotes the tensor consisting of point cloud features and v denotes the tensor consisting of multi-view features, are combined. The combined features are subjected to a normalization function after passing through the MLP layer. The normalization function $\eta(.)$ is

$$\eta(.) = \sigma(\log(abs(.))) \tag{119}$$

where σ and $abs(.)$ represent the sigmoid activation function and the absolute value operator, respectively. The final feature $\hbar(u, v)$ is

$$\begin{aligned} \hbar(u, v) &= \psi(u)^* (1 + S(u, v)) \\ \text{s.t.} S(u, v) &= \eta(MLP(\varphi(u, v))) \end{aligned} \tag{120}$$

where, $S(u, v)$ and $\varphi(u, v)$ represent the soft attention mask and the fusion function, respectively. The classification task is ultimately completed using the fused features. In the classification comparison using ModelNet40, PV-Net outperformed many methods.

Another method developed in a similar logic to PV-Net is the Point-View Relation Neural Network (PVR-Net) [127]. Using the PVR-Net method, the relationship between point cloud data and Multiview is extracted. The 2D global feature point single view fusion and point multi view fusion obtained with Multiview provide more detailed information. As a result of the tests, PVR-Net surpassed PV-Net in terms of accuracy.

50) 3D POINT CAPSULE NETWORKS

Zhao et al. developed a 3D Point Capsule Networks (3DPointCapsNet) with auto encoder [128]. In 3DPointCapsNet, point cloud inputs are routed to the decoder with dynamic routing (DR) after passing through MLP, convolution and max-pooling layers in the encoder part. In the decoder part, the data reaching the latent capsules are then processed in a way suitable for classification or segmentation. In the encoder stage of the 3D Point Capsule Networks scheme, the point cloud data is transformed into a 3D surface $S\{s_m\}$, which is a differentiable 2-manifold embedded in 3D Euclidean space. Then, through a series of convolution and max-pooling operations, the features of the obtained surfaces are merged as capsules $V\{v_1, \ldots, v_m\}$. The features that pass to the decoder stage with DR are used to obtain the best surface by looking at the chamfer loss between the 3D surfaces and the input surfaces. Chamfer loss $L_{ch}\left(S, \tilde{S}\right)$ is

$$\begin{aligned} L_{ch}\left(S, \tilde{S}\right) = &\frac{1}{|S|} \sum_{s \in S} \min_{\tilde{s} \in \tilde{S}} \|s - \tilde{s}\|_2 \\ &+ \frac{1}{\left|\tilde{S}\right|} \sum_{\tilde{s} \in \tilde{S}} \min_{s \in S} \|s - \tilde{s}\|_2. \end{aligned} \tag{121}$$

Three different activation functions are used during the encoder and decoder stages. In the encoder stage, the layers are connected to each other with ReLU, while in the decoder stage ReLU is used except for the last layer where the tanh activation function is used. The squash activation function is used throughout the DR layer. 3DPointCapsNet uses ShapeNet-Core, Shapenet-Part, ModelNet40 and 3DMatch datasets for tasks such as classification, reconstruction, part interpretation and part interpolation. One of the main advantages of 3DPointCapsNet is that it can be used in a new application such as part interpolation and replacement.

51) DEEP RBFNet

Chen et al. developed the Deep Radial Basis Function Network (Deep RBFNet) method in which Radial Basis Function (RBF) kernels and nonlinear activation layer are encoded [129]. First, the spatially transformed point data are subjected to feature extraction in the layer with RBF kernels. After feature extraction, the data is passed to a vanilla network consisting of a single MLP layer or an enhanced network consisting of multiple MLP layers to extract global features. RBF, $\lambda(u, v)$, is

$$\lambda(u, v) = \lambda(\|u - v\|) \tag{122}$$

where u is the input, v and $\|u - v\|$ represent the center of the RBF kernel and the Euclidean distance between two points, respectively. The Gaussian function $G(u, v)$, often used as

RBF, is defined as

$$G(u, v) = \exp\left(-\frac{(u-v)^2}{\kappa^2}\right) \tag{123}$$

where κ represents the kernel size. The ReLU activation function is used for the FC layer, while MNIST and ModelNet40 databases are used for the classification task tests. Deep RBFNet, in addition to its state-of-the-art performance, reduces the computational cost by reducing the network parameters needed and can run faster than existing methods. The fast speed of the method also means that it can be integrated into portable devices with limited resources.

52) Point2Sequence

Recurrent neural network (RNN) based Point2Sequence method was developed for shape classification and segmentation tasks on point cloud data [130]. The learning model of the correlation relationship between different areas in the local regions of the data is created with RNN. Point cloud data is processed in 6 stages. In the first stage, k-NN is applied to the data sampled with FPS to determine the nearest neighbours. The first stage ends with grouping. In the second stage, a series of MLP layers with abstraction and max-pooling operations are used to extract the features of the points. In the third stage, an attention-based encoder-decoder network (ABEDN) developed with RNN is used to capture the fine-grained features of local regions. The point features $\Psi_m = \{\psi_{1,m}, \ldots, \psi_{i,m}, \ldots, \psi_{N,m}\}$ in each local region are processed by entering the RNN-based encoder stage consisting of a hidden unit $\hbar$. Accordingly, the hidden unit $\hbar_i$, is

$$\hbar_i = \lambda\left(\hbar_{i-1}, \psi_{i,m}\right) \tag{124}$$

where λ represents the non-linear activation function, which in the Point2Sequence represents the long short-term memory (LSTM) unit, while $\hbar_{i-1}$ represents the previous hidden unit. The encoder output o_i is

$$o_i = W_e \hbar_i \tag{125}$$

where W_e represents the learnable weight matrix. Similarly, in the decoder stage, which consists of a hidden unit, the updated hidden unit $\tilde{\hbar}_1$ is

$$\tilde{\hbar}_1 = \lambda\left(s_0, \hbar_N\right) \tag{126}$$

where s_0 and $\hbar_N$ represent the zero state and the hidden unit updated in the last step of the encoder, respectively. The decoder output $\tilde{o}_1$ is

$$\tilde{o}_1 = W_d \tilde{\hbar}_1 \tag{127}$$

where W_d represents the learnable weight matrix for the decoder. During the decoder stage, the hidden units updated with the attention mechanism are connected to each other with the tanh activation function. In the fourth stage, global features are collected from local features, similar to PointNet structure. In the fifth stage, classification is made using FC layers connected to each other with ReLU. In the sixth and last stage, the partitioning task is performed by using shared MLP, FC and upsampling mechanisms. FC layers are also connected to each other with ReLU. The interpolation process used for upsampling (Inverse Square Euclidean distance weighted average based on k-NN) $\varphi(x)$ is

$$\varphi(x) = \frac{\sum_{j=1}^{n} d\left(x_j\right) \varphi\left(x_j\right)}{\sum_{j=1}^{n} d\left(x_j\right)}$$
$$\text{s.t. } d\left(x_j\right) = \frac{1}{x - x_j} \tag{128}$$

where x represents the points and $d\left(x_j\right)$ represents the inverse square Euclidean distance between the points $\left(x, x_j\right)$. Point2Sequence shows that superior performance compared to many methods as a result of tests for classification and part segmentation tasks. Additionally, Point2Sequence is the first method to process and sense point cloud data with RNN.

53) RECURRENT SET ENCODING

The Recurrent Set Encoding based Network (RCNet) method has been developed, which makes the typical 2D convolution methods usable by dividing the space where the point cloud data is located [131]. In the RCNet, the STN represents the spatial transformation network, the MP represents max-pooling, and the FC represents the fully connected layer. The data is first turned into a grid by dividing into parallel beams with the help of an encoder. Then, the points inside the beams modelled as sequences are encoded into geometric features with RNN layers. After the encoded features pass through a series of convolution and MP layers, they continue in separate ways for classification and segmentation processes.

RCNet has been evolved into the RCNet-Ensemble (RCNet-E) method by further developing in order to extract more detailed features. While RCNet splits data into beams on a single axis like z, RCNet-E splits points into beams on three axes (x, y and z), providing more detailed features. In benchmarks, RCNet showed a state of art performance and RCNet-E improved this performance a bit.

IV. 3D OBJECT DETECTION AND TRACKING METHODS

are used in many areas from daily life tasks such as autonomous driving to defense and military tasks such as target detection and target locking of unmanned aerial vehicles. In 3D object detection, objects are detected in an input consisting of point cloud data and the object is framed with the oriented bounding box. In object tracking, identification numbers (ID) are given to the objects in the frame, so that the same object is marked with the same ID in the next frame. 3D object detection and tracking methods developed for the processing of point cloud data are examined.

A. REGION PROPOSAL-BASED METHODS

In Region proposal-based methods, possible regions containing objects are first identified and then the properties of these regions are extracted. Methods were examined in

three categories: Multi-mapping approach, Segmentation-based approach and Frustum-based approach.

In the multi mapping approach, detection of objects is aimed by combining data obtained with different imaging types such as LiDAR front view, Bird's Eye View (BEV), and RGB image. The process of combining different types puts a heavy computational burden on the methods. In the field of object detection and tracking, different methods have been developed to increase the accuracy rate and to make a fast calculation.

1) MV3D

Chen et al. developed the Multi-View 3D Object Detection (MV3D) method that estimates the 3D bounding box by combining front view (FV) and Bird's Eye View (BEV) images from LiDAR with RGB images [132]. In MV3D, first, region estimates of objects in BEV images are extracted. Then, FV, BEV and RGB images are combined with deep fusion, which includes 2D convolution and pooling processes. Finally, the bounding box drawing operation is performed. As a result of the tests performed using the KITTI dataset, the method achieves 99.1% success in terms of the Intersection over Union (IoU) parameter, but the working speed is quite low.

2) AVOD

Aggregate View Object Detection network (AVOD) is a method developed for object detection to be used mostly in autonomous driving technologies [133]. The method, which consists of two main layers, a region proposal network (RPN) and a second stage detector network (SDN), is based on the logic of combining LiDAR's BEV image and RGB image. After the features of the images are combined in high resolution with RPN, the object is enclosed in a bounding box with SDN. Although AVOD is basically based on the same idea as MV3D, it is considered to be a candidate to be used in autonomous driving due to the performing better in terms of running speed.

3) ContFuse

Deep Continuous Fusion (ContFuse) is a method developed for object detection by combining LiDAR and RGB camera images at different resolution levels [134]. The features of the images obtained from the camera are determined with ResNet blocks in the first stage, and they are combined in the multi-scale fusion layer and passed to the fusion layers. Since the data obtained with LiDAR has a discrete format, the gaps between the data are filled with features obtained from RGB images by using the continuous fusion (CF) method found in fusion layers. CF,$\hbar_m$, is

$$\hbar_m = \sum_n MLP\left(concat\left[\psi_n, p_n - p_m\right]\right) \tag{129}$$

where ψ_n is the input image feature of the n point, $concat\,[.]$ is the vectors coupling operator, p_n and p_m represent the neighbouring and target points, respectively. The target pixels are determined by sending the features extracted from the image and continuous geometric information to the MLP. In the training stage, the sum of classification loss and regression loss is used as a cost function. Binary cross entropy is used for classification loss. As a result of tests on KITTI and TOR4D [135] databases, ContFuse has demonstrated state of art performance.

4) MMF

Multi-Task Multi-Sensor Fusion (MMF) method has also been developed by combining LiDAR and RGB camera images [136]. The method aimed to detect the 3D object as accurately as possible by performing multiple tasks such as object detection, depth completion and ground estimation. In the training stage, the binary cross-entropy loss function is used for classification, while the l1 loss function is used for box estimation. Tests with KITTI and TOR4D also gave much better results than previous MMF methods.

5) RT3D

In order to increase the working speed of multi-view-based approaches, Zeng et al. developed the Real-Time 3-D Vehicle Detection (RT3D) method, which is a suitable method for autonomous driving [137]. With Pre-RoIpooling convolution, most of the functions used in convolution are run before the RoI pooling layer, significantly increasing the computational speed. In addition, the pose-sensitive feature map module, which captures the features of the vehicles from different angles with the RT3D method, increases the accuracy of the location and detection of the vehicles even more. As a result of tests using the KITTI dataset, RT3D works 5x faster than MV3D.

6) IN SEGMENTATION-BASED OBJECT DETECTION METHODS

consists of two stages. Firs, most of the background information in the images is removed with semantic segmentation methods. Second, object detection algorithms after removing background are applied on the remaining points. Thus, the computational burden is reduced as fewer points will be processed. In images with many objects, segmentation-based methods perform better than multi-view-based approaches.

7) IPOD

Segmentation-based Intensive Point-based Object Detector (IPOD) method consists of three main layers [138]. In the first layer, foreground pixels are estimated by applying segmentation network to 2D images. After the estimation of the front pixels, the predicted background information is removed by projecting to the point cloud. In the second layer, PointNet++ based feature extraction and estimation of the areas containing the objects are made. In the last layer, the detected objects are taken for the frame. Multi-task loss consisting of classification, location regression, angle and corner loss functions is used to train the neural network. Softmax cross entropy loss function is used as classification cost function. Location regression loss consists of T-Net

center estimation, center residual prediction and size residual prediction loss functions. Angle loss includes the functions of orientation classification and residual prediction loss, while corner loss includes the distances between 8 corners and ground truth. IPOD demonstrated state of art performance in tests with the KITTI dataset.

8) PointRGCN

Zarzar et al. developed the PointRGCN method, an object detection method based on graph convolutional networks (GCN) [140]. PointRGCN consists of three basic layers in total. The first layer is the region proposal network (RPN) derived from the PointNet++ structure. Boxes delimited using point cloud data with RPN are recommended. To further improve object detection, residual GCN (R-GCN) modules are used as the second layer and contextual GCN (C-GCN) modules as the third layer. While 3D proposals are classified and restricted with R-GCN, it is aimed to provide more accurate object detection by sharing contextual information between proposals with C-GCN. The convolution structure used in the C-GCN layer is edgeconv. While binary cross-entropy loss is used for classification during training, smooth L1 loss function is used for regression. PointRGCN, which has been developed for autonomous driving, outperformed PointRCNN as a result of comparisons with the KITTI dataset. In addition, the method gives 2.13% better results than PointRCNN for object detection on BEV images in the easy and difficult categories.

9) PointPainting

PointPainting was developed as a method applicable to different object detection and tracking methods based on LiDAR image processing [141]. The PointPainting consistis of three stages. In the first stage, semantic segmentation is done in 2D images. In the second stage, painting, the interpreted features are projected on the front-view point cloud data obtained with LiDAR. Homogeneous and camera matrices are used for projection. In the third and final stage, object detection is performed on LiDAR data. KITTI and nuScenes datasets are used for application tests. PointPainting provides a great improvement in accuracy by integrating with methods such as PointRCNN and VoxelNet. In addition, the method increases the working speed by reducing the delay time.

10) STD

The Sparse-to-dense 3D Object Detector (STD) method developed for object detection is based on the PointNet-based proposal generation network (PGM) module [142]. With PGM, each point is associated with a spherical anchor and proposals containing the objects are extracted. The advantages of both voxel and point cloud approaches are used in the PointsPool layer to strengthen the prediction of objects. The PointsPool layer transforms sparse point cloud data into a compact structure, reducing the runtime inference. While STD detects objects in the difficult category in the KITTI database at a state of art level, it increases the average FPS inference rate 10 times compared to other methods.

11) THE BASIS OF FRUSTUM-BASED METHODS

is to extract the 3D frustum proposal for each 2D region after detecting the possible regions containing the objects on the 2D images. Although frustum-based methods are good at region estimation, the limitation of 2D images is a disadvantage.

12) FRUSTUM PointNets:

In the Frustum-PointNets (F-PointNets) method, which is the first of the frustum-based methods, firstly an object detection algorithm is applied on 2D images, and then the data is projected to 3D and converted to the frustum base [143]. One of the advantages of F-PointNet is the abundance of object detection algorithms for 2D images. Since operations are performed on RGB-D data, transition from 2D to 3D is provided with depth information. In the middle layer, after the segmentation is applied to the data, binary classification is made. In the last layer, the detected objects are framed by applying box estimation to the classified data. Application tests for F-PointNet, which essentially contains the PointNet structure, were performed on KITTI and SUN-RGBD datasets. Since F-PointNet processes directly on raw point cloud data, it works fast and is suitable for real-time systems. In addition, the method successfully detects objects even under strong congestion or in very sparse spots.

13) SIFRNet

Zhao et al. developed the Scale Invariant and Feature Reweighting Network (SIFRNet) method, which is used for object detection. SIFRNet consists of 3 main modules, 3D instance segmentation network (Point-UNet), T-Net and 3D box estimation network (Point-SENet) [144]. Box estimation is performed by applying Point-UNet, T-Net and Point-SENet to the frustum structure formed after object detection on 2D images, respectively. SIFRNet, which was superior to other methods, including in F-PointNet, was later integrated with PointSIFT to provide even better performance. The datasets used for the tests are KITTI and SUN-RGBD.

14) PointFusion

In the PointFusion method, after RGB image and point cloud data are processed separately in ResNet and PointNet structures, box estimation is performed by combining them with the fusion algorithm [145]. Benchmarks were made with KITTI and SUNRGB-D datasets.

15) RoarNet

With RegiOn Approximation Refinement Network (RoarNet), 3D poses are extracted from 2D images after objects are detected in 2D images and box estimation is done [146]. Finally, 3D box regression is used to make an accurate 3D box estimation. RoarNet demonstrates state of art performance on the KITTI dataset.

16) F-ConvNet

Feature vectors were extracted by applying PointNet to each 2D image arrayed as frustum in the Frustum ConvNet (F-ConvNet) method [147]. Then, box estimation is made from feature vectors subjected to fully convolution network (FCN). F-ConvNet ranks top in the KITTI dataset benchmark for accuracy and speed.

17) PATCH REFINEMENT

After making initial predictions on the BEV image with the Patch Refinement-based network Region Proposal Network (RPN), it extracts patches on these predictive regions [148]. Then, 3D box estimation is made by learning the local properties of the patches with the Local Refinement Network (LRN).

PointVoxel-RCNN (PV-RCNN), one of the methods in the other category, has the best score in the car class parameter in the KITTI dataset on 12 June 2020 [149]. With PV-RCNN, point cloud data is first converted to voxel view, and then 3D convolution is applied to produce high quality proposals by extracting voxel point features. The data with extracted voxel features are coded into small clusters using the voxel set abstraction module. Finally, with the box estimation, the objects are enclosed in the frame.

3D IoU loss [150], Fast Point R-CNN [151], VoteNet [152], ImVoteNet [153] and Part-A^2 [154] methods, which are not examined in this study, but which show state of art performance, are also available in the literature.

B. SINGLE SHOT NETWORKS (SSN)

Single shot networks (SSN) are methods that perform direct object detection and box estimation without the need for region proposal generation (RPG) modules. SSNs work fast as the RPG process is disabled. The methods that are divided into BEV-based, discretization-based and point-based according to the data entry type are eamined.

1) BEV BASED METHODS

haven been developed by using input data in BEV image type.

2) PIXOR

PIXOR was developed for autonomous driving applications by performing real-time object detection from BEV images [155]. Point cloud data is first divided into equal parts and a scene flow structure is created. Then, object detection is performed by applying FCN to images converted into regular structures. It has been proven as a result of studies that PIXOR performs better than most of the single shot methods.

3) HDNET

The HDNET method offers high definition (HD) maps in order to improve the performance of existing object detection algorithms [156]. Since HD maps cannot be provided from every region, an online map prediciton module that will extract HD maps from the LiDAR point cloud has also been developed for the method. It has been determined by experimental studies that the methods used together increase the performance.

4) BirdNet

BirdNet performs object detection in three stages. In the first stage, point cloud data is projected to BEV format [157]. In the second stage, object locations are detected with CNN. Finally, 3D oriented box drawing is performed.

5) IN DISCRETIZATION-BASED METHODS,

After point cloud data is converted into a discrete presentation with a regular structure, it is applied with CNN to perform object detection and box prediction drawing.

6) VeloFCN

VeloFCN, developed in accordance with the datasets obtained with Velodyne LiDAR, converts point cloud data into 2D images using 2D point mapping [158]. Then, object detection and bounding box drawing are made with FCNs.

7) Vote3Deep

Since empty voxels slow down computations in voxel-based approaches, the Vote3Deep method, which estimates only filled voxels and applies CNN, has been developed [159]. Object detection and box estimation are performed with the method developed to solve the sparsity problem.

8) 3DBN

Three-Dimensional Backbone Network (3DBN) method performs object detection and box estimation tasks by subjecting point cloud data converted to voxel presentation to a series of 3D CNN structures [160]. Thanks to 3DBN, the burden on the memory is reduced by using all of the voxels and the calculation speed is increased.

9) VoxelNet

VoxelNet converts the input point cloud data to voxels in equal parts and each voxel is sent to the voxel feature encoding (VFE) layers [161]. The properties coded into 4D tensors pass through the convolution layer, and the detection of objects and box estimation are completed with the latest region proposal network. While VoxelNet is good at detecting objects correctly, sparsity is slow due to the voxelization.

10) MVX-Net

Multimodal VoxelNet (MVX-Net) has been developed by further improving VoxelNet [162]. MVX-Net is faster than previous methods and achieves more accurate results. Object detection is performed by combining the features extracted from the RGB image and the features extracted from the point cloud.

11) PointPillars

PointPillars applies PointNet for feature extraction after arranging the point cloud data vertically (pillar) [163]. The

learned features are then coded as pseudo-images and subjected to 2D convolution layers. Finally, box estimaiton is done. PointPillars, which works fast, is superior to many methods in the KITTI benchmark in terms of Average Precision (AP).

12) SA-SSD

Zhang et al. developed an object detection network based on structure-aware single-stage detector (SA-SSD) [164]. Firstly, the point cloud data is converted to tensors and then its properties are extracted by entering the backbone network. Finally, object detection is performed with the detection network.

C. POINT BASED METHODS

are methods that directly process point cloud data. The general characteristics of frequently used methods are given below.

1) 3DSSD

With the 3D Single Stage Object Detector (3DSSD), the first of the Point based methods, up sampling layers and refinement stage layers have been abandoned in order to reduce the computational cost [165]. Fusion sampling has been developed for Distance-FPS (D-FPS) and Feature-FPS (F-FPS) in order to shorten the time spent in the Feature Propagation (FP) layer. In the Candidate generation (CG) layer, features are extracted and sent to the final step for object detection. In KITTI and nuScenes datasets, 3DSSD is ahead of most methods in terms of both speed and accuracy.

Among other methods, methods such as LaserNet [166], LaserNet++ [167], OHS-Dense [168], OHS-Direct [168] and Point-GNN [169], which have shown state of art performance, are also included in the literature.

2) OBJECT TRACKING

Is to detect and follow the object detected in one frame in the next frames. Point cloud data processing for 3D object tracking also has its challenges, such as occlusion, illumination, and scale variation. Many methods have been developed for object tracking by overcoming these difficulties.

3) 3D SIAMESE TRACKING

The siamese tracking method developed for 2D images has been adapted for object tracking in point cloud data [170]. With the 3D siamese tracking method, firstly the candidates are determined with the Kalman filter, and then the model and candidate shapes are coded into the compact latent representation. With cosine similarity, the region where the previously detected object is located in subsequent frames is searched. The tests were made for cars in the KITTI dataset and the result was 76.94% successful.

4) COMPLEXER-YOLO

In the Complexer-YOLO method, which is suitable for real-time applications, RGB images and voxelized point cloud images are used for accuracy and multi-target detection [171]. In the first stage, while segmentation is applied to the RGB image, the point cloud data is transformed into voxel. The two images obtained are then combined and entered into the network layer where predictions will be made. Multiple target detection and tracking can be done in real time with Complexer-YOLO. In addition, there is a Scale-Rotation-Translation score (SRTs) module that will increase the speed in the method.

D. 3D SCENE FLOW ESTIMATION

Is based on 2D optical flow. However, as in other 3D computer vision tasks, the irregular structure of the point cloud presents some difficulties in scene flow. 3D scene flow is to extract the motion plan from the point cloud data where the points in the data are at the time t1 according to their location at the time t2. Methods developed for 3D scene flow, which is an essential topic for robotics and autonomous vehicles, have been examined.

1) FlowNet3D

One of the first studies in the field of 3D scene flow, FlowNet3D estimates scene flow by working directly on point cloud data [172]. Thanks to the flow embedding layers, point feature and motion feature are extracted. FlowNet3D has two big problems. The first is that motion vectors differ significantly from ground truth, and the second is that FlowNet3D is a difficult method to implement in dynamic screens.

2) FlowNet3D++

Wang et al. developed the FlowNet3D++ method, which will improve the accuracy and performance by eliminating the problems experienced by FlowNet3D [173]. In order to minimize the error rate between ground truth and predictions, the cosine distance loss algorithm is used in the method. A point-to-plane distance loss algorithm has been developed to increase the accuracy on dynamic displays. According to the experimental results, the correct prediction rate of FlowNet3D increased from 57.85% to 63.43% compared to FlowNet3D, while its operating speed increased.

3) PillarFlowNet

The PillarFlowNet method has been developed using a single network to perform multiple tasks in real time [174]. The computational complexity caused by the use of separate deep neural networks in normal methods is reduced in PillarFlowNet thanks to the single network. After the features are encoded with the feature encoding network from the LiDAR point cloud data, they are sent to the layers that will perform the 3D object detection and scene flow estimation tasks. With the benchmark made using the KITTI dataset, an average precision score of 16.3% is achieved.

4) MeteorNet

Dynamic point cloud sequences are learned and the task of scene flow is fulfilled with the MeteorNet developed by Liu et al. [175]. With the developed model, spatiotemporal neighbourhoods are created for each point directly on the point cloud data. The points collected from the neighbourhoods and the characteristics of the points are learned and used in different tasks such as action recognition, semantic segmentation and scene flow estimation.

The methods analyzed in this section are grouped according to high-level tasks such as classification, segmentation, detection, tracking and optical flow of objects. The most frequently used methods in the classification and segmentation task group are those in the raw point cloud processing subgroup. Since the methods in the Multiview subgroup try to extract a three-dimensional meaning and analysis from 2D images, information losses and computational burden are high. Methods in the volumetric subgroup such as Multiview have high computational costs since they first apply voxelization to the 3D point cloud data and then use these voxels.

Methods that process point cloud data directly, on the other hand, are more preferred and more successful since they do not need to use any transformations or 2D images. Direct point cloud data processing methods, which are presented in 5 sub-categories as Pointwise MLP, CNN, Graph, hierarchical data structure and other, achieve different successes in different criteria. Each category has its own advantages and disadvantages. Since the methods in the Pointwise MLP category use only MLP and point cloud data, they generally have superior operating speed, accurate classification and segmentation capabilities. The advantages of the methods in the convolution category are that 2D or 3D convolution methods can be adapted to existing methods. However, a transform is needed for the convolution to become viable. Moreover, transform adds an extra computational cost to the methods and reduces the working speed. Graph-based approaches category, on the other hand, has a natural compatibility in handling point cloud data. Points can generate graphs by connecting to each other, just like in graph theory. In addition to this natural advantage of graphs, graphs also have their own evolved Fourier transform function, like the Fourier transform that can be used in other methods. Thus, the presence of the graph Fourier transform offers the graph approach the opportunity to operate in the spectral domain as well. The methods in the hierarchical data structure-based category perform classification and segmentation tasks using indexing structures such as standard kd tree. Although the methods have a significant advantage in terms of speed thanks to indexing, they perform poorly in terms of correct classification rate compared to other categories. The methods in the other category are different from each other in terms of approach, but show a state of art performance in terms of accuracy and speed. In addition, the methods in this category provide ideas on how different approaches can evolve into point cloud processing. For example, the point2sequence method shows for the first time how the recurrent neural network (RNN) structure can be used for point cloud.

Object detection and tracking classification have been examined in a separate category. The object detection and tracking task group have been also analyzed under two main headings as regional and single shot. Regional is divided into subgroups and methods in 4 sub-levels, namely multimapping, segmentation, frustum-based and other, have been examined. Methods in the category of multimapping have developed object detection methods by combining 3D and 2D images. In the segmentation-based category, on the other hand, it is ensured that the objects can be detected more easily by first extracting the background information from the frame containing the object. In frustum-based methods, object detection is performed after removing possible regions with objects in 2D images. While making region estimations with high accuracy in 2D images is an advantage, the possibility of loss of information in three dimensions due to the use of two-dimensional images is a disadvantage. The methods in the Single Shot networks category can detect objects without the need for regional estimation. Among the methods examined in 3 sub-groups, BEV, discretization and point-based, those in the BEV-based group operate in the BEV image type. Discretization-based methods, on the other hand, can make object detection with structures such as CNN by transforming the point cloud space into a discrete space. Methods in the point-based approach, on the other hand, detect objects directly in point cloud data. The methods in the SSN category generally perform well in terms of speed, as they do not need any preliminary estimates. When SSN subgroups are analyzed, the discretization-based approach works slower than the other two approaches because there is an extra computational cost for discretization. Having excelled in object tracking task in 2D images, YOLO has also successfully evolved into 3D point cloud data.

The optical flow task of 3D objects has been also examined under a separate heading. 3D scene flow, which is similar to 2D optical flow, has been developed in accordance with PillarFlowNet real time systems, which is one of the methods developed in the task group.

V. COMPARISON OF METHODS

Methods developed for processing and understanding point cloud data have advantages and disadvantages in many different categories such as speed, data size, efficient use of memory, noise resistance and accuracy. In this section, a comparison of the methods in different categories is made. Mean accuracy (mAcc), overall accuracy (OA) and mean intersection over union (mIoU) metrics are generally used for accuracy performance measurement of tasks such as classification and segmentation. While mAcc and OA are mostly used for 3D object classification, mAcc, OA and mIoU parameters are generally used for 3D segmentation [176]. To formulate the metrics, let ρ_i represent the number of true positives, μ_i the

TABLE 1. Comparison of methods for 3D classification task.

Methods		Parameters	(OA)	(mAcc)
Multi-View-Based Approach	Multi-view Convolutional Neural Networks (MVCNN)	-	90.1	-
	GIFT	-	-	83.1
	Multi-view Harmonized Bilinear Network (MHBN)	-	-	92.2
	Multi-view Spherical Projections	-	-	-
Volumetric Based Approach	VoxNet	-	85.9	83
	VRN	-	-	91.33
	VRN-Ensemble	-	-	95.54
	3D ShapeNets	-	84.7	77.3
	PointGrid	-	92	-
Octree Based Approach	OctNet	-	-	87.83
	Octree-based Convolutional Neural Networks (O-CNN)	-	-	90.6
Pointwise MLP Methods	PointNet	3.47M	89.2	86
	PointNet ++	1.74M	90.7	-
	MomentNet	-	92.4	90.3
	PATs	-	91.7	-
	LSA-Net	-	92.3	89.2
	PointWeb	-	92.3	89.4
	Structural Relation Network (SRN)-PointNet++	-	91.5	-
	JUSTLOOKUP	-	89.5	86.4
	ShellNet	-	93.1	-
	PointASNL	-	93.2	-
	RPNet	-	94.1	-
	PointTransformer	-	93.7	90.6
	PointMLP	-	94.5	91.4
Convolutional Neural Network (CNN) based methods	Pointwise-CNN	-	86.1	81.4
	PointCNN	-	92.2	88.1
	Flex-Convolution based Network	-	90.2	-
	PCNN	-	92.3	-
	SpiderCNN	-	92.4	-
	Monte Carlo Convolution Neural Network (MCCNN)	-	90.9	-
	PointConv	-	92.5	-
	Geo-CNN	-	93.4	91.1
	LP-3DCNN	-	-	92.1
	Annularly Convolutional Neural Networks (A-CNN)	-	92.6	-
	Kernel Point Convolution (KPConv)	-	92.9	-
	Relation-Shape Convolution (RS-Conv)	-	93.6	-
	Spherical Fractal Convolutional Neural Networks (SFCNN)	-	91.4	-
	Interpolated Convolutional Neural Networks (InterpCNN)	-	93	-
	DensePoint	-	93.2	-
	Continuous Convolutions (ConvPoint)	-	91.8	88.5
	CurveNet	-	94.2	-
Graph based methods	Edge-Conditioned Convolution based Network	-	87.4	83.2
	Kernel Convolution Network (KCNet)	0.9	91	-
	Dynamic Graph-CNN (DGCNN)	1.81	92.2	90.2
	Linked DGCNN (LDGCNN)	-	92.9	90.3
	Adaptive Graph Convolutional Neural Networks (AGCNN)	-	93.4	90.7
	LocalSpecGCN	-	92.1	-
	Regularized Graph CNN (RGCNN)	2.24	90.5	87.3
	PyramNet	-	91.5	-
	Dynamic Points Agglomeration Module (DPAM)	-	91.9	89.9
	Grid-GCN	-	93.1	91.3
Hierarchical Data Structure-Based Methods	KD-Net	-	91.8	-
	3DContextNet	-	91.1	-
	Self-Organizing Network (SO-Net)	-	90.9	-
Other Networks	Point-View Network (PV-Net)	-	93.2	-
	PVR-Net	-	93.6	-
	3D Point Capsule Networks (3DPointCapsNet)	-	89.3	-
	Deep RBFNet	3.2	92.1	88.8
	Point2Sequence	-	92.6	-
	Recurrent Set Encoding based Network (RCNet)	-	91.6	-

TABLE 2. Comparison of methods for 3D semantic segmentation task.

Methods		S3DIS (6-fold)		
		(OA)	(mAcc)	mIoU
Octree Based Approach	OctNet	81.5	-	59.2
	Octree-based Convolutional Neural Networks (O-CNN)	-	-	-
Pointwise MLP Methods	PointNet	78.62	66.2	47.71
	PointNet ++	81.03	67.05	54.49
	PointSIFT	88.72	-	70.23
	Point neighbourhoods-based segmentation	83.95	67.77	58.27
	MomentNet	-	-	-
	PATs	64.3	-	-
	LSA-Net	86.8		62.2
	PointWeb	87.31	76.19	66.73
	Structural Relation Network (SRN)	-	-	-
	JUSTLOOKUP	-	-	-
	ShellNet	-	-	66.8
	RandLA-Net	88	82	70
	PointASNL	88.8	79.0	68.7
	RPNet	-	-	70.8
	PointTransformer	90.2	81.9	73.5
	SCF-Net	88.4	82.7	71.6
	BAAF-Net	88.9	83.1	72.2
	PointMLP	-	-	-
	Paintwise-CNN	81.5	-	-
Convolutional Neural Network (CNN) based methods	PointCNN	85.9	-	65.4
	Flex-Convolution based Network	-	-	-
	PCNN	88.14	75.61	65.39
	SpiderCNN	-	-	-
	PointConv	-	-	-
	Annularly Convolutional Neural Networks (A-CNN)	87.3	-	62.9
	Kernel Point Convolution (KPConv)	-	79.1	70.6
	Relation-Shape Convolution (RS-Conv)	-	-	-
	Interpolated Convolutional Neural Networks (InterpCNN)	88.7	-	66.7
	DensePoint	-	-	-
	Continuous Convolutions (ConvPoint)	88.8	-	68.2
	Kernel Convolution Network (KCNet)	-	-	-
Graph based methods	Dynamic Graph-CNN (DGCNN)	84.1	-	56.1
	Adaptive Graph Convolutional Neural Networks (AGCNN)	90	73.2	67.9
	PyramNet	85.6	-	55.6
	Dynamic Points Agglomeration Module (DPAM)	87.6	-	64.5
	KD-Net	-	-	-
H.D. S.B.M.	3D ContextNet	84.9	-	55.6

number of predicted positives, and G_i the number of true positives. Accordingly, the mAcc is

$$mAcc = \frac{1}{N}\sum_{i=1}^{N} \lambda_i s.t. \lambda_i = \frac{\rho_i}{G_i} \tag{130}$$

where N represents the number of classes and λ_i represents the accuracy rate. OA and $mIoU$ are

$$OA = \sum_{i=1}^{N} \rho_i \Big/ \sum_{i=1}^{N} \mu_i \tag{131}$$

$$mIoU = \frac{1}{N}\sum_{i=1}^{N} IoU_i$$
$$s.t.\ IoU_i = \frac{\rho_i}{(G_i + \mu_i - \rho_i)} \quad (132)$$

Table 1 shows the comparison results according to the mAcc and OA metrics for the classification task. The values in the column named parameters in the table give the average number of parameters in the relevant networks are used. For example, the PointNet network uses 3.47 million parameters. Two criteria are taken into account when comparing the methods. These are semantic segmentation and classification. The extreme values of the comparison results are given below. For 3D classification methods: Multi-view-based approach, (parameter):- (OA):90.1, (mAcc):92.2, Volumetric based approach, (prameter):-, (OA):92, (mAcc):95.54, mAcc:90.6 for Octree Based Approach, (parameter):3.47M for Pointwise MLP methods, (OA):94.5, (mAcc):91.4, for Convolutional Neural Network (CNN) based methods, (parameter): -, (OA):93.6, (mAcc):92.1, for Grap based methods, (parameter):2.24, (OA):93.4, (mAcc):90.7, for Hierarchical Data Structure –based Methods (H.D.S.B.M), (parameter):- (OA):91.8 and (mIoU):93.1 and for Other networks, (parameter):3.2, (OA):93.6 and (mIoU):88.8 are obtained. While the method with the most accurate results according to the OA parameter is PointMLP, VRN-Ensemble is chosen the best method according to the mAcc parameter.

In Table 2, for the 3D semantic segmentation task, an evaluation was made according to the OA, mAcc and mIoU parameters. 3D semantic segmentation results are: Octree Based Approach (OA):81.5, (mIoU):59.2, Pointwise MLP methods (OA):90.2, (mAcc):83.1 and (mIoU):72.2, Convolutional Neural Network (CNN) based methods for (OA): 88.8, (mAcc): 79.1 and (mIoU): 70.6, for Grap based methods (OA): 90, (mAcc): 73.2 and (mIoU): 67.9, Hierarchical Data Structure –based Methods for (H.D.S.B.M), (OA):84.9 and (mIoU):55.6 were obtained. As a result of the evaluation made with the S3DIS database, BAAF-Net received the title of the best method according to the mAcc parameter, while PointTransformer received the title of the best method according to the OA and mIoU parameters.

While evaluating the developed methods, it should be evaluated by considering the field to be applied. Each method use should be evaluated in the area for which it is targeted. If a method is developed on small-scale point cloud data, it may yield undesirable results on large-scale data. The same is true for the opposite situation. From another point of view, if the speed parameter is more important than the accuracy parameter for the targeted application, an optimum accuracy rate and high speed are expected from the method. While the accuracy rate is important, run-time is also important that the methods work fast, be adaptable to real-time systems and appeal to big data. In this sense, among the best methods, GIFT, VoxNet, PointNet, PointNet++, JUSTLOOKUP, RandLA-Net, PointMLP, Flex-ConvNet, FG-Net, Grid-GCN and Deep RBFNet methods can be compared with each other. The methods other than these methods are relatively slow and perform lower than the others in terms of accuracy. GIFT has a low accuracy rate despite its fast operation since the data loss is occurred due to the multi-view approach. VoxNet, on the other hand, has a disadvantage due to voxelization. Since PointNet and PointNet++ are the first methods to directly process point cloud data, they were among the best in terms of speed and accuracy in their own time, but they lost these advantages as new methods were developed. Although JUSTLOOKUP runs 32 times faster than PointNet, it has low mAcc and OA values as given in Table 1. Grid-GCN runs on average 11 times faster than PointNet++ and has higher accuracy rates, but fails to handle millions of points.

Similarly, although Deep RBFNet runs approximately 44 times faster than PointNet++, it fails to process large-scale points. RandLA-Net, which is one of the best methods that can process millions of points in fast times, has been one of the most suitable candidates for real-time systems by processing approximately 1 million points 200 times faster than its peers in its period. Although Flex-ConvNet's ability to process about 7 million points in an average of 4.7 seconds makes it stand out, low accuracy rate of Flex-ConvNet is a disadvantage. Running on average 38.5% faster than RandLA-Net and reducing memory consumption by 8.6%, FG-Net is one of the best candidates for real-time systems. FG-Net gives a satisfactory result not only in terms of speed but also in terms of accuracy. PointMLP, another method that gives good results in terms of speed, gives the best results in terms of classification accuracy, although it is slower than RandLA-Net, Flex-ConvNet and FG-Net. Another purpose of the developed methods is to be robust against adverse conditions, as negative situations such as noise, data loss or a small number of points can be experienced in point cloud data obtained from real environment data. PointASNL, RPNet, MCCNN, DensePoint and AGCNN are among the prominent methods with their noise resistance. SpiderCNN and Continuous Convolutions-based methods can be given as examples of methods whose performances are at acceptable levels despite the low number of points. LP-3DCNN can be given as an example of methods with good performance even with a small number of parameters.

VI. CONCLUSION

Point cloud data processing methods are candidate methods that are widely used in many fields such as computer vision, robotics and autonomous driving, and automation systems in industry and will be developed and used in the future. Most of these methods are based on deep learning, and according to the types of data processing; multi-view, voxel and point cloud, and in terms of task, it has been examined in different categories such as 3D point classification, semantic segmentation, object detection-tracking and optical-flow. General mathematical models of the methods, deep learning architectures and block system structures on which the developed

algorithms are based are given. There is no method that can be evaluated or recommended as the best for every application in semantic segmentation and 3D classification methods. The best method is the one that is suitable for the purpose of the system to be used and shows high performance depending on the size of the data to be processed and the type of system. While speed is important in the real time system, high accuracy rate is important to find someone or object. For this reason, the methods and data collection techniques used should be chosen in accordance with the purpose. Unnecessary data density and operations should be avoided. In this study, deep learning algorithms, learning-based methods developed for the perception of 3D images are also given. The purpose of deep learning algorithms is to extract certain features from the data of objects or to obtain meaningful features from previously undeciphered data, which is assumed to have the same distribution as the input data. The most important result obtained in this study is that there is no best method or best algorithm. The best method and the best algorithm are the ones to be chosen that are most suitable for the purpose. Since each system and application may show different features, it will be the most correct approach to choose according to the system.

REFERENCES

[1] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-D mapping with an RGB-D camera," *IEEE Trans. Robot.*, vol. 30, no. 1, pp. 177–187, Feb. 2014.

[2] P. F. McManamon, *LiDAR Technologies and Systems*. Bellingham, WA, USA: SPIE Press, 2019.

[3] Y. Xu, G. Fang, N. Lv, S. Chen, and J. J. Zou, "Computer vision technology for seam tracking in robotic GTAW and GMAW," *Robot. Comput.-Integr. Manuf.*, vol. 32, pp. 25–36, Apr. 2015.

[4] J. Janai, F. Güney, A. Behl, and A. Geiger, "Computer vision for autonomous vehicles: Problems, datasets and state of the art," *Found. Trends Comput. Graph. Vis.*, vol. 12, no. 1–3, pp. 1–308, 2020.

[5] N. Ayache, "Medical computer vision, virtual reality and robotics," *Image Vis. Comput.*, vol. 13, no. 4, pp. 295–313, 1995.

[6] Q. Chen, "Airborne LiDAR data processing and information extraction," *Photogramm. Eng. Remote Sens.*, vol. 73, no. 2, p. 109, 2007.

[7] R. Behringer, "Registration for outdoor augmented reality applications using computer vision techniques and hybrid sensors," in *Proc. IEEE Virtual Reality*, Mar. 1999, pp. 244–251.

[8] J. P. Dandois and E. C. Ellis, "Remote sensing of vegetation structure using computer vision," *Remote Sens.*, vol. 2, no. 4, pp. 1157–1176, Apr. 2010.

[9] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3D point clouds: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 12, pp. 4338–4364, Dec. 2021.

[10] S. Chen, B. Liu, C. Feng, C. Vallespi-Gonzalez, and C. Wellington, "3D point cloud processing and learning for autonomous driving," 2020, *arXiv:2003.00601*.

[11] X. Yue, B. Wu, S. A. Seshia, K. Keutzer, and A. L. Sangiovanni-Vincentelli, "A LiDAR point cloud generator: From a virtual world to autonomous driving," in *Proc. ACM Int. Conf. Multimedia Retr.*, Jun. 2018, pp. 458–464.

[12] A. Khaloo, D. Lattanzi, A. Jachimowicz, and C. Devaney, "Utilizing UAV and 3D computer vision for visual inspection of a large gravity dam," *Frontiers Built Environ.*, vol. 4, p. 31, Jul. 2018.

[13] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[14] Q. Wang and M.-K. Kim, "Applications of 3D point cloud data in the construction industry: A fifteen-year review from 2004 to 2018," *Adv. Eng. Inform.*, vol. 39, pp. 306–319, Jan. 2019.

[15] X.-F. Han, J. S. Jin, M.-J. Wang, W. Jiang, L. Gao, and L. Xiao, "A review of algorithms for filtering the 3D point cloud," *Signal Process., Image Commun.*, vol. 57, pp. 103–112, Sep. 2017.

[16] A. Nguyen and B. Le, "3D point cloud segmentation: A survey," in *Proc. 6th IEEE Conf. Robot., Autom. Mechatronics (RAM)*, Nov. 2013, pp. 225–230.

[17] B. Bellekens, V. Spruyt, R. Berkvens, R. Penne, and M. Weyn, "A benchmark survey of rigid 3D point cloud registration algorithms," *Int. J. Adv. Intell. Syst.*, vol. 8, pp. 118–127, Jan. 2015.

[18] H. Lu and H. Shi, "Deep learning for 3D point cloud understanding: A survey," 2020, *arXiv:2009.08920*.

[19] W. Liu, J. Sun, W. Li, T. Hu, and P. Wang, "Deep learning on point clouds and its application: A survey," *Sensors*, vol. 19, no. 19, p. 4188, Sep. 2019.

[20] D. Fernandes, A. Silva, R. Névoa, C. Simões, D. Gonzalez, M. Guevara, P. Novais, J. Monteiro, and P. Melo-Pinto, "Point-cloud based 3D object detection and classification methods for self-driving applications: A survey and taxonomy," *Inf. Fusion*, vol. 68, pp. 161–191, 2021.

[21] Z. Zhang, "Microsoft Kinect sensor and its effect," *IEEE MultiMedia*, vol. 19, no. 2, pp. 4–10, Apr. 2012.

[22] R. Jafri and M. M. Khan, "Obstacle detection and avoidance for the visually impaired in indoors environments using Google's project Tango device," in *Proc. Int. Conf. Comput. Helping People Special Needs*. Cham, Switzerland: Springer, Jul. 2016, pp. 179–185.

[23] J. E. Bae, J. W. Kim, H. J. Kim, and H. S. Park, "Determination of golf ball flight based on planar structure sensor," in *Proc. Pacific-Rim Conf. Multimedia*. Berlin, Germany: Springer, Dec. 2009, pp. 344–355.

[24] L. Keselman, J. I. Woodfill, A. Grunnet-Jepsen, and A. Bhowmik, "Intel(R) RealSense(TM) stereoscopic depth cameras," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 1–10.

[25] *3D Computer Aided Design*. Accessed: May 27, 2022. [Online]. Available: https://www.autodesk.com/solutions/3d-cad-software

[26] [Online]. Available: https://www.daz3d.com/

[27] 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.

[28] 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*.

[29] X. Wang, B. Zhou, Y. Shi, X. Chen, Q. Zhao, and K. Xu, "Shape2Motion: Joint analysis of motion parts and attributes from 3D shapes," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 8876–8884.

[30] M. A. Uy, Q.-H. Pham, B.-S. Hua, T. Nguyen, and S.-K. Yeung, "Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1588–1597.

[31] Y. Gao, A. Liu, W. Nie, Y. Su, Q. Dai, F. Chen, and G. Zhu, "Shrec'15 track: 3D object retrieval with multimodal views," in *Proc. Eurograph. Workshop 3D Object Retr.*, 2015, pp. 129–136.

[32] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *Proc. Eur. Conf. Comput. Vis.*, Berlin, Germany: Springer, Oct. 2012, pp. 746–760.

[33] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese, "3D semantic parsing of large-scale indoor spaces," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1534–1543.

[34] J. Xiao, A. Owens, and A. Torralba, "SUN3D: A database of big spaces reconstructed using SfM and object labels," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 1625–1632.

[35] B.-S. Hua, Q.-H. Pham, D. T. Nguyen, M.-K. Tran, L.-F. Yu, and S.-K. Yeung, "SceneNN: A scene meshes dataset with annotations," in *Proc. 4th Int. Conf. 3D Vis. (3DV)*, Oct. 2016, pp. 92–101.

[36] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Niessner, "ScanNet: Richly-annotated 3D reconstructions of indoor scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5828–5839.

[37] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Nießner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3D: Learning from RGB-D data in indoor environments," 2017, *arXiv:1709.06158*.

[38] A. Zeng, S. Song, M. Niessner, M. Fisher, J. Xiao, and T. Funkhouser, "3DMatch: Learning local geometric descriptors from RGB-D reconstructions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1802–1811.

[39] J. Valentin, A. Dai, M. Niessner, P. Kohli, P. Torr, S. Izadi, and C. Keskin, "Learning to navigate the energy landscape," in *Proc. 4th Int. Conf. 3D Vis. (3DV)*, Oct. 2016, pp. 323–332.

[40] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon, "Scene coordinate regression forests for camera relocalization in RGB-D images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 2930–2937.

[41] K. Lai, L. Bo, and D. Fox, "Unsupervised feature learning for 3D scene labeling," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2014, pp. 3050–3057.

[42] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3354–3361.

[43] B. Vallet, M. Brédif, A. Serna, B. Marcotegui, and N. Paparoditis, "TerraMobilita/iQmulus urban point cloud analysis benchmark," *Comput. Graph.*, vol. 49, pp. 126–133, Jun. 2015.

[44] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 year, 1000 km: The Oxford robotcar dataset," *IJ Robot. Res.*, vol. 36, no. 1, pp. 3–15, 2016.

[45] N. Carlevaris-Bianco, A. K. Ushani, and R. M. Eustice, "University of Michigan north campus long-term vision and LiDAR dataset," *Int. J. Robot. Res.*, vol. 35, no. 9, pp. 1023–1035, Aug. 2016.

[46] T. Hackel, N. Savinov, L. Ladicky, J. D. Wegner, K. Schindler, and M. Pollefeys, "Semantic3D.Net: A new large-scale point cloud classification benchmark," 2017, *arXiv:1704.03847*.

[47] Y. Chen, J. Wang, J. Li, C. Lu, Z. Luo, H. Xue, and C. Wang, "LiDAR-video driving dataset: Learning driving policies effectively," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5870–5878.

[48] X. Song, P. Wang, D. Zhou, R. Zhu, C. Guan, Y. Dai, H. Su, H. Li, and R. Yang, "ApolloCar3D: A large 3D car instance understanding benchmark for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5452–5462.

[49] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuScenes: A multimodal dataset for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11621–11631.

[50] J. Xue, J. Fang, T. Li, B. Zhang, P. Zhang, Z. Ye, and J. Dou, "BLVD: Building a large-scale 5D semantics benchmark for autonomous driving," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 6685–6691.

[51] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, and S. S. Iyengar, "A survey on deep learning: Algorithms, techniques, and applications," *ACM Comput. Surv.*, vol. 51, no. 5, pp. 1–36, 2018.

[52] F. Agostinelli, M. Hoffman, P. Sadowski, and P. Baldi, "Learning activation functions to improve deep neural networks," 2014, *arXiv:1412.6830*.

[53] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," 2016, *arXiv:1603.07285*.

[54] T. Chaperon and F. Goulette, "Extracting cylinders in full 3D data using a random sampling method and the Gaussian image," in *Proc. Vis. Modeling Vis. Conf.*, Nov. 2001, pp. 1–9.

[55] C. Moenning and N. A. Dodgson, "Fast marching farthest point sampling," Univ. Cambridge, Comput. Lab., Cambridge, MA, USA, Tech. Rep., UCAM-CL-TR-562, 2003.

[56] W. A. McNeely, K. D. Puterbaugh, and J. J. Troy, "Six degree-of-freedom haptic rendering using Voxel sampling," in *Proc. ACM SIGGRAPH*, 2005, p. 42.

[57] K. Fukunaga and P. M. Narendra, "A branch and bound algorithm for computing k-nearest neighbors," *IEEE Trans. Comput.*, vol. C-24, no. 7, pp. 750–753, Jul. 1975.

[58] 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.

[59] S. Bai, X. Bai, Z. Zhou, Z. Zhang, and L. J. Latecki, "GIFT: A real-time and scalable 3D shape search engine," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 5023–5032.

[60] T. Yu, J. Meng, and J. Yuan, "Multi-view harmonized bilinear network for 3D object recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 186–194.

[61] Z. Cao, Q. Huang, and R. Karthik, "3D object classification via spherical projections," in *Proc. Int. Conf. 3D Vis. (3DV)*, Oct. 2017, pp. 566–574.

[62] Y. Li, S. Pirk, H. Su, C. R. Qi, and L. J. Guibas, "FPNN: Field probing neural networks for 3D data," 2016, *arXiv:1605.06240*.

[63] 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.

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

[65] G. Riegler, A. O. Ulusoy, and A. Geiger, "OctNet: Learning deep 3D representations at high resolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3577–3586.

[66] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong, "O-CNN: Octree-based convolutional neural networks for 3D shape analysis," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 1–11, 2017.

[67] T. Le and Y. Duan, "PointGrid: A deep network for 3D shape understanding," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9204–9214.

[68] 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)*, Jul. 2017, pp. 652–660.

[69] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," 2017, *arXiv:1706.02413*.

[70] M. Jiang, Y. Wu, T. Zhao, Z. Zhao, and C. Lu, "PointSIFT: A SIFT-like network module for 3D point cloud semantic segmentation," 2018, *arXiv:1807.00652*.

[71] F. Engelmann, T. Kontogianni, J. Schult, and B. Leibe, "Know what your neighbors do: 3D semantic segmentation of point clouds," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 1–9.

[72] M. Joseph-Rivlin, A. Zvirin, and R. Kimmel, "Momen^{e}t: Flavor the moments in learning to classify shapes," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops*, Oct. 2019, pp. 1–10.

[73] J. Yang, Q. Zhang, B. Ni, L. Li, J. Liu, M. Zhou, and Q. Tian, "Modeling point clouds with self-attention and Gumbel subset sampling," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3323–3332.

[74] L.-Z. Chen, X.-Y. Li, D.-P. Fan, K. Wang, S.-P. Lu, and M.-M. Cheng, "LSANet: Feature learning on point sets by local spatial aware layer," 2019, *arXiv:1905.05442*.

[75] H. Zhao, L. Jiang, C.-W. Fu, and J. Jia, "PointWeb: Enhancing local neighborhood features for point cloud processing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5565–5573.

[76] Y. Duan, Y. Zheng, J. Lu, J. Zhou, and Q. Tian, "Structural relational reasoning of point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 949–958.

[77] H. Lin, Z. Xiao, Y. Tan, H. Chao, and S. Ding, "Justlookup: One millisecond deep feature extraction for point clouds by lookup tables," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2019, pp. 326–331.

[78] Z. Zhang, B.-S. Hua, and S.-K. Yeung, "ShellNet: Efficient point cloud convolutional neural networks using concentric shells statistics," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1607–1616.

[79] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, "RandLA-Net: Efficient semantic segmentation of large-scale point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11108–11117.

[80] X. Yan, C. Zheng, Z. Li, S. Wang, and S. Cui, "PointASNL: Robust point clouds processing using nonlocal neural networks with adaptive sampling," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 5589–5598.

[81] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9297–9307.

[82] H. Ran, W. Zhuo, J. Liu, and L. Lu, "Learning inner-group relations on point clouds," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 15477–15487.

[83] H. Zhao, L. Jiang, J. Jia, P. Torr, and V. Koltun, "Point transformer," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 16259–16268.

[84] S. Fan, Q. Dong, F. Zhu, Y. Lv, P. Ye, and F.-Y. Wang, "SCF-Net: Learning spatial contextual features for large-scale point cloud segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 14504–14513.

[85] S. Qiu, S. Anwar, and N. Barnes, "Semantic segmentation for real point cloud scenes via bilateral augmentation and adaptive fusion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 1757–1767.

[86] X. Ma, C. Qin, H. You, H. Ran, and Y. Fu, "Rethinking network design and local geometry in point cloud: A simple residual MLP framework," 2022, *arXiv:2202.07123*.

[87] B.-S. Hua, M.-K. Tran, and S.-K. Yeung, "Pointwise convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 984–993.

[88] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "PointCNN: Convolution on χ-transformed points," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 820–830.

[89] M. Eitz, R. Richter, T. Boubekeur, K. Hildebrand, and M. Alexa, "Sketch-based shape retrieval," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 1–10, 2012.

[90] D. Ha and D. Eck, "A neural representation of sketch drawings," 2017, *arXiv:1704.03477*.

[91] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*.

[92] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009.

[93] F. Groh, P. Wieschollek, and H. P. A. Lensch, "Flex-convolution (million-scale point-cloud learning beyond grid-worlds)," 2018, *arXiv:1803.07289*.

[94] M. Atzmon, H. Maron, and Y. Lipman, "Point convolutional neural networks by extension operators," 2018, *arXiv:1803.10091*.

[95] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao, "SpiderCNN: Deep learning on point sets with parameterized convolutional filters," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 87–102.

[96] P. Hermosilla, T. Ritschel, P. P. Vázquez, and T. Ropinski, "Monte Carlo convolution for learning on non-uniformly sampled point clouds," *ACM Trans. Graph.*, vol. 37, no. 6, pp. 1–12, 2018.

[97] W. Wu, Z. Qi, and L. Fuxin, "PointConv: Deep convolutional networks on 3D point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9621–9630.

[98] S. Lan, R. Yu, G. Yu, and L. S. Davis, "Modeling local geometric structure of 3D point clouds using geo-CNN," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 998–1008.

[99] S. Kumawat and S. Raman, "LP-3DCNN: Unveiling local phase in 3D convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4903–4912.

[100] A. Komarichev, Z. Zhong, and J. Hua, "A-CNN: Annularly convolutional neural networks on point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 7421–7430.

[101] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. Guibas, "KPConv: Flexible and deformable convolution for point clouds," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6411–6420.

[102] X. Roynard, J.-E. Deschaud, and F. Goulette, "Paris-Lille-3D: A large and high-quality ground-truth urban point cloud dataset for automatic segmentation and classification," *Int. J. Robot. Res.*, vol. 37, no. 6, pp. 545–557, 2018.

[103] Y. Liu, B. Fan, S. Xiang, and C. Pan, "Relation-shape convolutional neural network for point cloud analysis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 8895–8904.

[104] Y. Rao, J. Lu, and J. Zhou, "Spherical fractal convolutional neural networks for point cloud recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 452–460.

[105] J. Mao, X. Wang, and H. Li, "Interpolated convolutional networks for 3D point cloud understanding," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1578–1587.

[106] Y. Liu, B. Fan, G. Meng, J. Lu, S. Xiang, and C. Pan, "DensePoint: Learning densely contextual representation for efficient point cloud processing," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 5239–5248.

[107] A. Boulch, "ConvPoint: Continuous convolutions for point cloud processing," *Comput. Graph.*, vol. 88, pp. 24–34, May 2020.

[108] T. Xiang, C. Zhang, Y. Song, J. Yu, and W. Cai, "Walk in the cloud: Learning curves for point clouds shape analysis," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 915–924.

[109] K. Liu, Z. Gao, F. Lin, and B. M. Chen, "FG-Net: Fast large-scale LiDAR point clouds understanding network leveraging correlated feature mining and geometric-aware modelling," 2020, *arXiv:2012.09439*.

[110] K. Mo, S. Zhu, A. X. Chang, L. Yi, S. Tripathi, L. J. Guibas, and H. Su, "PartNet: A large-scale benchmark for fine-grained and hierarchical Part—Level 3D object understanding," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 909–918.

[111] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3693–3702.

[112] Y. Shen, C. Feng, Y. Yang, and D. Tian, "Mining point cloud local structures by kernel correlation and graph pooling," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4548–4557.

[113] 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, Oct. 2019.

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

[115] R. Li, S. Wang, F. Zhu, and J. Huang, "Adaptive graph convolutional neural networks," in *Proc. AAAI Conf. Artif. Intell.*, Apr. 2018, vol. 32, no. 1, pp. 1–8.

[116] C. Wang, B. Samari, and K. Siddiqi, "Local spectral graph convolution for point set feature learning," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 52–66.

[117] G. Te, W. Hu, A. Zheng, and Z. Guo, "RGCNN: Regularized graph CNN for point cloud segmentation," in *Proc. 26th ACM Int. Conf. Multimedia*, Oct. 2018, pp. 746–754.

[118] K. Zhiheng and L. Ning, "PyramNet: Point cloud pyramid attention network and graph embedding module for classification and segmentation," 2019, *arXiv:1906.03299*.

[119] J. Liu, B. Ni, C. Li, J. Yang, and Q. Tian, "Dynamic points agglomeration for hierarchical point sets learning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 7546–7555.

[120] Y. Ma, Y. Guo, H. Liu, Y. Lei, and G. Wen, "Global context reasoning for semantic segmentation of 3D point clouds," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2020, pp. 2931–2940.

[121] Q. Xu, X. Sun, C.-Y. Wu, P. Wang, and U. Neumann, "Grid-GCN for fast and scalable point cloud learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 5661–5670.

[122] K. Zhou, Q. Hou, R. Wang, and B. Guo, "Real-time KD-tree construction on graphics hardware," *ACM Trans. Graph.*, vol. 27, no. 5, pp. 1–11, Dec. 2008.

[123] 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)*, Oct. 2017, pp. 863–872.

[124] W. Zeng and T. Gevers, "3DContextNet: K-d tree guided hierarchical learning of point clouds using local and global contextual cues," in *Proc. Eur. Conf. Comput. Vis. (ECCV) Workshops*, 2018, pp. 1–16.

[125] 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.*, Jun. 2018, pp. 9397–9406.

[126] H. You, Y. Feng, R. Ji, and Y. Gao, "PVNet: A joint convolutional network of point cloud and multi-view for 3D shape recognition," in *Proc. 26th ACM Int. Conf. Multimedia*, Oct. 2018, pp. 1310–1318.

[127] H. You, Y. Feng, X. Zhao, C. Zou, R. Ji, and Y. Gao, "PVRNet: Point-view relation neural network for 3D shape recognition," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 9119–9126.

[128] Y. Zhao, T. Birdal, H. Deng, and F. Tombari, "3D point capsule networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1009–1018.

[129] W. Chen, X. Han, G. Li, C. Chen, J. Xing, Y. Zhao, and H. Li, "Deep RBFNet: Point cloud feature learning using radial basis functions," 2018, *arXiv:1812.04302*.

[130] X. Liu, Z. Han, Y.-S. Liu, and M. Zwicker, "Point2Sequence: Learning the shape representation of 3D point clouds with an attention-based sequence to sequence network," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 8778–8785.

[131] P. Wu, C. Chen, J. Yi, and D. Metaxas, "Point cloud processing via recurrent set encoding," in *Proc. AAAI Conf. Artif. Intell.*, Jul. 2019, vol. 33, no. 1, pp. 5441–5449.

[132] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D object detection network for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1907–1915.

[133] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, "Joint 3D proposal generation and object detection from view aggregation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 1–8.

[134] 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.

[135] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun, "Multi-task multi-sensor fusion for 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 7345–7353.

[136] D. Feng, C. Haase-Schutz, L. Rosenbaum, H. Hertlein, C. Glaser, F. Timm, W. Wiesbeck, and K. Dietmayer, "Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 3, pp. 1341–1360, Mar. 2021.

[137] Y. Zeng, Y. Hu, S. Liu, J. Ye, Y. Han, X. Li, and N. Sun, "RT3D: Real-time 3-D vehicle detection in LiDAR point cloud for autonomous driving," *IEEE Robot. Automat. Lett.*, vol. 3, no. 4, pp. 3434–3440, Oct. 2018.

[138] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, "IPOD: Intensive point-based object detector for point cloud," 2018, *arXiv:1812.05276*.

[139] S. Shi, X. Wang, and H. Li, "PointRCNN: 3D object proposal generation and detection from point cloud," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 770–779.

[140] J. Zarzar, S. Giancola, and B. Ghanem, "PointRGCN: Graph convolution networks for 3D vehicles detection refinement," 2019, *arXiv:1911.12236*.

[141] S. Vora, A. H. Lang, B. Helou, and O. Beijbom, "PointPainting: Sequential fusion for 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 4604–4612.

[142] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, "STD: Sparse-to-dense 3D object detector for point cloud," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1951–1960.

[143] 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/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 918–927.

[144] X. Zhao, Z. Liu, R. Hu, and K. Huang, "3D object detection using scale invariant and feature reweighting networks," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 9267–9274.

[145] D. Xu, D. Anguelov, and A. Jain, "PointFusion: Deep sensor fusion for 3D bounding box estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 244–253.

[146] K. Shin, Y. P. Kwon, and M. Tomizuka, "RoarNet: A robust 3D object detection based on region approximation refinement," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2019, pp. 2510–2515.

[147] Z. Wang and K. Jia, "Frustum ConvNet: Sliding frustums to aggregate local point-wise features for amodal 3D object detection," 2019, *arXiv:1903.01864*.

[148] J. Lehner, A. Mitterecker, T. Adler, M. Hofmarcher, B. Nessler, and S. Hochreiter, "Patch refinement—Localized 3D object detection," 2019, *arXiv:1910.04093*.

[149] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li, "PV-RCNN: Point-voxel feature set abstraction for 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10529–10538.

[150] D. Zhou, J. Fang, X. Song, C. Guan, J. Yin, Y. Dai, and R. Yang, "IoU loss for 2D/3D object detection," in *Proc. Int. Conf. 3D Vis. (3DV)*, Sep. 2019, pp. 85–94.

[151] Y. Chen, S. Liu, X. Shen, and J. Jia, "Fast point R-CNN," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9775–9784.

[152] Z. Ding, X. Han, and M. Niethammer, "VoteNet: A deep learning label fusion method for multi-atlas segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.*, Cham, Switzerland: Springer, Oct. 2019, pp. 202–210.

[153] C. R. Qi, X. Chen, O. Litany, and L. J. Guibas, "ImVoteNet: Boosting 3D object detection in point clouds with image votes," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 4404–4413.

[154] S. Shi, Z. Wang, J. Shi, X. Wang, and H. Li, "From points to parts: 3D object detection from point cloud with Part—Aware and Part—Aggregation network," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 8, pp. 2647–2664, Aug. 2021.

[155] B. Yang, W. Luo, and R. Urtasun, "PIXOR: Real-time 3D object detection from point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7652–7660.

[156] B. Yang, M. Liang, and R. Urtasun, "HDNET: Exploiting HD maps for 3D object detection," in *Proc. Conf. Robot Learn.*, Oct. 2018, pp. 146–155.

[157] J. Beltrán, C. Guindel, F. M. Moreno, D. Cruzado, F. Garcia, and A. De La Escalera, "BirdNet: A 3D object detection framework from LiDAR information," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 3517–3523.

[158] B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3D LiDAR using fully convolutional network," 2016, *arXiv:1608.07916*.

[159] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner, "Vote3Deep: Fast object detection in 3D point clouds using efficient convolutional neural networks," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 1355–1361.

[160] X. Li, J. E. Guivant, N. Kwok, and Y. Xu, "3D backbone network for 3D object detection," 2019, *arXiv:1901.08373*.

[161] Y. Zhou and O. Tuzel, "VoxelNet: End-to-end learning for point cloud based 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4490–4499.

[162] V. A. Sindagi, Y. Zhou, and O. Tuzel, "MVX-Net: Multimodal VoxelNet for 3D object detection," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 7276–7282.

[163] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast encoders for object detection from point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12697–12705.

[164] C. He, H. Zeng, J. Huang, X.-S. Hua, and L. Zhang, "Structure aware single-stage 3D object detection from point cloud," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11873–11882.

[165] Z. Yang, Y. Sun, S. Liu, and J. Jia, "3DSSD: Point-based 3D single stage object detector," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11040–11048.

[166] G. P. Meyer, A. Laddha, E. Kee, C. Vallespi-Gonzalez, and C. K. Wellington, "LaserNet: An efficient probabilistic 3D object detector for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12677–12686.

[167] G. P. Meyer, J. Charland, D. Hegde, A. Laddha, and C. Vallespi-Gonzalez, "Sensor fusion for joint 3D object detection and semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2019, pp. 1–8.

[168] Q. Chen, L. Sun, Z. Wang, K. Jia, and A. Yuille, "Object as hotspots: An anchor-free 3D object detection approach via firing of hotspots," in *Proc. Eur. Conf. Comput. Vis.*, Switzerland: Springer, Aug. 2020, pp. 68–84.

[169] W. Shi and R. Rajkumar, "Point-GNN: Graph neural network for 3D object detection in a point cloud," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1711–1719.

[170] S. Giancola, J. Zarzar, and B. Ghanem, "Leveraging shape completion for 3D Siamese tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1359–1368.

[171] M. Simon, K. Amende, A. Kraus, J. Honer, T. Samann, H. Kaulbersch, S. Milz, and H. M. Gross, "Complexer-YOLO: Real-time 3D object detection and tracking on semantic point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2019, pp. 1–10.

[172] X. Liu, C. R. Qi, and L. J. Guibas, "FlowNet3D: Learning scene flow in 3D point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 529–537.

[173] Z. Wang, S. Li, H. Howard-Jenkins, V. A. Prisacariu, and M. Chen, "FlowNet3D++: Geometric losses for deep scene flow estimation," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2020, pp. 91–98.

[174] F. Duffhauss and S. A. Baur, "PillarFlowNet: A real-time deep multitask network for LiDAR-based 3D object detection and scene flow estimation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 10734–10741.

[175] X. Liu, M. Yan, and J. Bohg, "MeteorNet: Deep learning on dynamic 3D point cloud sequences," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9246–9255.

[176] X. Liang and Z. Fu, "MHNet: Multiscale hierarchical network for 3D point cloud semantic segmentation," *IEEE Access*, vol. 7, pp. 173999–174012, 2019.

ABDURRAHMAN HAZER received the B.S. degree in electrical and electronics engineering from Abant İzzet Baysal University, in 2014, and the M.S. degree in electrical and electronics engineering from Ankara Yıldırım Beyazıt University, Ankara, Turkey, in 2019, where he is currently pursuing the Ph.D. degree in computer engineering. His research interests include computer vision, Fourier optic processing, signal processing, deep learning, and cryptography.

REMZI YILDIRIM received the B.S. and M.S. degrees in electronics and computer engineering from Gazi University, Ankara, Turkey, in 1988 and 1993, respectively, and the Ph.D. degree in electronics from Erciyes University, Kayseri, Turkey, in 1996. From 1999 to 2002, he was a Visiting Scholar with the Massachusetts Institute of Technology, Boston, MA, USA. From 2004 to 2005, he was with Liverpool University, U.K. He is currently a Professor with the Department of Computer Engineering, Ankara Yıldırım Beyazıt University, Ankara. He has published more than 100 journals and conference papers and 21 books. His research interests include optoelectronics, cyber-physical systems, and model checking.