# Deep Learning for LiDAR Point Clouds in Autonomous Driving: A Review

Ying Li, *Graduate Student Member, IEEE*, Lingfei Ma, *Graduate Student Member, IEEE*, Zilong Zhong, *Member, IEEE*, Fei Liu, Michael A. Chapman, *Senior Member, IEEE*, Dongpu Cao, *Senior Member, IEEE*, and Jonathan Li, *Senior Member, IEEE*

***Abstract*— Recently, the advancement of deep learning (DL) in discriminative feature learning from 3-D LiDAR data has led to rapid development in the field of autonomous driving. However, automated processing uneven, unstructured, noisy, and massive 3-D point clouds are a challenging and tedious task. In this article, we provide a systematic review of existing compelling DL architectures applied in LiDAR point clouds, detailing for specific tasks in autonomous driving, such as segmentation, detection, and classification. Although several published research articles focus on specific topics in computer vision for autonomous vehicles, to date, no general survey on DL applied in LiDAR point clouds for autonomous vehicles exists. Thus, the goal of this article is to narrow the gap in this topic. More than 140 key contributions in the recent five years are summarized in this survey, including the milestone 3-D deep architectures, the remarkable DL applications in 3-D semantic segmentation, object detection, and classification; specific data sets, evaluation metrics, and the state-of-the-art performance. Finally, we conclude the remaining challenges and future researches.**

***Index Terms*— Autonomous driving, deep learning (DL), LiDAR, object classification, object detection, point clouds, semantic segmentation.**

## I. Introduction

ACCURATE environment perception and precise localization are crucial requirements for reliable navigation, information decision, and safely driving of autonomous vehicles (AVs) in complex dynamic environments [1], [2]. These two tasks need to acquire and process highly accurate and information-rich data of real-world environments [3]. To obtain such data, multiple sensors, such as LiDAR and digital cameras [4], are equipped on AVs or mapping vehicles to collect and extract target context. Traditionally, image data captured by the digital camera, featured with 2-D appearance-based representation, low cost, and high efficiency, is the most commonly used data in perception tasks [5]. However, image data lack of 3-D geo-referenced information [6]. Thus, the dense, geo-referenced, and accurate 3-D point cloud data collected by LiDAR are exploited. Besides, LiDAR is not sensitive to the variations of lighting conditions and can work under day and night, even with glare and shadows [7].

The application of LiDAR point clouds for AVs can be described in two aspects: 1) real-time environment perception and processing for scene understanding and object detection [8] and 2) high-definition maps and urban models generation and construction for reliable localization and referencing [2]. These applications have some similar tasks, which can be roughly divided into three types: 3-D point cloud segmentation, 3-D object detection and localization, and 3-D object classification and recognition. Such a technique has led to an increasing and urgent requirement for automatic analysis of 3-D point clouds [9] for AVs.

Driven by the breakthroughs brought by deep learning (DL) techniques and the accessibility of 3-D point clouds, the 3-D DL frameworks have been investigated based on the extension of 2-D DL architectures to 3-D data with a notable string of empirical successes. These frameworks can be applied to several tasks specifically for AVs, such as semantic segmentation and scene understanding [10]–[12], object detection [13], [14], and classification [10], [15], [16]. Thus, we provide a systematic survey in this article, which focuses explicitly on framing the LiDAR point clouds in segmentation, detection, and classification tasks for autonomous driving using DL techniques.

Several related surveys based on DL have been published in recent years. The basic and comprehensive knowledge of DL is described in detail in [17] and [18]. These surveys normally focus on reviewing DL applications in visual data [19], [20] and remote sensing imagery [21], [22]. Some are targeted at more specific tasks, such as object detection [23], [24], semantic segmentation [25], and recognition [26]. Although DL in 3-D data has been surveyed in [27]–[29], these 3-D data are mainly 3-D computer-aided design (CAD) models [30]. In [1], challenges, data sets, and methods in computer vision

Manuscript received August 15, 2019; revised March 15, 2020 and June 25, 2020; accepted August 8, 2020. This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under Grant 50503-10284. *(Corresponding authors: Dongpu Cao; Jonathan Li.)*

Ying Li, Lingfei Ma, and Zilong Zhong are with the Department of Geography and Environmental Management, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: y2424li@uwaterloo.ca; l53ma@uwaterloo.ca; z26zhong@uwaterloo.ca).

Fei Liu is with Xilinx Technology Beijing, Ltd., Beijing 100083, China (e-mail: fledaliu@xilinx.com).

Michael A. Chapman is with the Department of Civil Engineering, Ryerson University, Toronto, ON M5B 2K3, Canada (e-mail: mchapman@ryerson.ca).

Dongpu Cao is with the Waterloo Cognitive Autonomous Driving Laboratory, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: dongpu.cao@uwaterloo.ca).

Jonathan Li is with the Department of Geography and Environmental Management, University of Waterloo, Waterloo, ON N2L 3G1, Canada, and also with the Department of Systems Design Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: junli@uwaterloo.ca).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2020.3015992

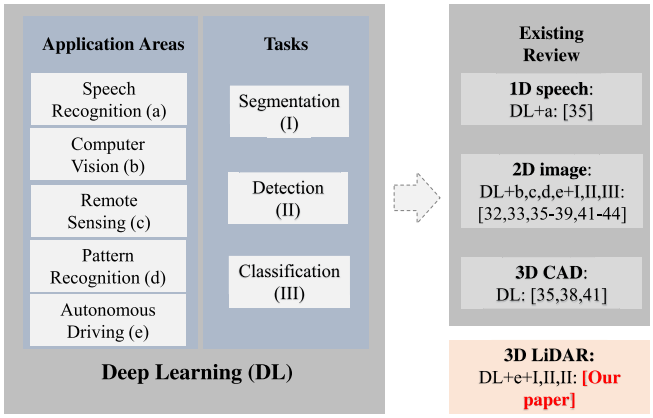



Fig. 1. Existing review article related to DL and their application with different tasks. We summarize that our article is the first one to survey the application of LiDAR point clouds in segmentation, detection, and classification tasks for autonomous driving using DL techniques.

for AVs are reviewed. However, DL applications in LiDAR point cloud data have not been comprehensively reviewed and analyzed. We summarize these surveys related to DL in Fig. 1.

There also have several surveys published for LiDAR point clouds. In [31]–[34], 3-D road object segmentation, detection, and classification from mobile LiDAR point clouds are introduced, but they are focusing on general methods not specific for DL models. In [35], comprehensive 3-D descriptors are analyzed. In [36] and [37], approaches of 3-D object detection applied for autonomous driving are concluded. However, DL models applied in these tasks have not been comprehensively analyzed. Thus, the goal of this article is to provide a systematic review of DL using LiDAR point clouds in the field of autonomous driving for specific tasks, such as segmentation, detection/localization, and classification.

The main contributions of our work can be summarized as follows.

1) An in-depth and organized survey of the milestone 3-D deep models and a comprehensive survey of DL methods aimed at tasks, such as semantic segmentation, object detection/localization, and object classification/recognition in AVs, their origins, and their contributions.
2) A comprehensive survey of existing LiDAR data sets that can be exploited in training and evaluating DL models for AVs.
3) A detailed introduction for quantitative evaluation metrics and performance comparison for semantic segmentation, object detection, and object classification.
4) A list of the remaining challenges and future researches that help to advance the development of DL in the field of autonomous driving.

The remainder of this article is organized as follows. The tasks in autonomous driving and the challenges of DL using LiDAR point cloud data are introduced in Section II. A summary of existing LiDAR point cloud data sets and evaluation metrics are described in Section III. Then, the milestone 3-D deep models with four data representations of LiDAR point clouds are described in Section IV. The DL applications in semantic segmentation, object detection/localization, and classification/recognition for AVs based on LiDAR point clouds are reviewed and discussed in Section V. Section VI proposes a list of the remaining challenges for future researches. We finally conclude this article in Section VII.

## II. Tasks and Challenges

### A. Tasks

In the perception module of autonomous vehicles, semantic segmentation, object detection, object localization, and object classification/recognition constitute the foundation for reliable navigation and accurate decision [38]. These tasks are described as follows, respectively.

1) 3-D point cloud semantic segmentation. Point cloud semantic segmentation is the process to cluster the input data into several homogeneous regions, where points in the same region have the identical attributes [39]. Each input point is predicted with a semantic label, such as ground, tree, and building. This task can be concluded as given a set of ordered 3-D points $X = \{x_1, x_2, x_i, \ldots, x_n\}$ with $x_i \in R^3$ and a candidate label set $Y = \{y_1, y_2, \ldots, y_k\}$, assign each input point $x_i$ with one of the $k$ semantic labels [40]. Segmentation results can further support object detection and classification, as shown in Fig. 2(a).
2) 3-D object detection/localization. Given an arbitrary point cloud data, the goal of 3-D object detection is to detect and locate the instances of predefined categories [e.g., cars, pedestrians, and cyclists, as shown in Fig. 2(b)], and return their geometric 3-D location, orientation, and semantic instance label [41]. Such information can be represented coarsely using a 3-D bounding box which is tightly bounding the detected object [13], [42], [42]. This box is commonly represented as $(x, y, z, h, w, l, \theta, c)$, where $(x, y, z)$ denotes the object (bounding box) center position, $(h, w, l)$ represents the bounding box size with width, length, and height, and $\theta$ is the object orientation. The orientation refers to the rigid transformation that aligns the detected object to its instance in the scene, which are the translations in each of the of $x$-, $y$-, and $z$-directions and a rotation about each of these three axes [43], [44]. $c$ represents the semantic label of this bounding box (object).
3) 3-D object classification/recognition. Given several groups of point clouds, the objectiveness of classification/recognition is to determine the category [e.g., mug, table, or car, as shown in Fig. 2(c)] the group points belong to. The problem of 3-D object classification can be defined as: given a set of 3-D ordered points $X = \{x_1, x_2, x_i, \ldots, x_n\}$ with $x_i \in R^3$ and a candidate label set $Y = \{y_1, y_2, \ldots, y_k\}$, assign the whole point set $X$ with one of the $k$ labels [45].

### B. Challenges and Problems

In order to segment, detect, and classify the general objects using DL for AVs with robust and discriminative performance, several challenges and problems must be addressed, as shown in Fig. 2. The variation of sensing conditions and unconstrained environments results in challenges on data.

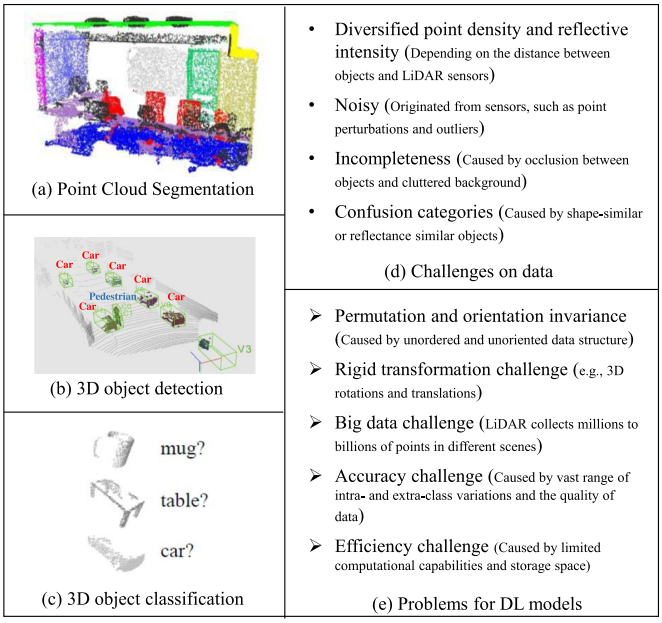



Fig. 2. Tasks and challenges related to DL-based applications on 3-D point clouds. (a) Point cloud segmentation [10]. (b) 3-D object detection [41]. (c) 3-D object classification [10]. (d) Challenges on LiDAR point clouds. and (e) Problems for DL models.

The irregular data format and requirements for both accuracy and efficiency pose the problems that DL models need to solve.

*1) Challenges on LiDAR Point Clouds:* Changes in sensing conditions and unconstrained environments have dramatic impacts on the appearances of objects. In particular, the objects captured at different scenes or instances and even for the same scene, the scanning times, locations, weather conditions, sensor types, sensing distances, and backgrounds are all brought about differences. All these conditions produce significant variations for both intraclass and extra-class objects in LiDAR point clouds.

1) Diversified point density and reflective intensity. Due to the scanning mode of LiDAR, the density and the intensity for objects vary a lot. The distribution of these two characteristics highly depends on the distances between objects and LiDAR sensors [46]–[48]. Besides, the ability of the LiDAR sensors, the time constraints of scanning and needed resolution also affect their distribution and intensity.
2) Noisy. All sensors are noisy. There are a few types of noise that include point perturbations and outliers [49]. It means that a point has some probability of being within a sphere of a certain radius around the place it was sampled (perturbations), or it may appear in a random position in space [50].
3) Incompleteness. Point clouds obtained by LiDAR are commonly incomplete [51]. This mainly results from the occlusion between objects [50], cluttered background in urban scenes [46], [49], and unsatisfactory material surface reflectivity. Such problems are severe in real-time capturing of moving objects, which exist large gaping holes and severe undersampling.
4) Confusion categories. In a natural environment, shape-similar or reflectance similar objects have interference in object detection and classification. For example, some man-made objects, such as commercial billboards, have similar shapes and reflectance with traffic signs.

*2) Problems for 3-D DL Models:* The irregular data format and the requirements for accuracy and efficiency from tasks bring some new challenges for DL models. A discriminate and general-purpose 3-D DL model should solve the following problems when designing and constructing its framework.

1) Permutation and orientation invariance. Compared with 2-D grid pixels, the LiDAR point clouds are a set of points with irregular order and no specific orientation [52]. Within the same group of $N$ points, the network should feed N! permutations in order to be invariant. Besides, the orientation of point sets is missing, which poses a great challenge for object pattern recognition [53].
2) Rigid transformation challenge. There exist various rigid transformations among point sets, such as 3-D rotations and 3-D translations. These transformations should not affect the performance of networks [12], [52].
3) Big data challenge. LiDAR collects millions to billions of points in different urban or rural environments with natural scenes [49]. For example, in the Kitti data set [54], each frame captured by 3-D Velodyne laser scanners contains 100k points. The smallest collected scene has 114 frames, which has more than 10 million points. Such amounts of data bring difficulties in data storage and processing.
4) Accuracy challenge. Accurate perception of road objects is crucial for AVs. However, the variation for both intraclass and extra-class objects and the quality of data poses challenges for accuracy. For example, objects in the same category have a set of different instances, in terms of various material, shape, and size. Besides, the model should be robust to the unevenly distributed, sparse, and missing data.
5) Efficiency challenge. Compared with 2-D images, processing a large quantity number of point clouds produces high computation complexity and time costs. Besides, the computation devices on AVs have limited computational capabilities and storage space [55]. Thus, an efficient and scalable deep network model is critical.

## III. Data Sets and Evaluation Metrics

### A. Data Sets

Data sets pave the way toward the rapid development of 3-D data application and exploitation using DL networks. There are two roles of reliable data sets: one for providing a comparison for competing algorithms, another for pushing the fields toward more complex, and challenging tasks [23]. With the increasing application of LiDAR in multiple fields, such as autonomous driving, remote sensing, and photogrammetry, there is a rise of large scale data sets with more than millions of points. These data sets accelerate the crucial breakthroughs and discriminate performances in point cloud semantic segmentation, 3-D object detection, and classification. Apart from the mobile LiDAR data, some data sets [56] acquired by terrestrial laser scanning (TLS) by static LiDAR are also employed due to they provide high-quality point cloud data.

TABLE I
SURVEY OF EXISTING LIDAR DATA SET

| Dataset | Format | Primary Fields | Points / Objects | # Classes | Sparsity | Highlight |
|---|---|---|---|---|---|---|
| **Segmentation** | | | | | | |
| Semantic3D [56] | ASCII | X, Y, Z, Intensity, R, G, B | 4 billion points | 8 | Dense | training & testing; competing with the most algorithms |
| Oakland [57] | ASCII | X, Y, Z, Class | 1.6 million points | 5 | Sparse | training to tune model architecture |
| iQmulus [58] | PLY | X, Y, Z, Intensity, GPS time, Scan origin, # echoes, Object ID, Class | 300.0 million points | 22 | Moderate | training & testing |
| Paris-Lille-3D [59] | PLY | X, Y, Z, Intensity, Class | 143.1 million points | 50 | Moderate | training & testing; competing with limited algorithms |
| **Localization/Detection** | | | | | | |
| KITTI Object Detection/ Bird's Eye View Benchmark [60] | - | 3D bounding boxes | 80,256 objects | 3 | Sparse | training & testing; competing with the most algorithms |
| **Classification/Recognition** | | | | | | |
| Sydney Urban Objects Dataset [61] | ASCII | Timestamp, Intensity, Lser id, X, Y, Z, Azimuth,Range, id | 588 objects | 14 | Sparse | training & testing; competing with limited algorithms |
| ModelNet [30] | ASCII | X, Y, Z, number of Vertices, edges, faces | 12,311 objects (ModelNet40) 4,899 objects (ModelNet10) | 40 (ModelNet40) 10 (ModelNet10) | Dense | training & testing; competing with most algorithms |

As shown in Table I, we classify those existing data sets related to our topic into three types: segmentation-based data sets, detection-based data sets, and classification-based data sets. Besides, the long-term autonomy data set is also summarized.

*1) Segmentation-Based Data Sets (Semantic3D [56]):* Semantic3D is the existing largest LiDAR data set for outdoor scene segmentation tasks with more than 4 billion points and around 110 000 $m^2$ covering area. This data set is labeled with eight classes and split into train and test sets with nearly equal size. These data are acquired by a static LiDAR with high measurement resolution and covered long measurement distance. The challenges for this data set mainly stems from the massive point clouds, unevenly distributed point density, and severe occlusions. In order to fit the high computation algorithms, a reduced-8 data set is introduced for training and testing, which shares the same training data but fewer test data compared with Semantic3D.

Oakland 3-D Point Cloud Data Set [57]. This data set is acquired in an early year compared with the above-mentioned two data sets. A mobile platform equipped with LiDAR is used to scan the urban environments and generated around 1.3 million points, while 100 000 points are split into a validation set. The whole data set is labeled with five classes, such as wire, vegetation, ground, pole/tree-trunk, and facade. This data set is small and, thus, suitable for lightweight networks. Besides, this data set can be used to test and tune the network architectures without a lot of training time before final training on other data sets.

IQmulus & TerraMobilita Contest [58]. This data set is also acquired by a mobile LiDAR system in the urban environment in Paris. There are more than 300 million points in this data set, which covered 10-km street. The data is split into 10 separate zones and labeled with more than 20 fine classes. However, this data set also has severe occlusion.

Paris-Lille-3D [59]. Compared with Semantic3D [56], Paris-Lille-3D contains fewer points (140 million points) and smaller covering area (55 000 $m^2$). The main difference is that this data set is acquired by a Mobile LiDAR system in two cities: Paris and Lille. Thus, the points in this data set are sparse and comparatively low measurement resolution compared with Semantic3D [56]. But this data set is more similar to the LiDAR data acquired by AVs. The whole data set is fully annotated into 50 classes unequally distributed in three scenes: Lille1, Lille2, and Paris. For simplicity, these 50 classes are combined into 10 coarse classes for challenging.

*2) Detection-Based Data Sets (KITTI Object Detection/Bird's Eye View Benchmark [60]):* Different from the above LiDAR data sets which are specific for the segmentation task, the KITTI data set is acquired from an autonomous driving platform and records six hours driving using digital cameras, LiDAR, and global positioning system/ inertial measurement unit (GPS/IMU) inertial navigation system. Thus, apart from the LiDAR data, the corresponding imagery data are also provided. Both the Object Detection and Bird's Eye View (BEV) Benchmark contains 7481 training images and 7518 test images and the corresponding point clouds. Due to the moving scanning mode, the LiDAR data in this benchmark is highly sparse. Thus, only three objects are labeled with bounding boxes: cars, pedestrians, and cyclists.

*3) Classification-Based Data Sets (Sydney Urban Objects Data Set [61]):* This data set contains a set of general urban road objects scanned with a LiDAR in the CBD of Sydney, NSW, Australia. There are 588 labeled objects which are classified into 14 categories, such as vehicles, pedestrians, signs, and trees. The whole data set is split into four folds

TABLE II
EVALUATION METRICS FOR 3-D POINT CLOUD SEGMENTATION, DETECTION/LOCALIZATION, AND CLASSIFICATION

| Metric | Equation | Description |
|---|---|---|
| $IoU$ | $IoU_i = \frac{c_{ii}}{c_{ii}+\sum_{j\neq i} c_{ij}+\sum_{k\neq i} c_{ki}}$ | Intersection over Union, where $c_{ij}$ is the number of points from ground-truth class $i$ predicted as class $j$ [62] |
| $\overline{IoU}$ | $\overline{IoU} = \frac{\sum_{i=1}^{N} IoU_i}{N}$ | Mean IoU, where N is the number of classes |
| OA | $\text{OA} = \frac{\sum_{i=1}^{N} c_{ii}}{\sum_{j=1}^{N}\sum_{k=1}^{N} c_{jk}}$ | Overall accuracy |
| $Precision$ | $Precision = \frac{TP}{TP+FP}$ | The ratio of correctly detected objects in the whole detection results, where $TP, TN, FP$,and $FN$ are the number of true positives, true negatives, false positives,and false negatives, respectively [63] |
| $Recall$ | $Recall = \frac{TP}{(TP+FN)}$ | The ratio of correctly detected objects in the ground truth |
| $F_1$ | $F_1 = \frac{2TP}{(2TP+FP+FN)}$ | The balance between precision and recall |
| $MCC$ | $MCC = \frac{(TP\times TN - FP\times FN)}{((TP+FP)(TP+FN)(TN+FP)(TN+FN))}$ | The combined ratio of detected and undetected objects as well as non-objects |
| AP | $\text{AP} = \frac{1}{11}\sum_{r\in\{0,1,\ldots,1\}} \max_{\tilde{r};\tilde{r}\geq r} p(\tilde{r})$ | Average Precision, where $r$ represents the recall, $p_(r)$ represents the precision |
| $AOS$ | $AOS = \frac{1}{11}\sum_{r\in\{0,0.1,\ldots1\}} \max_{\tilde{r};\tilde{r}\geq r} s(\tilde{r}),$ | Average Orientation Similarity |
| $s(r)$ | $s(r) = s(r) = \frac{1}{\lvert\mathcal{D}(r)\rvert}\sum_{i\in\mathcal{D}(r)} \frac{1+\cos\Delta_\theta^{(i)}}{2}\delta_i$ | Orientation similarity, where $\mathcal{D}(r)$ represents the whole object detection at recall rate $r$ and $\Delta_\theta^{(i)}$ is the angle difference between predicted and ground truth orientation of detection $i$, $\delta_i$ is the penalty value when multiple detection tasks describe one object |

for training and testing. Similar to other LiDAR data sets, the collected objects in this data set are sparse with incomplete shape. Although it is small and not ideal for the classification task, it is the most commonly used benchmark due to the limitation of the tedious labeling process.

ModelNet [30]. This data set is the existing largest 3-D benchmark for 3-D object recognition. Different from the Sydney Urban Objects data set [61], which contains road objects collected by LiDAR sensors, this data set is composed of general objects in CAD models with evenly distributed point density and complete shape. There are approximately 130K labeled models in a total of 660 categories (e.g., car, chair, and clock). The most commonly used benchmark is the ModelNet40 that contains 40 general objects and the ModelNet10 with 10 general objects. The milestone 3-D deep architectures are commonly trained and tested on these two data sets due to the affordable computation burden and time.

Long-Term Autonomy. To address challenges of long-term autonomy, a novel data set for autonomous driving has been presented by Maddern *et al.* [64]. They collected images, LiDAR, and GPS data while traversing 1000 km in central Oxford in the U.K. for one year. This allowed them to capture different scene appearances under various illumination, weather, and season with dynamic constructions. Such long-term data sets allow for in-depth investigation of problems that detain the realization of autonomous vehicles, such as localization, at different times of the year.

### B. Evaluation Metrics

To evaluate the performances of those proposed methods, several metrics, as summarized in Table II, are proposed for those tasks: segmentation, detection, and classification. The detail of these metrics is given as follows.

For the segmentation task, the most commonly used evaluation metrics are the Intersection over Union (IoU) metric, $\overline{\text{IoU}}$, and overall accuracy (OA) [62]. IoU defines the quantify of the percent overlap between the target mask and the prediction output [56].

For detection and classification tasks, the results are commonly analyzed regionwise. Precision, recall, $F_1$-score, and Matthews correlation coefficient (MCC) [65] are commonly used to evaluate the performance. The precision represents the ratio of correctly detected objects in the whole detection results, while the recall means the percentage of the correctly detected objects in the ground truth, the $F_1$-score conveys the balance between the precision and the recall, and the MCC is the combined ratio of detected and undetected objects and nonobjects.

For 3-D object localization and detection task, the most frequently used metrics are: average precision ($AP_{3D}$) [66] and average orientation similarity [36]. The average precision is used to evaluate the localization and detection performance by calculating the averaged valid bounding box overlaps, which exceed predefined values. For orientation estimation, the orientation similarities with different thresholded valid bounding box overlaps are averaged to report the performance.

## IV. GENERAL 3-D DEEP LEARNING FRAMEWORKS

In this section, we review the milestone DL frameworks on 3-D data. These frameworks are pioneers in solving the problems defined in Section II. Besides, their stable and efficient performance makes them suitable for use as the backbone framework in detection, segmentation, and classification tasks. Although 3-D data acquired by LiDAR is often in the form of point clouds, how to represent point cloud and what DL models to use for detection, segmentation, and classifications remains an open problem [41]. Most existing 3-D DL models process point clouds mainly in form of voxel grids [30], [67]–[69], point clouds [10], [12], [70], [71], graphs [72]–[75], and 2-D images [15], [76]–[78]. In this section, we analyze the frameworks, attributes, and problems of these models in detail.

### A. Voxel-Based Models

Conventionally, convolutional neural networks (CNNs) are mainly applied to data with regular structures, such as the 2-D

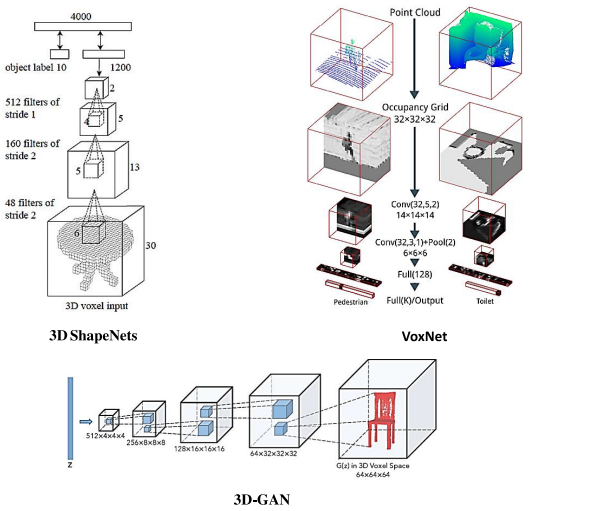



Fig. 3. Deep architectures of 3-D ShapeNet [30], VoxNet [67], 3-D-GAN [68].

pixel array [79]. Thus, in order to apply CNNs to unordered 3D point clouds, such data are divided into regular grids with a certain size to describe the distribution of the data in 3-D space. Typically, the size of the grid is related to the resolution of the data [80]. The advantage of voxel-based representation is that it can encode the 3-D shape and the viewpoint information by classifying the occupied voxels into several types, such as visible, occluded, or self-occluded. Besides, 3-D convolution (Conv) and pooling operations can be directly applied in voxel grids [69].

3-D ShapeNet proposed by Wu *et al.* [30] and shown in Fig. 3, is the pioneer in exploiting 3-D volumetric data using a convolutional deep belief network. The probability distribution of binary variables is used to represent the geometric shape of a 3-D voxel grid. Then, these distributions are input to the network, which is mainly composed of three Conv layers. This network is initially pretrained in a layerwise fashion and then trained with a generative fine-tuning procedure. The input and Conv layers are modeled based on the contrastive divergence, where the output layer was trained based on the fast-persistent contrastive divergence. After training, the input test data is output with a single-depth map and then transformed to represent the voxel grid. ShapeNet has notable results in low-resolution voxels. However, the computation cost increases cubically with the increment of input data size or resolution, which limits the model's performance in large-scale or dense point clouds. Besides, multiscale and multiview information from the data is not fully exploited, which hinder the output performance.

VoxNet is proposed by Maturana and Scherer [67] to conduct 3-D object recognition using 3-D convolution filters based on volumetric data representation, as shown in Fig. 3. Occupancy grids represented by a 3-D lattice of random variables are employed to show the state of the environment. Then a probabilistic estimate is used to estimate the occupancy of these grids, which is maintained as the prior knowledge. Three different occupancy grid models, such as binary occupancy grid, density grid, and hit grid, are experimented to select the best model. This network framework is mainly composed of Conv, pooling layer, and fully connected (FC) layers. Both ShapeNet [30] and VoxNet employ rotation augmentation for training. Compared with ShapeNet [30], VoxNet has a smaller architecture that has less than 1 million parameters. However, some occupancy grids contain useless information but only increase the computation cost.

3-D-GAN [68] combines the merits of both general-adversarial network (GAN) [81] and volumetric convolutional networks [67] to learn the features of 3-D objects. This network is composed of a generator and a discriminator, as shown in Fig. 3. The adversarial discriminator is conducted to classify objects into synthesized and real categories. This operation has the following two merits: the generative-adversarial criterion has the advantage in capturing the structural variation between two 3-D objects; and the employment of generative-adversarial loss is helpful to avoid the possible criterion-dependent overfitting. The generator attempts to confuse the discriminator. Both generator and discriminator consist of five volumetric fully Conv layers. This network provides a powerful 3-D shape descriptor with unsupervised training in 3-D object recognition. But the density of data affects the performance of adversarial discriminator for finest feature capturing. Consequently, this adaptive method is suitable for evenly distributed point clouds.

In conclusion, there are some limitations of this general volumetric 3-D data representation.

1) First, not all voxel representations are useful because they contain occupied and nonoccupied parts of the scanning environment. Thus, the high demand for computer storage is actually unnecessary within this ineffective data representation [69].
2) Second, the size of the grid is hard to set, which affects the scale of input data and may disrupt the spatial relationship between points.
3) Third, computation and memory requirements grow cubically with the resolution [69]. Thus, existing voxel-based models are maintained at low 3-D resolutions, and the most commonly used size is $30^3$ for each grid [69].

A more advanced voxel-based data representation is the octree-based grids [69], [82], which use adaptive size to divide the 3-D point clouds into cubes. It is a hierarchical data structure that recursively decomposes the root voxels into multiple leaf voxels.

OctNet is proposed by Riegler *et al.* [69], which exploits the sparsity of the input data. Motivated by the observation that the object boundaries have the highest probability in producing the maximum responses across all feature maps generated by the network at different layers, they partitioned the 3-D space hierarchically into a set of unbalanced octrees [83] based on the density of the input data. Specifically, the octree nodes that have point clouds are split recursively into its domain, ending at the finest resolution of the tree. Thus, the size of leaf nodes varies. For each leaf node, those features that activate their comprised voxel is pooled and stored. Then, the convolution filters are conducted in these trees. In [82], the deep model is constructed by learning the structure of the octree and the represented occupancy value for each grid. This octree-based data representation largely reduces the computation and memory resources for DL architectures, which achieves

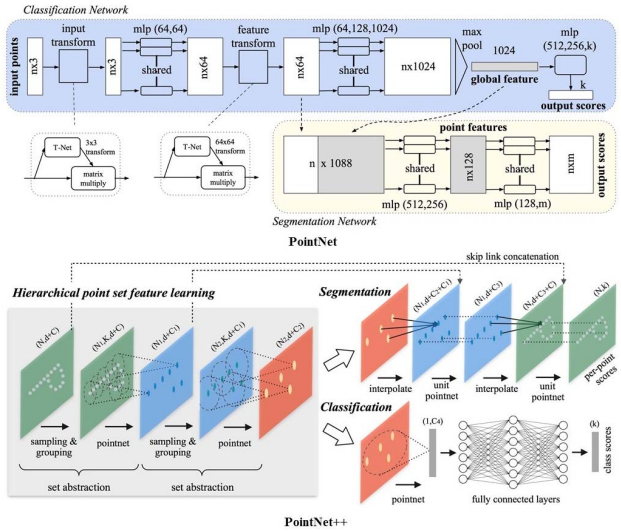



Fig. 4. PointNet [10] and PointNet++ [12] architectures.

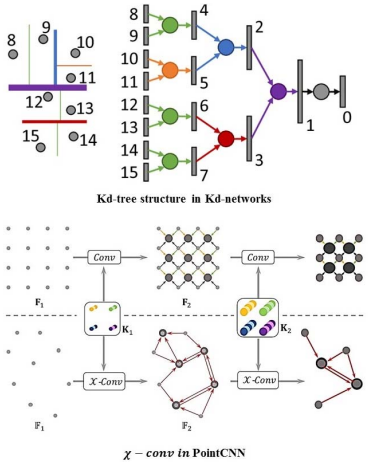



Fig. 5. Kd-tree structure in Kd-networks [70] and $\chi$-Conv in PointCNN [71].

better performance in high-resolution 3-D data compared with voxel-based models. However, the disadvantage of octree data is similar to voxels; both of them fail to exploit the geometry feature of 3-D objects, especially the intrinsic characteristics of patterns and surfaces [29].

### *B. Point Clouds Based Models*

Different from the volumetric 3-D data representation, point clouds can preserve the 3-D geospatial information and the internal local structure. Besides, the voxel-based models that scan the space with fixed strides are constrained by the local receptive fields. But for point clouds, the input data and their metric decide the range of receptive fields, which has high efficiency and accuracy.

PointNet [10], as a pioneer in consuming 3-D point clouds directly for deep models, learns the spatial feature of each point independently via MLP layers and then accumulates their features by max pooling. The point clouds are input directly to the PointNet, which predicts per-point label or per-object label, and its framework is shown in Fig. 4. In PointNet, a spatial transform network and a symmetric function are designed to improve the data invariance to permutation. The spatial feature of each input point is learned through the networks. Then, the learned features are assembled across the whole region of point clouds. PointNet has achieved outstanding performances in 3-D object classification and segmentation tasks. However, the individual point features are grouped and pooled by max pooling, which fails to preserve the local structure. As a result, PointNet is not robust to fine-grained patterns and complex scenes.

PointNet++ is proposed by Qi *et al.* [12] after PointNet, which compensates the local feature extraction problems in PointNet. Within the raw unordered point clouds as input, these points are initially divided into overlapping local regions using the Euclidean distance metric. In order to sample the points evenly over the whole point set, the farthest point sampling algorithm is applied. Local features are extracted from the small neighborhoods around the selected points using the K-nearest-neighbor (KNN) or query-ball searching methods. These neighborhoods are gathered into larger clusters and leveraged to extract high-level features via PointNet [10]. The sampling and grouping module are repeated until the local and global features of the whole points are learned, as shown in Fig. 4. For the segmentation task, these features are backpropagated to the finest layer to extract per-point features. This network, which outperforms the PointNet [10] network in classification and segmentation tasks, extracts the local features for points in different scales. However, features from the local neighborhood points in different sampling layers are learned in an isolated fashion. Besides, the max-pooling operation based on PointNet [10] for high-level feature extraction in PointNet++ fails to preserve the spatial information between the local neighboring points.

Kd-networks [70] uses the kd-tree to create the order of the input points, which is different from PointNet [10] and PointNet++ [12] as both of them use the symmetric function to solve the permutation problem. Klokov and Lempitsky [70] used the maximum range of point coordinates along the coordinate axis to recursively split the certain size point clouds $N = 2^D$ into subsets with a top–down fashion to construct a kd-tree. As shown in Fig. 5, this kd-tree is ending with a fixed depth. Within this balanced tree structure, vectorial representations in each node, which represents a subdivision along a certain axis, are computed using kd-networks. These representations are then exploited to train a linear classifier. This network has a better performance than PointNet [10] and PointNet++ [12] in small objects classification. However, it is not robust to rotations and noise, since these variations can lead to the change of tree structure. Besides, it lacks the overlapped receptive field, which reduces the spatial correlation between leaf nodes.

PointCNN, proposed by Li *et al.* [71], solves the input points permutation and transformation problems based on an $\chi$-Conv operation, as shown in Fig. 5. They proposed the $\chi$-transformation, which is learned from the input points by weighting the input point features and permutating the points into a latent and potentially canonical order. Then, the traditional convolution operators are applied in the learned $\chi$-transformation features. These spatially local correlation features in each local range are aggregated to construct a

hierarchical CNN network architecture. However, this model still has not exploited the correlations of different geometric features and their discriminate information toward results, which limits the performance.

Point cloud-based deep models are mostly focused on solving permutation problems. Although they treat points independently at local scales to maintain permutation invariance, this independence, however, neglects the geometric relationships among points and their neighbors, presenting a fundamental limitation that leads to the missing of local features.

### C. Graph-Based Models

Graphs are a type of non-Euclidean data structure that can be used to represent point clouds. Their node corresponds to each input point, and the edges represent the relationship between each point neighbors. Graph neural networks propagate the node states until equilibrium in an iterative manner [75]. With the advancement of CNNs, there is an increment graph convolutional networks applied to 3-D data. Those graph CNNs define convolutions directly on the graph in the spectral and nonspectral (spatial) domain, operating on groups of spatially close neighbors [84]. The advantage of graph-based models is to explore the geometric relationships among points and their neighbors. Thus, spatially local correlation features are extracted from the grouped edge relationships on each node. But there are two challenges for constructing graph-based deep models as follows.

1) First, defining an operator that is suitable for dynamically sized neighborhoods and maintaining the weight sharing scheme of CNNs [75].
2) Second, exploiting the spatial and geometric relationships among each node's neighbors.

SyncSpecCNN [72] exploited the spectral eigendecomposition of the graph Laplacian to generate a convolution filter applied to point clouds. Yi *et al.* [72] constructed SyncSpecCNN based on those two considerations: the first is the coefficients sharing and multiscale graph analyzing, and the second is information sharing across related but different graphs. They solved these two problems by constructing the convolution operation in the spectral domain: the signal of point sets in the Euclidean domain is defined by the metrics on the graph nodes, and the convolution operation in the Euclidean domain is related to the scaling signals based on eigenvalues. Actually, such operation is linear and only applicable to the graph weights generated from eigenvectors of the graph Laplacian. Although SyncSpecCNN has achieved excellent performance in 3-D shape part segmentation, it has several limitations.

1) Basis-dependent. The learned spectral filter's coefficients are not suitable for another domain with a different basis.
2) Computationally expensive. The spectral filtering is calculated based on the whole input data, which requires high computation capability.
3) Missing local edge features. The local graph neighborhood contains useful and distinctive local structural information, which is not exploited.

Edge-conditioned convolution (ECC) [73] considers the edge information in constructing the convolution filters based on the graph signal in the spatial domain. The edge labels

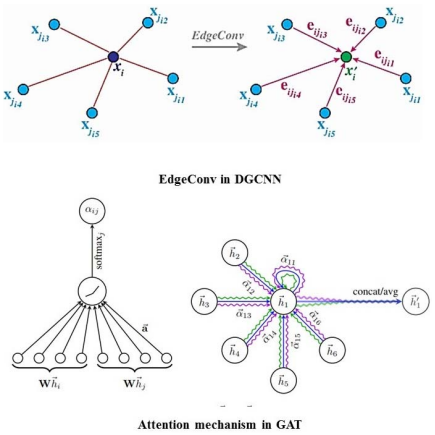



Fig. 6. EdgeConv in DGCNN [74] and attention mechanism in GAT [75].

in a vertex neighborhood are conditioned to generate the filter weights. Besides, in order to solve the basis-dependent problem, the convolution operator is dynamically generalized for arbitrary graphs with varying sizes and connectivity. The whole network follows the common structure of a feedforward network with interlaced convolutions and pooling followed by global pooling and FC layers. Thus, features from local neighborhoods are extracted continually from these stacked layers, which increase the receptive field. Although the edge labels are fixed for a specific graph, the learned interpretation networks may vary in different layers. ECC learns the dynamic pattern of local neighborhoods, which is scalable and effective. However, the computation cost remains high, and it is not applicable for large-scale graphs with continuous edge labels.

Dynamic graph CNN (DGCNN) [74] also constructs a local neighborhood graph to extract the local geometric features and applies Conv-like operations, named EdgeConv, as shown in Fig. 6, on the edges connecting neighboring pairs of each point. Different from ECC [73], EdgeConv dynamically updates the given fixed graph with Conv-like operations for each layer output. Thus, DGCNN can learn how to extract local geometric structures and group point clouds. This model takes $n$ points as input, and then find the K neighborhoods of each point to calculate the edge feature between the point and its K neighborhoods in each EdgeConv layer. Similar to the PointNet [10] architecture, the features convolved in the last EdgeConv layer are aggregated globally to construct a global feature, while all the EdgeConv outputs are treated as local features. Local and global features are concatenated to generate results' scores. This model extracts distinctive edge features from point neighborhoods, which can be applied in different point cloud-related tasks. However, the fixed size of the edge features limits the performance of the model when facing different scales and resolution point clouds.

ECC [73] and DGCNN [74] propose general convolutions on graph nodes and their edge information, which is isotropy about input features. However, not all the input features contribute equally to its nodes. Thus, attention mechanisms [75] are introduced to deal with variable sized inputs and focus on the most relevant parts of the nodes' neighbors to make decisions.

Graph Attention Networks (GAT) [75]. The core insight behind GAT is to calculate the hidden representations of

each node in the graph, by assigning different attentional weights to different node neighbors, following a self-attention strategy. Within a set of node features as input, a shared linear transformation, parametrized by a weight matrix is applied to each node. Then, a self-attention, a shared attentional mechanism which is shown in Fig. 6, is applied on the nodes to computes attention coefficients. These coefficients indicate the importance of corresponding nodes' neighbor features, respectively, and are further normalized to make them comparable across different nodes. These local features are combined according to the attentional weights to form the output features for each node. In order to improve the stability of the self-attention mechanism, multihead attention is employed to conduct $k$ independent attention schemes, which are then concatenated together to form the final output features for each node. This attention architecture is efficient and can extract fine-grained representations for each graph node by assigning different weights to the neighbors. However, the local spatial relationship between neighbors is not considered in calculating the attentional weights. To further improve its performance, Wang *et al.* [85] proposed graph attention convolution to generate attentional weights by considering different neighboring points and feature channels.

### *D. View-Based Models*

The last type of MLS data representation is 2-D views obtained from 3-D point clouds from different directions. With the projected 2-D views, traditional well-established CNNs and pretrained networks on image data sets, such as AlexNet [86], VGG [87], GoogLeNet [88], and ResNet [89], can be exploited. Compared with voxel-based models, these methods can improve the performance for different 3-D tasks by taking multiview of the interest object or scenes and then fusing or voting the outputs for final prediction. Compared with the above-mentioned three different 3-D data representations, view-based models can achieve near-optimal results, as shown in Table III. Su *et al.* [90] experimented that multiview methods have the optimal generalization ability even without using pretrained models compared with point cloud and voxel data representation models. The advantages of view-based models compared with 3-D models can be concluded as follows.

1) Efficiency. Compared with 3-D data representations, such as point clouds or voxel grids, the reduced one dimension information can greatly reduce the computation cost but with increased resolution [76].
2) Exploiting established 2-D deep architectures and data sets. The well-developed 2-D DL architectures can better exploit the local and global information from projected 2-D view images [91]. Besides, existing 2-D image databases (such as ImageNet [92]) can be used to train 2-D DL architectures.

Multiview CNN (MVCNN) [76] is the pioneer in exploiting 2-D DL models to learn the 3-D representation. Multiple views of 3-D objects are extracted without specific order using a view pooling layer. Two different CNN models are proposed and tested. The first CNN model takes 12 views rendered from the object via placing 12 virtual cameras with equal distance around the objects as the input, while the second CNN model takes 80 views rendered in the same way as the input. These views are first learned separately and then fused through max-pooling operation to extract the most representative feature among all views for the whole 3-D shape. This network is effective and efficient compared with volumetric data representation. However, the max-pooling operation only considers the most important views and discards information from other views, which fails to preserve comprehensive visual information.

MVCNN-MultiRes is proposed by Qi *et al.* [15] to improve multiview CNNs. Different from traditional view rendering methods, the 3-D shape is projected to 2-D via a convolution operation based on an anisotropic probing kernel applied to the 3-D volume. Multiorientation pooling is combined together to improve the 3-D structure capturing capability. Then, the MVCNN [76] is applied to classify the 2-D projects. Compared with MVCNN [76], multiresolution 3-D filtering is introduced to capture multiscale information. Sphere rendering is performed at different volume resolutions to achieve view-invariant and improve the robust to potential noise and irregularities. This model achieves better results in 3-D object classification task compared with the MVCNN [76].

3DMV [77] combines the geometry and imagery data as input to train a joint 3-D deep architecture. Feature maps extracted from imagery data are first extracted and then mapped into the 3-D feature extracted from the volumetric grid data derived from a differentiable back-projection layer. Because there exists redundant information among multiple views, a multiview pooling approach is applied to extract useful information from these views. This network has achieved remarkable results in 3-D objects classification. However, compared with models using one source of data, such as LiDAR points or RGB images solely, the computation cost of this method is higher.

RotationNet [78] is proposed following the assumption that when the object is observed by a viewer from a partial set of full multiview images, the observation direction should be recognized to correctly infer the object's category. Thus, the multiview images of an object are input to the RotationNet, which outputs its pose and category. The most representative characteristic of RotationNet is that it treats viewpoints which are the observation of training images as latent variables. Then unsupervised learning of object poses is conducted based on an unaligned object data set, which can eliminate the process of pose normalization to reduce noise and individual variations in shape. The whole network is constructed as a differentiable MLP network with softmax layers as the final layer. The outputs are the viewpoint category probabilities, which correspond to the predefined discrete viewpoints for each input image. These likelihoods are optimized by the selected object pose.

However, there are some limitations of 2-D view-based models.

1) The first is that the projection from 3-D space to 2-D views can lose some geometrically related spatial information.
2) The second is redundant information among multiple views.

TABLE III
SUMMARIZING OF MILESTONE DL ARCHITECTURES BASED ON FOUR POINT CLOUD DATA REPRESENTATIONS

| Model | Input Size | Hightlights | Disadvanatges | Model size(MB) | Acc (%) |
|---|---|---|---|---|---|
| **Voxel** | | | | | |
| 3dShapeNet [30] | voxels | Pioneer in exploiting 3D volumetric data; Permutation and orientation invariance. | Computation and memory requirement grows cubically; Use one view in a fixed voxel size. | 12 | 84.7 |
| VoxNet [67] | voxels | Occupancy grids are employed to represent the distribution of the scene as a 3D lattice of random variables for each grid; Permutation and orientation invariance; Improved efficiency. | Not all occupancies are useful. | 1.0 | 85.9 |
| 3D-GAN [68] | voxels | Combines the adversarial modeling and volumetric convolutional networks to learn features; Permutation and orientation invariance; Rigid transformation invariance. | Not invariance to data density variation. | 7 | 83.3 |
| OctNet [69] | hybrid grid octree | Hierarchically divide the data into a series of unbalanced octrees according to data density; Permutation and orientation invariance; Efficient. | Fail to preserve the geometry relationship among points. | 0.4 | 86.5 |
| **Point Clouds** | | | | | |
| PointNet [10] | 1024 points | Pioneer in applying DL using 3D point clouds and solving the permutation problem via maxpooling. | Not capture local structure induced by the metric; Hard to generalize to unseen point configurations. | 40 | 89.2 |
| PointNet++ [12] | 5000 points +normal | Hierarchically learn multi-scale local geometric features and aggregate them for inference; Permutation and rigid transformation invariance and efficient. | Local spatial relationship among point neighborhoods is not exploited. | 12 | 90.7 |
| Kd-networks [70] | 1024 points | Use the kd-tree to create the order of the input points and hierarchically extract features from the leaves to root; Permutation and rigid transformation invariance. | Non-invariance to rotations and noises; Computation grows linearly with increasing resolution; Low spatial-correlation between leaf nodes. | 120 | 91.8 |
| PointCNN [71] | 1024 points | Propose X-Conv operator that permutes and weights input points and features; Permutation and rigid transformation invariance. | Not exploit the correlations of different geometric features and their discriminative information toward final results. | 4.5 | 92.2 |
| **Graph** | | | | | |
| Spectral -CNN [72] | graphs | Exploit the spectral eigen-decomposition of the graph Laplacian to generate a Conv-like operator. | Basis-dependent; Computationally expensive; Missing local edge features. | 0.8 | - |
| ECC [73] | graphs | The edge labels in a vertex neighborhood are conditioned to generate the Conv filter weights; Permutation invariance. | High computation cost; Not suitable for large-scale graphs with continuous labels; Isotropic about input features. | - | 87.4 |
| DGCNN [74] | graphs | Extract edge features and dynamically update the graph for each layer; Permutation invariance. | Fixed size edge features are not invariance to points with different resolution and scale; Isotropic about input features. | 21 | 92.2 |
| GAT [75] | graphs | Compute the hidden representations of each node's neighbors, following a self-attention strategy; Permutation and invariance, improved accuracy. | Apply the attention mechanism only to input points not to their local features. | - | - |
| **2D View** | | | | | |
| MVCNN [76] | 12 views | Pioneer in applying CNN to each view and then aggregate the features by a view pooling procedure; Permutation and orientation invariance; Efficient. | Multi-resolution features are not considered. | 99 | 90.1 |
| MVCNN -MultiRes [15] | 20 views | Propose multi-resolution 3D filtering to capture comprehensive information at multi-scales; Permutation and orientation invariance; Efficient. | Geometric information are not exploited. | 16.6 | 91.4 |
| 3DMV [77] | 20 views | Extract RGB and geometric features and aggregate them via a joint 2D-3D network; Permutation and orientation invariance; Efficient. | 2D occlusion and background clutter affects the 3D network performance. | - | - |
| RotationNet [78] | 12 views | Treat the viewpoints of the observed training images as latent variables; Permutation and orientation invariance; High accuracy. | Not suitable for per-point processing tasks. | 59 | **97.37** |

### *E. 3-D Data Processing and Augmentation*

Due to the massive amount of data and the tedious labeling process, there exist limited reliable 3-D data sets. To better exploit the architecture of deep networks and improve the model generalization ability, data augmentation is commonly conducted. Augmentation can be applied to both data space and feature space, while the most common augmentation is conducted in the first space. This type of augmentation can not only enrich the variations of data but also generate new samples by conducting transformations to the existing 3-D data. There are several types of transformations, such as translation, rotation, and scaling. Several requirements for data augmentation are summarized as follows.

1) There must exist similar features between original augmented data, such as shapes.
2) There must exist different features between original and augmented data, such as orientation.

Based on those existing methods, classical data augmentation for point clouds can be concluded as follows.

1) Mirror $x$- and $y$-axes with the predefined probability [59], [93].
2) Rotation around the $z$-axis with certain times and angles [13], [59], [93], [94].

3) Random (uniform) height or position jittering in certain ranges [67], [93], [95].
4) Random scale with certain ratio [13], [59].
5) Random occlusions or randomly down-sampling points within the predefined ratio [59].
6) Random artifacts or randomly down-sampling points within the predefined ratio [59].
7) Randomly adding noise, following certain distribution, to the points' coordinates and local features [45], [59], [96].

## V. Deep Learning in LiDAR Point Cloud for AVs

The application of LiDAR point clouds for AVs can be concluded into three types: 3-D point cloud semantic segmentation, 3-D object detection and localization, and 3-D object classification and recognition. Targets for these tasks vary a lot; for example, the scene segmentation focuses on the per-point label prediction, while the detection and the classification concentrate on integrated point set labeling. But they all need to exploit the input point feature representations before feature embedding and network construction.

We first make a survey of input point cloud feature representations applied in DL architectures for all these three tasks, such as local density and curvature. These features are representations of a specific 3-D point or position in 3-D space, which describe the geometrical structures and features based on the extracted information around the point. These features can be grouped into two types: one is derived directly from the sensors, such as coordinates and the intensity, we term them as direct point feature representations; the second is extracted from the information provided by each point's neighbors, we term them as geo-local point feature representations.

*1) Direct Input Point Feature Representations:* The direct input point feature representations are mainly provided by laser scanners, which include the $x$, $y$, and $z$ coordinates, and other characteristics (e.g., intensity, angle, and number of returns). Two most frequently used features applied in DL are listed in the following.

1) $XYZ$ coordinates. The most direct point feature representation is the $XYZ$ coordinates provided by the sensors, which means the position of a point in the real-world coordinate system.
2) Intensity. The intensity represents the reflectance characteristics of the material surface, which is one common characteristic of laser scanners [97]. Different objects have different reflectance, thus producing different densities in point clouds. For example, traffic signs have a higher intensity than vegetations.

*2) Geo-Local Point Feature Representations:* Local input point feature embeds the spatial relationship of points and their neighborhoods, which plays a significant role in point cloud segmentation [12], object detection [42], and classification [74]. Besides, the searched local region can be exploited by some operations, such as CNNs [98]. Two most representative and widely used neighborhood searching methods are KNNs [12], [96], [99] and spherical neighborhood [100].

The geo-local feature representations are usually generated from the searched region using the above-mentioned two neighborhood searching algorithms. They are composed of eigenvalues [e.g., $\eta_0$, $\eta_1$ and $\eta_2$ ($\eta_0 > \eta_1 > \eta_2$)] or eigenvectors (e.g., $\overrightarrow{v_0}$, $\overrightarrow{v_1}$, and $\overrightarrow{v_2}$) by decomposing the covariance matrix defined in the searched region. We list five most commonly used 3-D local feature descriptors applied in DL in the following.

1) Local density. The local density is typically determined by the quantity of points in a selected area [101]. Typically, the point density decreases when the distance of objects to the LiDAR sensor increases. In voxel-based models, the local density of points is related to the setting of voxel sizes [102].
2) Local normal. It infers the direction of the normal at a certain point on the surface. The equation about normal extraction can be found in [65]. In [103], the eigenvector $\overrightarrow{v_2}$ of $\eta_2$ in $C_i$ is selected as the normal vector for each point. However, in [10], the eigenvectors of $\eta_0$, $\eta_1$ and $\eta_2$ are all chose as the normal vectors of point $p_i$.
3) Local curvature. The local curvature is defined to be the rate at which the unit tangent vector changes the direction. Similar to the local normal calculation in [65], the surface curvature change in [103] can be estimated from the eigenvalues derived from the Eigen decomposition: curvature $= \eta_0/(\eta_0 + \eta_1 + \eta_2)$.
4) Local linearity. It is a local geometric characteristic for each point to indicate the linearity of its local geometry [104]: linearity $= (\eta_1 - \eta_2)/\eta_1$.
5) Local planarity. It describes the flatness of a given point neighbors. For example, ground points have higher planarity compared with tree points [104]: planarity $= (\eta_2 - \eta_3)/\eta_1$.

### A. LiDAR Point Cloud Semantic Segmentation

The goal of semantic segmentation is to label each point as belonging to a specific semantic class. For AVs segmentation tasks, these classes cloud be a street, buildings, cars, pedestrians, trees, or traffic lights. When applying DL for point cloud segmentation, the classification of small features is required [38]. However, the LiDAR 3-D point clouds are usually acquired in large scale, and they are irregularly shaped with changeable spatial contents. In a review of the recent five-year articles related in this region, we group these articles into three schemes according to the types of data representation: point cloud-based, voxel-based, and multiview-based models. There is limited research focusing on graph-based models, and thus, we combine the graph-based and point cloud-based models together to illustrate their paradigms. Each type of model is represented by a compelling deep architecture, as shown in Fig. 7.

*1) Point Cloud-Based Networks:* For point cloud-based networks, they are mainly composed of two parts: feature embedding and network construction. For the feature representing, both local and global features have demonstrated to be crucial for the success of CNNs [12]. However, in order to apply conventional CNNs, the permutation and orientation problems for unordered and unoriented points require a discriminative feature embedding network. Besides, lightweight, effective, and efficient deep network construction is another key module that affects the segmentation performance.

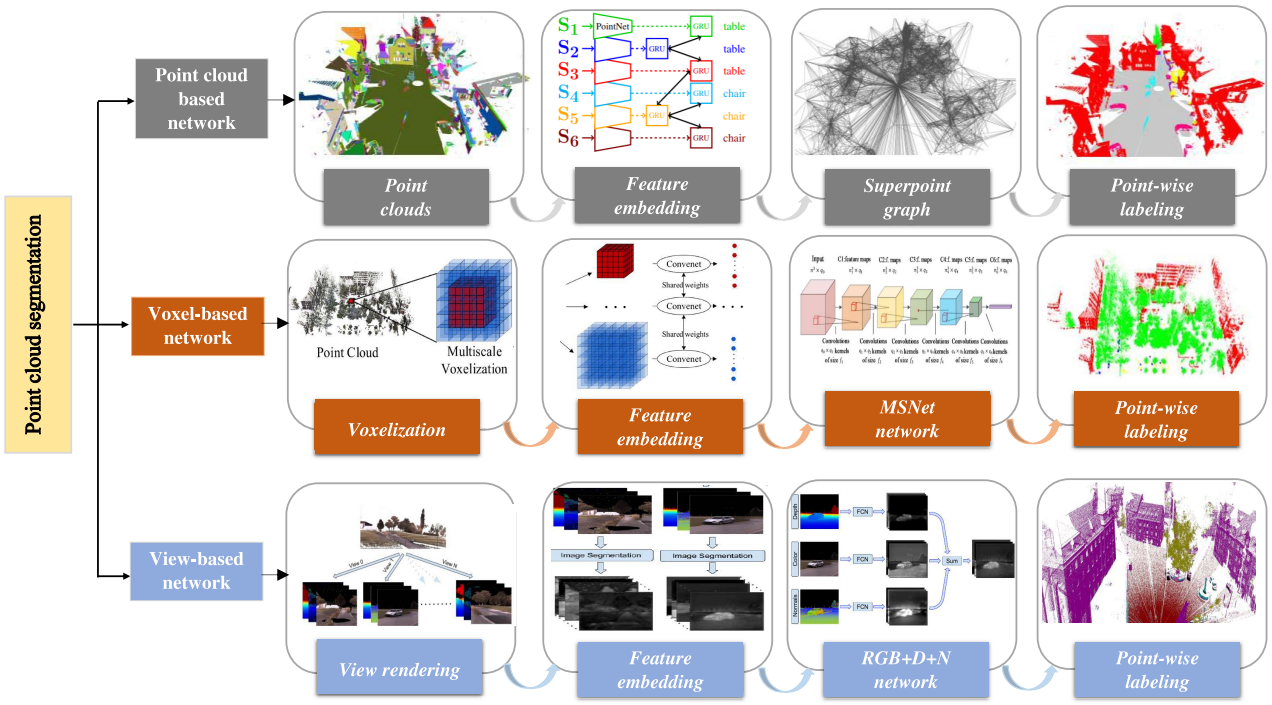



Fig. 7. DL architectures on LiDAR point cloud segmentation with three different data representations: point cloud-based networks represented by SPG [105], voxel-based networks represented by MSNet [106], view-based networks represented by DeePr3SS [107].

The local feature is commonly extracted from point neighborhoods [104]. The most frequently used local features are the local normal and curvature [10], [12]. To improve the receptive field, PointNet [10] has been proved to be a compelling architecture to extract semantic features from unordered point sets. Thus, in [12], [105], [108], and [109], a simplified PointNet is exploited to abstract local features from sampled point sets into high-level representations. Landrieu and Simonovsky [105] proposed superpoint graph (SPG) to represent large 3-D point clouds as a set of simple interconnected shapes coined superpoints, and then PointNet is operated on these superpoints to embed features.

To solve the permutation problem and extract local features, Huang *et al.* [40] proposed a novel slice pooling layer to extract the local context layer from the input point features and output an ordered sequence of aggregated features. To this end, the input points are first grouped into slices, and then a global representation for each slice is generated via concatenating point features within the slice. The advantage of this slice pooling layer is the low computation cost compared with point-based local features. However, the slice size is sensitive to the density of data. In [110], bilateral Conv layers (BCLs) are applied to perform convolutions on occupied parts of the lattice for the hierarchical and spatially aware feature learning. BCL first maps input points onto a sparse lattice and applies convolutional operations on the sparse lattice, and then the filtered signals are interpolated smoothly to recover the original input points.

To reduce the computation cost, in [108], an encoding–decoding framework is adopted. Features extracted from the same scale of abstraction are combined and then upsampled by 3-D deconvolutions to generate the desired output sampling density. Finally, these features are interpolated by the latent nearest-neighbor interpolation to output per-point label. However, the downsampling and upsampling operations are hard to preserve the edge information, and thus, cannot extract the fine-grained features. In [40], RNNs are applied to model dependencies of the ordered global representation derived from the slice pooling. Similar to the sequence data, each slice is viewed as one timestamp, and the interaction information with other slices also follows the timestamps in RNN units. This operation enables the model to generate dependencies between slices.

Although Zhang *et al.* [65] proposed the ReLu-NN to learn embedded point features, which is a four-layer MLP architecture. However, for objects without discriminative features, such as shrubs or trees, their local spatial relationship is not fully exploited. To better leverage the rich spatial information of objects, Wang *et al.* [111] proposed the spatial pooling to learn point features. The input data are clustered into groups, and then the minimum spanning tree-based pooling is applied to extract the spatial information among the points in the clustered point sets. Finally, an MLP is used for classification with these features. In order to achieve multiple tasks, such as the instance segmentation and the object detection with simple architecture, Wang *et al.* [109] proposed a similarity group proposal network-SGPN. Within the extracted local and global point features by PointNet, a feature extraction network generates a matrix which is then diverged into three subsets that each passes through a single PointNet layer to obtain three similarity matrices. These three matrices are used to produce a similarity matrix, a confidence map, and a semantic segmentation map.

*2) Voxel-Based Networks:* In voxel-based networks, the point clouds are first voxelized into grids, and then features are learned from these grids. The deep network is

finally constructed to map these features into segmentation masks.

Wang *et al.* [106] conducted a multiscale voxelization method to extract objects' spatial information at different scales to form a comprehensive description. At each scale, a neighboring cubic with selected length is constructed for a given point [112]. After that, the cube is divided into grid voxels with different size as a patch. The smaller the size is, the finer the scale is to provide. The point density and occupancy are selected to represent each voxel. The advantage of this kind of voxelization is that it can accommodate objects with different sizes without losing their spatial space information. In [113], the class probabilities for each voxel are predicted using 3-D-FCNN, which are then transferred back to the raw 3-D points based on the trilinear interpolation. In [106], after the multiscale voxelization of point clouds, features at different scales and spatial resolutions are learned by a set of CNNs with shared weights, which are finally fused together for the final prediction.

In the voxel-based point cloud segmentation task, there are two ways to label each point: 1) using the voxel label derived from the argmax of the predicted probabilities and 2) further globally optimizing the class label of the point cloud based on the spatial consistency. The first method is simple, but the result is provided at the voxel level and inevitably influenced by noise. The second one is more accurate but complex with an additional computation. Because the inherent invariance of CNN networks to spatial transformations affects the segmentation accuracy [25]. To extract the fine-grained details for volumetric data representations, the conditional random field (CRF) [106], [113], [114] is commonly adopted in a postprocessing stage. The CRFs have the advantage in combining the low-level information such as the interactions between points to output multiclass inferences for multiclass per-point labeling tasks, which compensate the fine local details that CNNs fail to capture.

*3) Multiview-Based Networks:* As for multiview-based models, view rendering and deep architecture construction are two key modules for the segmentation task. The first one is used to generate structural and well-organized 2-D grids that can exploit existing CNN-based deep architectures. The second one is proposed to construct the most suitable and generative models for different data.

In order to extract local and global features simultaneously, some hand-designed feature descriptors are employed for representative information extraction. In [65] and [111], the spin image descriptor is employed to represent point-based local features, which contains the global description of objects from partial views and clutters of local shape description. In [107], point splatting was applied to generate view images by projecting the points with a spread function into the image plane. The point is first projected into the image coordinate system of a virtual camera. For each projected point, its corresponding depth value and feature vectors such as the normal are stored.

Once the points are projected into multiview 2-D images, some discriminative 2-D deep networks can be exploited, e.g., VGG16 [87], AlexNet [86], GoogLeNet [88], and ResNet [89]. In [25], these deep networks have been detailed analyzed in 2-D semantic segmentation [25]. Among these methods, VGG16 [87], composed of 16 layers, is the most frequently used. Its main advantage is the use of stacked Conv layers with small receptive fields, which produces a lightweight network with limited parameters and increasing nonlinearity [25], [107], [115].

*4) Evaluation on Point Cloud Segmentation:* Due to the high volume of point clouds, which pose a great challenge for the computation capability. We choose the models tested on the reduced-8 Semantic3D data set to compare their performances, as shown in Table IV. Reduced-8 shares the same training data as semantic-8 but only use a small part of test data, which can also suit the high computation cost algorithm for competing. The metrics used to compare these models are $\mathrm{IoU}_i$, $\overline{\mathrm{IoU}}$, and OA. The computation efficiency for these algorithms is not reported and compared due to the difference between the computation capacity, selected training data sets, and model architectures.

### *B. 3-D Objects Detection (Localization)*

The detection (localization) of 3-D objects in LiDAR point clouds can be summarized as the bounding box prediction and objectness prediction [14]. In this article, we mainly survey the LiDAR-only paradigm, which takes advantage of the accurate geo-referenced point information. Overall, there are two ways for data representation in this paradigm: one detects and locates 3-D objects directly from point clouds [118]; another first converts 3-D points into regular grids, such as voxel grids or BEV images and front views, and then utilizes architectures in 2-D detectors to extract objects from images, the 2-D detection results are finally back projected into 3-D space for final 3-D object location estimation [50]. Fig. 8 shows the representative frameworks of the above-listed data representations for 3-D object detection.

*1) 3-D Objects Detection (Localization) From Point Clouds:* The challenges for 3-D object detection from sparse and large-scale point clouds are concluded as follows.

1) The detected objects only occupy a very limited amount of the whole input data.
2) The 3-D object centroid can be far from any surface point thus hard to regress accurately in one step. As LiDAR sensors only capture surfaces of objects, 3-D object centers are likely to be in empty space, far away from any point [42].
3) The incompleteness of 3-D object shapes.

Thus, to solve the above-mentioned problems, a common procedure of 3-D object detection and localization is composed of the following processes: first, the whole scene is roughly segmented, and then the coarse location of interest object is approximately proposed; second, the feature for each proposed region is extracted; finally, the localization and the object class are predicted through a bounding box prediction network [118], [119].

In [119], the PointNet++ [12] is applied to generate per-point feature within the whole input point clouds. Different from [118], each point is viewed as an effective proposal, which preserves the localization information. Then, the localization and detection prediction is conducted based on the

TABLE IV
SEGMENTATION RESULTS ON SEMANTIC3D REDUCED-8 DATA SET

| Method | Input | Backbone | IoU | | | | | | | | mIoU | OA (%) | Highlights |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | IoU1 | IoU2 | IoU3 | IoU4 | IoU5 | IoU6 | IoU7 | IoU8 | | | |
| SPGraph [105] | Graph & point cloud | PointNet [10] | 0.974 | 0.926 | 0.879 | 0.44 | 0.932 | 0.31 | 0.635 | 0.762 | 0.732 | 94.0 | 3D point clouds are represented as a set of interconnected superpoints; Solve the big data challenge and the unevenly distributed point density problem. |
| MSDeepVoxNet [59] | voxels | VGG16 [87] | 0.83 | 0.672 | 0.838 | 0.367 | 0.924 | 0.313 | 0.500 | 0.782 | 0.653 | 88.4 | Extract multi-scale local features in a multi-scale neighborhood. |
| RF_MSSF [104] | point cloud | Random forest [117] | 0.876 | 0.803 | 0.818 | 0.364 | 0.922 | 0.241 | 0.426 | 0.566 | 0.627 | 90.3 | Define multi-scale neighborhoods in point clouds to extract features. |
| SEGCloud [113] | voxels | FCNN | 0.839 | 0.66 | 0.86 | 0.405 | 0.911 | 0.309 | 0.275 | 0.643 | 0.613 | 88.1 | Refine the labels generated at the voxel level for each point using Trilinear Interpolation; a FC CRF is connected with the FCNN to improve the segmentation result. |
| SnapNet [11] | images | CNN | 0.82 | 0.773 | 0.797 | 0.229 | 0.911 | 0.184 | 0.373 | 0.644 | 0.591 | 88.6 | Generate RGB and depth images from point cloud; Use CNN to conduct a pixel-wise labeling of each pair of 2D snapshot; Back-projection of the label predictions in the 3D space. |
| DeePr3SS [107] | images | VGG16 [87] | 0.856 | 0.832 | 0.742 | 0.324 | 0.897 | 0.185 | 0.251 | 0.592 | 0.585 | 88.9 | Generate view images from point clouds with a spread function into the image plane. Depth value and feature vectors of each projected point are stored and input to VGG16 for segmentation. |

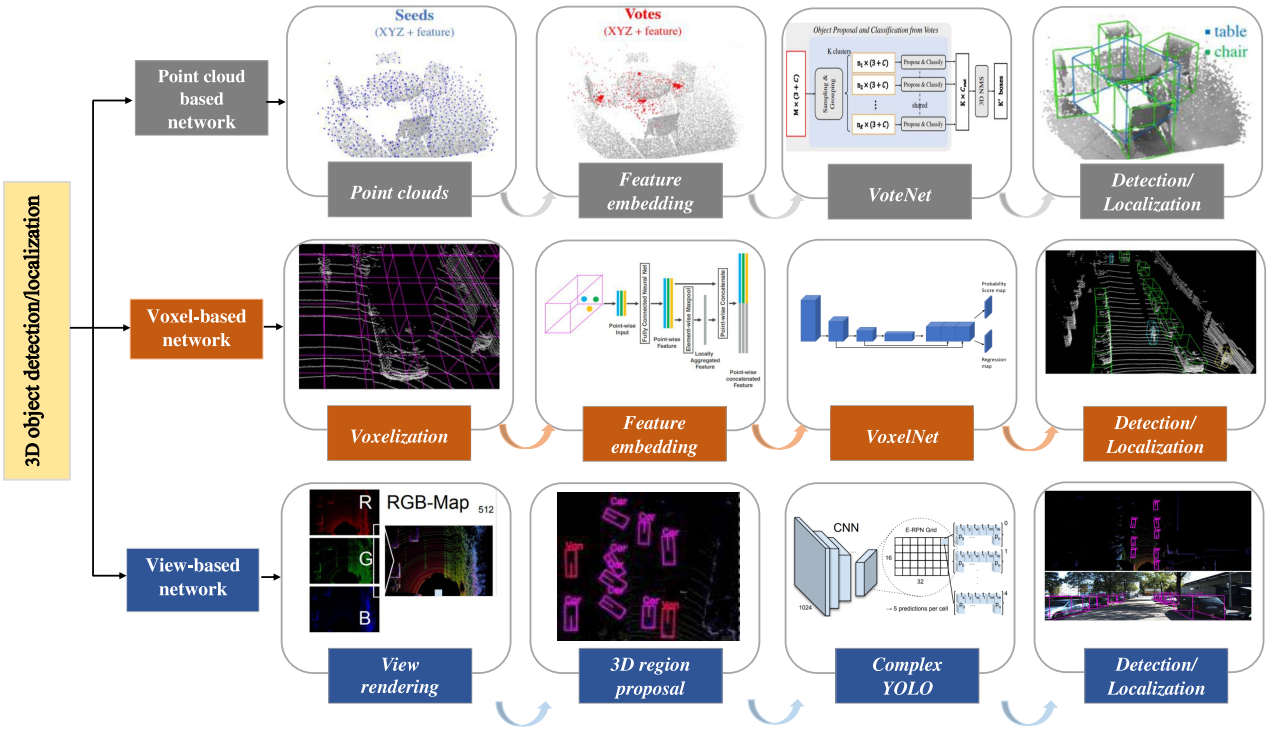



Fig. 8. DL architectures on 3-D object detection/localization with three different data representations: point cloud-based networks represented by VoteNet [42], voxel-based networks represented by VoxelNet [13], and view-based networks represented by ComplexYOLO [116].

extracted point-based proposal features and local neighbor context information captured by increasing receptive fields and input point features. This network preserves the accurate localization information but has a high computation cost for directly operating on point sets.

In [118], 3-D CNN with three Conv layers and multiple FC layers is applied to learn features of objects. Then, an intelligent eye window (EW) algorithm is applied to the scene. The label of the point that belongs to the EW is predicted using the pretrained 3-D CNN. The evaluation result is then inputted to the deep Q-network (DQN) to adjust the size and position of EW. Then, the new EW is evaluated by 3-D CNN and DQN until the EW only contains one object. EW can reshape the bounding box size and change the window center

automatically, which is suitable for objects with different scales. Once the position of the object is located, the object in the input window is predicted with learned features. In [118], the object features are extracted based on 3-D CNN models and then fed into the residual RNN [120] for category labeling.

Qi *et al.* [42] proposed the VoteNet for 3-D object detection based on Hough voting. The raw point clouds are input to PointNet++ [12] to learn point features. Based on these features, a group of seed points is sampled, and votes are generated from their neighbor features. These seeds are then gathered to cluster the object centers and generate bounding box proposals for a final decision. Compared with the above-mentioned two architectures, VoteNet can localize the object center with high accuracy. However, such a voting scheme is only suitable for objects without large orientation variances.

*2) 3-D Objects Detection (Localization) From Regular Voxel Grid:* To better exploit CNNs, some approaches voxelize the 3-D space into voxel grids, which are represented by scalar values, such as occupancy or vector data, extracted from voxels [8]. In [121] and [122], the 3-D space is first split into grids with a fixed size, and then each occupied cell is converted into a fixed-dimensional feature vector. Nonoccupied cells without any points are represented with zero feature vectors. A binary occupancy and the mean and the variance of the reflectance, and three shape factors are used to describe the feature vector. For simplicity, in [14], the grids are represented by a 4-D array with length, width, height, and channels. The binary value of one channel is used to represent the observation status of points in the corresponding grid. Zhou and Tuzel [13] voxelized the 3-D point clouds along with the $XYZ$ coordinates with the predefined distance and grouped points in grids. Then, a voxel feature encoding (VFE) layer is proposed to achieve the interpoint interaction within a voxel, by combining per-point features and local neighbor features. The combination of multiscale VFE layers enables this architecture to learn effective features from the local shape information.

The voting scheme is adopted in [121] and [122] to perform a sparse convolution on the voxelized grids. These grids, weighted by the convolution kernels and their surrounding cells in the receptive field, accumulate the votes from their neighbors by flipping the CNN kernel along each dimension. Finally, the voting scores for potential interest objects are predicted. Based on this voting scheme, Engelcke *et al.* [122] then used a ReLU nonlinearity to produce a novel sparse 3-D representation of these grids. This process is iterated and stacked in conventional CNN operations and finally output the predicting scores for each proposal. However, the voting scheme has high computation during voting. Thus, modified region proposal networks (RPNs) are employed by [13] in object detection to reduce computation. This RPN is composed of three blocks of Conv layers, which are used to downsample filter features and upsample the input feature map to produce a probability score map, and a regression map for object detection and localization.

*3) 3-D Objects Detection (Localization) From 2-D Views:* Some approaches also project LiDAR point clouds into 2-D views. Such approaches are mainly composed of those two steps: first is the projection of 3-D points; second is the object detection from projected images. There are several types of view generation methods to project 3-D points into 2-D images: BEV images [43], [116], [123], [124], front view images [123], spherical projections [50], and cylindrical projections [9].

Different from [50], in [43], [116], [123], and [124], the point cloud data are split into grids with fixed sizes. Then, these grids are converted to a BEV image with corresponding three channels which encodes height, intensity, and density information. Considering the efficiency and performance, only the maximum height, the maximum intensity, and the normalized density among the grids are converted to a single-BEV RGB map [116]. In [125], only the maximum, the median, and the minimum height values are selected to represent the channels of the BEV image to exploit conventional 2-D RGB detectors without modification. Dewan *et al.* [16] selected range, intensity, and height values to represent three channels. In [8], the feature representation for each BEV pixel is composed of occupancy and reflectance values.

However, due to the sparsity of point clouds, the projection of point clouds to the 2-D image plane produces a sparse 2-D point map. Thus, Chen *et al.* [123] added front view representation to compensate for the missing information in BEV images. The point clouds are projected to a cylinder plane to produce dense front view images. In order to keep the 3-D spatial information during projection, points are projected at multiview angles which are evenly selected on a sphere [50]. Pang and Neumann [50] first split 3-D points into cells with a fixed size. Then, the scene is sampled to generate multiview images to construct positive and negative training samples. The benefit of this operation is that the spatial relationship and the feature of the scene can be better exploited. However, this model is not robust to a new scene and cannot learn new features from a constructed data set.

As for 2-D object detectors, there exist enormous compelling deep models, such as VGG-16 [87], faster R-CNN [126]. In [23], a comprehensive survey of 2-D detectors for object detection is concluded.

*4) Evaluation on 3-D Objects Localization and Detection:* In order to compare 3-D objects localization and detection deep models, KITTI BEV benchmark and KITTI 3-D object detection benchmark [60] are selected. As reported in [60], all nonoccluded and weakly occluded (<20%) objects which are neither truncated nor smaller than 40 pixels in height are evaluated. Truncated or occluded objects are not counted as false positives. Only a bounding box overlap of at least 50% results for pedestrian and cyclist, and 70% results for the car are considered for detection, localization, and orientation estimation measurements. Besides, this benchmark classifies the difficulties of tasks into three types: easy, moderate, and hard.

Both the accuracy and execution time are compared to evaluate these algorithms because detection and localization in real time are crucial for AVs [127]. For the localization task, the KITTI BEV benchmark is chosen as the evaluation benchmark, and the comparison results are shown in Table V. The 3-D detection is evaluated on the KITTI 3-D object detection benchmark. Table V shows the run time and the average precision ($AP_{3D}$) on the validation set. For each bounding box

TABLE V
3-D CAR LOCALIZATION PERFORMANCE ON KITTI BEV BENCHMARK: AVERAGE PRECISION ($AP_{loc}$ [%])

| Method | Input | Times (s) | GPUs | Evaluation on AP (%) | | | | | | | | | | | | | | Highlights |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | object detection | | | | | | | | | object localization | | | | | |
| | | | | 0.25 | | | 0.5 | | | 0.7 | | | 0.5 | | | 0.7 | | |
| | | | | E | M | H | E | M | H | E | M | H | E | M | H | E | M | H |
| VeloFCN [14] | voxels | 1 | N/A | 89.0 | 81.1 | 75.9 | 67.9 | 57.6 | 52.6 | 15.2 | 13.7 | 16.0 | 79.7 | 63.8 | 62.8 | 40.1 | 32.1 | 30.5 | The voxelized grids are represented by length, width, height, and channels; The binary value of one channel is used to represent the observation status of points in corresponding grids. |
| DOBEM [125] | images | 0.6 | Titan X | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | 79.3 | 80.2 | 80.1 | 54.9 | 60.1 | 60.9 | The maximum, median and minimum height values are selected to represent the BEV image channels to exploit existing 2D RGB detectors. |
| MV3D [123] | images | 0.36 | Titan X | 96.5 | 89.6 | 88.9 | 96.0 | 89.1 | 88.4 | 71.3 | 62.7 | 56.6 | 96.3 | 89.4 | 88.7 | 86.6 | 78.1 | 76.7 | The point clouds are projected to a cylinder plane to produce dense front view images. |
| VoxelNet [13] | voxels | 0.23 | Titan X | N/A | N/A | N/A | N/A | N/A | N/A | 82.0 | 65.5 | 62.9 | N/A | N/A | N/A | 89.6 | 84.8 | 78.6 | Voxelize the 3D point clouds along XYZ coordinates with predefined distance and group points in each grid; Then a voxel feature encoding layer is proposed to achieve inter-point interaction within a voxel, by combining per-point features and local neighbor features. |
| RT3D [127] | point cloud | 0.09 | Titan X | 89.5 | 81.0 | 81.2 | 89.0 | 80.6 | 80.9 | 72.9 | 61.6 | 64.4 | 89.4 | 80.9 | 81.2 | 88.3 | 79.9 | 80.4 | Propose a pre-RoI-pooling convolution technique that moves a majority of the convolution operations to the RoI pooling. |

overlap, only 3-D IoU exceeds 0.25/0.5/0.7 is considered as a valid detection box and 0.5/0.7 for a localization box [127].

### C. 3-D Object Classification

Semantic object classification/recognition is crucial for safe and reliable driving of AVs in unstructured and uncontrolled real-world environments [67]. Existing 3-D object detection are mainly focus on CAD data (e.g., ModelNet40 [30]) or RGB-D data (e.g., NYUv2 [128]). However, these data have uniform point distribution, complete shapes, limited noise, occlusion, and background clutter, which pose limit challenges for 3-D classification compared with LiDAR point clouds [10], [12], [129]. Those compelling deep architectures applied on CAD data have been analyzed in the form of four types of data representations in Section III. In this part, we mainly focus on the LiDAR data-based deep models for the classification task.

*1) Volumetric Architectures:* The voxelization of point clouds depends on the data spatial resolution, orientation, and the origin [67]. This operation which can provide enough recognizable information but not increase the computation cost is crucial for DL models. Thus, for LiDAR data, a voxel with spatial resolution, such as $(0.1\ m)^3$ is adopted in [67] to voxelize the input points. Then, for each voxel, binary occupancy grid, density grid, and hit grid are calculated to estimate its occupancy. The input layer, Conv layer, pooling layer, and FC layer are combined to exploit the spatial structure among data and extract global features via pooling. However, the FC layer produces high computation cost and loses the spatial information between voxels. In [130], based on the VoxNet [67], 3-D voxel grids are input to two Conv layers with 3-D filters followed by two FC layers. Different from other category-level classification tasks, this task is treated as a multitask problem, where the orientation estimation and the class label prediction are processed parallel.

For simplicity and efficiency, Zhi *et al.* [93] and Ma *et al.* [131] adopted the binary grid of [67] to reduce the computation cost. However, they only consider the voxels inside the surface, ignoring the difference between unknown and free space. Normal vectors, which contain geo-local position and orientation information, have been demonstrated stronger than the binary grid in [132]. Similar to [130], the classification is treated as two tasks: the voxel object class predicting and the orientation prediction. To extract local and global features, there are two subtasks in the first task: the first is to predict the object label referencing the whole input shape while the second predicts the object label with part of the shape. The orientation prediction is proposed to exploit the orientation augmentation scheme.

*2) Multiview Architectures:* The merit of view-based methods is their ability to exploit both local and global spatial relationships among points. Luo *et al.* [45] designed the three feature descriptors to extract local and global features from point clouds: the horizontal geometric structure, the vertical information, and the complete spatial information. To better leverage the multiview data representations, You *et al.* [91] integrated the merits of point cloud and multiview data to achieve better results than MVCNN [76] in 3-D classification. Besides, the high-level features extracted from view representations based on MVCNN [76] are embedded with an attention fusion scheme to compensate the local features extracted from point clouds. Such attention-aware features are proved efficient in representing discriminative information of 3-D data.

However, for different objects, the view generation process varies because the special attributes of objects can contribute to computation saving and accuracy improving. For example, in road marking extraction tasks, the elevation derived mainly from $Z$ coordinate contributes little to the algorithm. But the road surface is actually a 2-D structure. As a result,

TABLE VI
3-D CLASSIFICATION PERFORMANCE ON THE SYDNEY URBAN OBJECTS DATA SET [45]

| Method | Input | $F_1$ score (%) | Highlights |
|---|---|---|---|
| VoxNet [67] | voxels | 72.0 | The input points are voxelized with spatial resolution; Binary occupancy grid, density grid, hit grid are calculated to estimate each voxel occupancy. |
| BV-CNNs [131] | voxels | 75.5 | Transform the inputs and weights in FC layers to binary values. |
| ORION [130] | voxels | 77.8 | Category level classification task is treated as the orientation estimation and class label prediction. |
| JointNet [45] | images | 74.9 | Horizontal geometric structure, vertical information, complete spatial information are proposed to extract point features. |

Wen *et al.* [47] directly projected 3-D point clouds onto a horizontal plane and girded it as a 2-D image. Luo *et al.* [45] input the acquired three-view descriptors separately to capture low-level features. Then, high-level features are learned by a convolutional operation based on the input features. Finally, the prediction scores are fused as output. The well-designed view descriptors help the network achieve compelling results in object classification tasks.

Wen *et al.* [47] proposed a modified U-net model to classify road markings. The point clouds are first mapped into the intensity images. Then, a hierarchical U-net module is applied to classify road markings by multiscale clustering via CNNs. Due to such downsampling and upsampling operation is hard to preserve the fine-grained patterns, a GAN network is adopted to reshape small-size road markings, broken lane lines, and missing markings considering the expert context knowledge. This architecture exploits the efficiency of U-net and completeness of GAN to classify the road markings with high efficiency and accuracy.

*3) Evaluation on 3-D Objects Classification:* There is limited published LiDAR point cloud benchmark specific for the 3-D object classification task. Thus, the Sydney Urban Objects data set is selected due to the performance of several state-of-the-art methods are available. The $F_1$ score is used to evaluate these published algorithms [45], as shown in Table VI.

## VI. RESEARCH CHALLENGES AND OPPORTUNITIES

DL architectures developed in recent five years using LiDAR point clouds have made significant success in the field of autonomous driving detailing for 3-D segmentation, detection, and classification tasks. However, there still exists a huge gap between cutting-edge results and human-level performance. Although there is much work to be done, we mainly summarize the remaining challenges specific for data, deep architectures, and tasks as follows.

*1) Multisource Data Fusion:* To compensate the absence of 2-D semantic, textual and incomplete information in 3-D points, imagery, LiDAR point clouds, and radar data can be fused to provide accurate, geo-referenced, and information-rich cues for AVs' navigation and decision making [133]. Besides, there also exists a fusion between data acquired by low-end LiDAR (e.g., Velodyne HDL-16E) and high-end LiDAR (e.g., Velodyne HDL-64E) sensors. However, there exist several challenges in fusing these data. The first is the sparsity of point clouds causes the inconsistent and missing data when fusing multisource data. The second is that the existing data fusion scheme using DL knowledge is processed in a separate line, which is not an end-to-end scheme [41], [119], [134].

*2) Robust Data Representation:* The unstructured and unordered data format [10], [12] poses a great challenge for robust 3-D DL applications. Although there are several effective data representations, such as voxels [67], point clouds [10], [12], graphs [74], [129], 2-D views [78], or novel 3-D data representations [135]–[137], there has not yet agreed on a robust and memory-efficient 3-D data representation. For example, although voxels solve the ordering problem, the computation cost increases cubically with the increment of voxel resolution [30], [67]. As for point clouds and graphs, the permutation invariance and the computation capability limit the processable quantity of points, which inevitably constrains the performance of the deep models [10], [74].

*3) Effective and More Efficient Deep Frameworks:* Due to the limitation of memory and computation facilities of the platform embedded in AVs, effective and efficient DL architectures are crucial for the wide application of automated AV systems. Although there are significant improvements in 3-D DL models, such as PointNet [10], PointNet++ [12], PointCNN [71], DGCNN [74], RotationNet [78], and other work [52], [138]–[140]. Limited models can achieve robust real-time segmentation, detection, and classification tasks. Researches should focus on lightweight and compact architecture designing.

*4) Context Knowledge Extraction:* Due to the sparsity of point clouds and incompleteness of scanned objects, detailed context information for objects is not fully exploited. For example, the semantic contexts in traffic signs are crucial cues for AVs navigation, but existing deep models cannot extract such information completely from point clouds. Those approaches [141]–[143] have demonstrated significant improvements in context information extraction using the multi-scale feature fusion strategy. Besides, GAN [47] can be utilized to improve the completeness of 3-D point clouds. However, these frameworks cannot solve the sparsity and incompleteness problems for context information extraction in an end-to-end trainable way.

*5) Multitask Learning:* The approaches related to LiDAR point clouds for AVs consist of several tasks, such as scene segmentation, object detection (e.g., cars, pedestrians, traffic lights, and so on), and classification (e.g., road markings and traffic signs). All these results are commonly fused together and reported to a decision system for final control [1]. However, there are a few DL architectures combining these multiple LiDAR point cloud tasks together [15], [130]. Thus, the inherent information among them is not fully exploited and used to generalize better models with less computation.

*6) Weakly Supervised/Unsupervised Learning:* The existing state-of-art deep models are commonly constructed under supervised modes using labeled data with 3-D objects bounding boxes or per-point segmentation masks [8],

[74], [119]. However, there are some limitations for fully supervised models. First is the limited availability of high-quality, large-scale, and enormous general object data sets and benchmarks. Second is the ineffective, fully supervised model generalization capability to unseen or untrained objects. Weakly supervised [144] or unsupervised learning [145], [146] should be developed to increase the model's generalization and solve the data absence problem.

## VII. Conclusion

In this article, we have provided a systematic review of the state-of-the-art DL architectures using LiDAR point clouds in the field of autonomous driving for specific tasks, such as segmentation, detection, and classification. Milestone 3-D deep models and 3-D DL applications on these three tasks have been summarized and evaluated with merits and demerits comparison. Research challenges and opportunities were listed to advance the potential development of DL in the field of autonomous driving.

## Acknowledgment

The authors would like to thank the Professors José Marcato Junior and Wesley Nunes Gonçalves for their carefully proofreading. They would also like to thank anonymous reviewers for their insightful comments and suggestions.

## References

[1] J. Janai, F. Güney, A. Behl, and A. Geiger, "Computer vision for autonomous vehicles: Problems, datasets and state of the art," 2017, *arXiv:1704.05519*. [Online]. Available: http://arxiv.org/abs/1704.05519

[2] J. Levinson *et al.*, "Towards fully autonomous driving: Systems and algorithms," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2011, pp. 163–168.

[3] J. Van Brummelen, M. O'Brien, D. Gruyer, and H. Najjaran, "Autonomous vehicle perception: The technology of today and tomorrow," *Transp. Res. C, Emerg. Technol.*, vol. 89, pp. 384–406, Apr. 2018.

[4] X. Huang *et al.*, "The ApolloScape dataset for autonomous driving," in *Proc. IEEE CVPR Workshops*, Jun. 2018, pp. 954–960.

[5] R. P. D. Vivacqua, M. Bertozzi, P. Cerri, F. N. Martins, and R. F. Vassallo, "Self-localization based on visual lane marking maps: An accurate low-cost approach for autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 2, pp. 582–597, Feb. 2018.

[6] F. Remondino, "Heritage recording and 3D modeling with photogrammetry and 3D scanning," *Remote Sens.*, vol. 3, no. 6, pp. 1104–1138, May 2011.

[7] B. Wu, A. Wan, X. Yue, and K. Keutzer, "SqueezeSeg: Convolutional neural nets with recurrent CRF for real-time road-object segmentation from 3D LiDAR point cloud," in *Proc. IEEE ICRA*, May 2018, pp. 1887–1893.

[8] B. Yang, W. Luo, and R. Urtasun, "PIXOR: Real-time 3D object detection from point clouds," in *Proc. IEEE CVPR*, Jun. 2018, pp. 7652–7660.

[9] B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3D lidar using fully convolutional network," 2016, *arXiv:1608.07916*. [Online]. Available: http://arxiv.org/abs/1608.07916

[10] 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 CVPR*, Jul. 2017, pp. 652–660.

[11] A. Boulch, B. Le Saux, and N. Audebert, "Unstructured point cloud semantic labeling using deep segmentation networks," in *Proc. 3DOR*, 2017, p. 7.

[12] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5099–5108.

[13] Y. Zhou and O. Tuzel, "VoxelNet: End-to-end learning for point cloud based 3D object detection," in *Proc. IEEE CVPR*, Jun. 2018, pp. 4490–4499.

[14] B. Li, "3D fully convolutional network for vehicle detection in point cloud," in *Proc. IEEE/RSJ IROS*, Sep. 2017, pp. 1513–1518.

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

[16] A. Dewan, G. L. Oliveira, and W. Burgard, "Deep semantic classification for 3D LiDAR data," in *Proc. IEEE/RSJ IROS*, Sep. 2017, pp. 3544–3549.

[17] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[18] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.

[19] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, "Deep learning for visual understanding: A review," *Neurocomputing*, vol. 187, pp. 27–48, Apr. 2016.

[20] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Comput. Intell. Neurosci.*, vol. 2018, pp. 1–13, Feb. 2018.

[21] L. Zhang, L. Zhang, and B. Du, "Deep learning for remote sensing data: A technical tutorial on the state of the art," *IEEE Geosci. Remote Sens. Mag.*, vol. 4, no. 2, pp. 22–40, Jun. 2016.

[22] X. X. Zhu *et al.*, "Deep learning in remote sensing: A comprehensive review and list of resources," *IEEE Geosci. Remote Sens. Mag.*, vol. 5, no. 4, pp. 8–36, Dec. 2017.

[23] L. Liu *et al.*, "Deep learning for generic object detection: A survey," 2018, *arXiv:1809.02165*. [Online]. Available: http://arxiv.org/abs/1809.02165

[24] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 11, pp. 3212–3232, Nov. 2019.

[25] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez, "A review on deep learning techniques applied to semantic segmentation," 2017, *arXiv:1704.06857*. [Online]. Available: http://arxiv.org/abs/1704.06857

[26] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, Apr. 2017.

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

[28] A. Ioannidou, E. Chatzilari, S. Nikolopoulos, and I. Kompatsiaris, "Deep learning advances in computer vision with 3D data: A survey," *ACM Comput. Surv.*, vol. 50, no. 2, p. 20, 2017.

[29] E. Ahmed *et al.*, "A survey on deep learning advances on different 3D data representations: A survey," 2018, *arXiv:1808.01462*. [Online]. Available: http://arxiv.org/abs/1808.01462

[30] Z. Wu *et al.*, "3D ShapeNets: A deep representation for volumetric shapes," in *Proc. IEEE CVPR*, Jun. 2015, pp. 1912–1920.

[31] L. Ma, Y. Li, J. Li, C. Wang, R. Wang, and M. Chapman, "Mobile laser scanned point-clouds for road object detection and extraction: A review," *Remote Sens.*, vol. 10, no. 10, p. 1531, Sep. 2018.

[32] H. Guan, J. Li, S. Cao, and Y. Yu, "Use of mobile LiDAR in road information inventory: A review," *Int. J. Image Data Fusion*, vol. 7, no. 3, pp. 219–242, Jul. 2016.

[33] E. Che, J. Jung, and M. Olsen, "Object recognition, segmentation, and classification of mobile laser scanning point clouds: A state of the art review," *Sensors*, vol. 19, no. 4, p. 810, Feb. 2019.

[34] R. Wang, J. Peethambaran, and D. Chen, "LiDAR point clouds to 3-D urban models: A review," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 11, no. 2, pp. 606–627, Feb. 2018.

[35] X.-F. Hana, J. S. Jin, J. Xie, M.-J. Wang, and W. Jiang, "A comprehensive review of 3D point cloud descriptors," 2018, *arXiv:1802.02297*. [Online]. Available: http://arxiv.org/abs/1802.02297

[36] E. Arnold, O. Y. Al-Jarrah, M. Dianati, S. Fallah, D. Oxtoby, and A. Mouzakitis, "A survey on 3D object detection methods for autonomous driving applications," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 10, pp. 3782–3795, Oct. 2019.

[37] 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.

[38] M. Treml *et al.*, "Speeding up semantic segmentation for autonomous driving," in *Proc. MLITS, NIPS Workshop*, vol. 1, 2016, p. 5.

[39] A. Nguyen and B. Le, "3D point cloud segmentation: A survey," in *Proc. RAM*, Nov. 2013, pp. 225–230.

[40] Q. Huang, W. Wang, and U. Neumann, "Recurrent slice networks for 3D segmentation of point clouds," in *Proc. IEEE CVPR*, Jun. 2018, pp. 2626–2635.

[41] 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 CVPR*, Jun. 2018, pp. 918–927.

[42] C. R. Qi, O. Litany, K. He, and L. J. Guibas, "Deep Hough voting for 3D object detection in point clouds," 2019, *arXiv:1904.09664*. [Online]. Available: http://arxiv.org/abs/1904.09664

[43] J. Beltrán, C. Guindel, F. M. Moreno, D. Cruzado, F. García, and A. De La Escalera, "BirdNet: A 3D object detection framework from LiDAR information," in *Proc. ITSC*, Nov. 2018, pp. 3517–3523.

[44] A. Kundu, Y. Li, and J. M. Rehg, "3D-RCNN: Instance-level 3D object reconstruction via render-and-compare," in *Proc. IEEE CVPR*, Jun. 2018, pp. 3559–3568.

[45] Z. Luo, J. Li, Z. Xiao, Z. G. Mou, X. Cai, and C. Wang, "Learning high-level features by fusing multi-view representation of MLS point clouds for 3D object recognition in road environments," *ISPRS J. Photogramm. Remote Sens.*, vol. 150, pp. 44–58, Apr. 2019.

[46] Z. Wang *et al.*, "A multiscale and hierarchical feature extraction method for terrestrial laser scanning point cloud classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 5, pp. 2409–2425, May 2015.

[47] C. Wen, X. Sun, J. Li, C. Wang, Y. Guo, and A. Habib, "A deep learning framework for road marking extraction, classification and completion from mobile laser scanning point clouds," *ISPRS J. Photogramm. Remote Sens.*, vol. 147, pp. 178–192, Jan. 2019.

[48] T. Hackel, J. D. Wegner, and K. Schindler, "Joint classification and contour extraction of large 3D point clouds," *ISPRS J. Photogramm. Remote Sens.*, vol. 130, pp. 231–245, Aug. 2017.

[49] B. Kumar, G. Pandey, B. Lohani, and S. C. Misra, "A multi-faceted CNN architecture for automatic classification of mobile LiDAR data and an algorithm to reproduce point cloud samples for enhanced training," *ISPRS J. Photogramm. Remote Sens.*, vol. 147, pp. 80–89, Jan. 2019.

[50] G. Pang and U. Neumann, "3D point cloud object detection with multi-view convolutional neural network," in *Proc. IEEE ICPR*, Dec. 2016, pp. 585–590.

[51] A. Tagliasacchi, H. Zhang, and D. Cohen-Or, "Curve skeleton extraction from incomplete point cloud," *ACM Trans. Graph.*, vol. 28, no. 3, p. 71, 2009.

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

[53] H. Huang, D. Li, H. Zhang, U. Ascher, and D. Cohen-Or, "Consolidation of unorganized point clouds for surface reconstruction," *ACM Trans. Graph.*, vol. 28, no. 5, p. 176, Dec. 2009.

[54] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, Sep. 2013.

[55] K. Jo, J. Kim, D. Kim, C. Jang, and M. Sunwoo, "Development of autonomous car—Part II: A case study on the implementation of an autonomous driving system based on distributed architecture," *IEEE Trans. Aerosp. Electron.*, vol. 62, no. 8, pp. 5119–5132, Aug. 2015.

[56] 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*. [Online]. Available: http://arxiv.org/abs/1704.03847

[57] D. Munoz, J. A. Bagnell, N. Vandapel, and M. Hebert, "Contextual classification with functional max-margin Markov networks," in *Proc. IEEE CVPR*, Jun. 2009, pp. 975–982.

[58] 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.

[59] X. Roynard, J.-E. Deschaud, and F. Goulette, "Classification of point cloud scenes with multiscale voxel deep network," 2018, *arXiv:1804.03583*. [Online]. Available: http://arxiv.org/abs/1804.03583

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

[61] M. De Deuge, A. Quadros, C. Hung, and B. Douillard, "Unsupervised feature learning for classification of outdoor 3D scans," in *Proc. ACRA*, vol. 2, 2013, p. 1.

[62] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes challenge: A retrospective," *Int. J. Comput. Vis.*, vol. 111, no. 1, pp. 98–136, Jan. 2015.

[63] L. Yan, Z. Li, H. Liu, J. Tan, S. Zhao, and C. Chen, "Detection and classification of pole-like road objects from mobile LiDAR data in motorway environment," *Opt. Laser Technol.*, vol. 97, pp. 272–283, Dec. 2017.

[64] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 year, 1000 km: The Oxford RobotCar dataset," *Int. J. Robot. Res.*, vol. 36, no. 1, pp. 3–15, Jan. 2017.

[65] L. Zhang, Z. Li, A. Li, and F. Liu, "Large-scale urban point cloud labeling and reconstruction," *ISPRS J. Photogramm. Remote Sens.*, vol. 138, pp. 86–100, Apr. 2018.

[66] X. Chen, K. Kundu, Y. Zhu, H. Ma, S. Fidler, and R. Urtasun, "3D object proposals using stereo imagery for accurate object class detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 5, pp. 1259–1272, May 2018.

[67] D. Maturana and S. Scherer, "VoxNet: A 3D convolutional neural network for real-time object recognition," in *Proc. IEEE/RSJ IROS*, Sep. 2015, pp. 922–928.

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

[69] G. Riegler, A. O. Ulusoy, and A. Geiger, "OctNet: Learning deep 3D representations at high resolutions," in *Proc. IEEE CVPR*, Jul. 2017, pp. 3577–3586.

[70] R. Klokov and V. Lempitsky, "Escape from cells: Deep KD-networks for the recognition of 3D point cloud models," in *Proc. IEEE ICCV*, Oct. 2017, pp. 863–872.

[71] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "PointCNN: Convolution on X-transformed points," in *Proc. NeurIPS*, 2018, pp. 820–830.

[72] L. Yi, H. Su, X. Guo, and L. Guibas, "SyncSpecCNN: Synchronized spectral CNN for 3D shape segmentation," in *Proc. IEEE CVPR*, Jul. 2017, pp. 2282–2290.

[73] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *Proc. IEEE CVPR*, Jul. 2017, pp. 3693–3702.

[74] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," 2018, *arXiv:1801.07829*. [Online]. Available: http://arxiv.org/abs/1801.07829

[75] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," 2017, *arXiv:1710.10903*. [Online]. Available: http://arxiv.org/abs/1710.10903

[76] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *Proc. IEEE ICCV*, Dec. 2015, pp. 945–953.

[77] A. Dai and M. Nießner, "3DMV: Joint 3D-multi-view prediction for 3D semantic scene segmentation," in *Proc. ECCV*, 2018, pp. 452–468.

[78] A. Kanezaki, Y. Matsushita, and Y. Nishida, "RotationNet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints," in *Proc. IEEE CVPR*, Jun. 2018, pp. 5010–5019.

[79] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE CVPR*, Jun. 2015, pp. 3431–3440.

[80] G. Vosselman, B. G. H. Gorte, G. Sithole, and T. Rabbani, "Recognising structure in laser scanner point clouds," *ISPRS Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.*, vol. 46, no. 8, pp. 33–38, 2004.

[81] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

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

[83] A. Miller, V. Jain, and J. L. Mundy, "Real-time rendering and dynamic updating of 3-D volumetric data," in *Proc. GPGPU*, 2011, p. 8.

[84] C. Wang, B. Samari, and K. Siddiqi, "Local spectral graph convolution for point set feature learning," in *Proc. ECCV*, 2018, pp. 52–66.

[85] L. Wang, Y. Huang, Y. Hou, S. Zhang, and J. Shan, "Graph attention convolution for point cloud semantic segmentation," in *Proc. IEEE CVPR*, Jun. 2019, pp. 10296–10305.

[86] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural. Inf. Process. Syst.*, 2012, pp. 1097–1105.

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

[88] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE CVPR*, Jun. 2015, pp. 1–9.

[89] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE CVPR*, Jun. 2016, pp. 770–778.

[90] J.-C. Su, M. Gadelha, R. Wang, and S. Maji, "A deeper look at 3D shape classifiers," in *Proc. ECCV*, 2018, pp. 645–661.

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

[92] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

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

[94] S. Zhi, Y. Liu, X. Li, and Y. Guo, "Lightnet: A lightweight 3D convolutional neural network for real-time 3D object recognition," in *Proc. 3DOR*, 2017, pp. 9–16.

[95] A. Dai, D. Ritchie, M. Bokeloh, S. Reed, J. Sturm, and M. Niessner, "ScanComplete: Large-scale scene completion and semantic segmentation for 3D scans," in *Proc. IEEE CVPR*, Jun. 2018, pp. 4578–4587.

[96] J. Li, B. M. Chen, and G. H. Lee, "SO-Net: Self-organizing network for point cloud analysis," in *Proc. IEEE CVPR*, Jun. 2018, pp. 9397–9406.

[97] P. Huang, M. Cheng, Y. Chen, H. Luo, C. Wang, and J. Li, "Traffic sign occlusion detection using mobile laser scanning point clouds," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 9, pp. 2364–2376, Sep. 2017.

[98] H. Lei, N. Akhtar, and A. Mian, "Spherical convolutional neural network for 3D point clouds," 2018, *arXiv:1805.07872*. [Online]. Available: http://arxiv.org/abs/1805.07872

[99] F. Engelmann, T. Kontogianni, J. Schult, and B. Leibe, "Know what your neighbors do: 3D semantic segmentation of point clouds," in *Proc. ECCVW*, 2018, pp. 395–409.

[100] M. Weinmann, B. Jutzi, S. Hinz, and C. Mallet, "Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers," *ISPRS J. Photogramm. Remote Sens.*, vol. 105, pp. 286–304, Jul. 2015.

[101] E. Che and M. J. Olsen, "Fast ground filtering for TLS data via scanline density analysis," *ISPRS J. Photogramm. Remote Sens.*, vol. 129, pp. 226–240, Jul. 2017.

[102] A.-V. Vo, L. Truong-Hong, D. F. Laefer, and M. Bertolotto, "Octree-based region growing for point cloud segmentation," *ISPRS J. Photogramm. Remote Sens.*, vol. 104, pp. 88–100, Jun. 2015.

[103] R. B. Rusu and S. Cousins, "Point cloud library (PCL)," in *Proc. IEEE ICRA*, Jun. 2011, pp. 1–4.

[104] H. Thomas, F. Goulette, J.-E. Deschaud, B. Marcotegui, and Y. LeGall, "Semantic classification of 3D point clouds with multiscale spherical neighborhoods," in *Proc. 3DV*, Sep. 2018, pp. 390–398.

[105] L. Landrieu and M. Simonovsky, "Large-scale point cloud semantic segmentation with superpoint graphs," in *Proc. IEEE CVPR*, Jun. 2018, pp. 4558–4567.

[106] L. Wang, Y. Huang, J. Shan, and L. He, "MSNet: Multi-scale convolutional network for point cloud classification," *Remote Sens.*, vol. 10, no. 4, p. 612, Apr. 2018.

[107] F. J. Lawin, M. Danelljan, P. Tosteberg, G. Bhat, F. S. Khan, and M. Felsberg, "Deep projective 3D semantic segmentation," in *Proc. CAIP*, 2017, pp. 95–107.

[108] D. Rethage, J. Wald, J. Sturm, N. Navab, and F. Tombari, "Fully-convolutional point networks for large-scale point clouds," in *Proc. ECCV*, 2018, pp. 596–611.

[109] W. Wang, R. Yu, Q. Huang, and U. Neumann, "SGPN: Similarity group proposal network for 3D point cloud instance segmentation," in *Proc. IEEE CVPR*, Jun. 2018, pp. 2569–2578.

[110] H. Su *et al.*, "SPLATNet: Sparse lattice networks for point cloud processing," in *Proc. IEEE CVPR*, Jun. 2018, pp. 2530–2539.

[111] Z. Wang, L. Zhang, L. Zhang, R. Li, Y. Zheng, and Z. Zhu, "A deep neural network with spatial pooling (DNNSP) for 3-D point cloud classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 8, pp. 4594–4604, Aug. 2018.

[112] J. Huang and S. You, "Point cloud labeling using 3D convolutional neural network," in *Proc. ICPR*, Dec. 2016, pp. 2670–2675.

[113] L. Tchapmi, C. Choy, I. Armeni, J. Gwak, and S. Savarese, "SEGCloud: Semantic segmentation of 3D point clouds," in *Proc. 3DV*, Oct. 2017, pp. 537–547.

[114] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proc. IEEE Int. Conf. Mach. Learn.*, Madrid, Spain, Oct. 2001, pp. 282–289.

[115] R. Zhang, G. Li, M. Li, and L. Wang, "Fusion of images and point clouds for the semantic segmentation of large-scale 3D scenes based on deep learning," *ISPRS J. Photogramm. Remote Sens.*, vol. 143, pp. 85–96, Sep. 2018.

[116] M. Simony, S. Milzy, K. Amendey, and H.-M. Gross, "Complex-Yolo: An Euler-region-proposal for real-time 3D object detection on point clouds," in *Proc. ECCVW*, 2018, pp. 197–209.

[117] A. Liaw and M. Wiener, "Classification and regression by randomforest," *R News*, vol. 2, no. 3, pp. 18–22, 2002.

[118] L. Zhang and L. Zhang, "Deep learning-based classification and reconstruction of residential scenes from large-scale point clouds," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 4, pp. 1887–1897, Apr. 2018.

[119] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, "IPOD: Intensive point-based object detector for point cloud," 2018, *arXiv:1812.05276*. [Online]. Available: http://arxiv.org/abs/1812.05276

[120] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE CVPR*, Jun. 2018, pp. 8697–8710.

[121] D. Z. Wang and I. Posner, "Voting for voting in online point cloud object detection," in *Proc. RSS*, 2015, vol. 1, no. 3, pp. 15607–15610.

[122] 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 ICRA*, May 2017, pp. 1355–1361.

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

[124] 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 IROS*, Oct. 2018, pp. 1–8.

[125] S.-L. Yu, T. Westfechtel, R. Hamada, K. Ohno, and S. Tadokoro, "Vehicle detection and localization on bird's eye view elevation images using convolutional neural network," in *Proc. IEEE SSRR*, Oct. 2017, pp. 102–109.

[126] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.

[127] Y. Zeng *et al.*, "RT3D: Real-time 3-D vehicle detection in LiDAR point cloud for autonomous driving," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3434–3440, Oct. 2018.

[128] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *Proc. ECCV*, 2012, pp. 746–760.

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

[130] N. Sedaghat, M. Zolfaghari, E. Amiri, and T. Brox, "Orientation-boosted voxel nets for 3D object recognition," 2016, *arXiv:1604.03351*. [Online]. Available: http://arxiv.org/abs/1604.03351

[131] C. Ma, Y. Guo, Y. Lei, and W. An, "Binary volumetric convolutional neural networks for 3-D object recognition," *IEEE Trans. Instrum. Meas.*, vol. 68, no. 1, pp. 38–48, Jan. 2018.

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

[133] M. Liang, B. Yang, S. Wang, and R. Urtasun, "Deep continuous fusion for multi-sensor 3D object detection," in *Proc. ECCV*, 2018, pp. 641–656.

[134] D. Xu, D. Anguelov, and A. Jain, "PointFusion: Deep sensor fusion for 3D bounding box estimation," in *Proc. IEEE CVPR*, Jun. 2018, pp. 244–253.

[135] T. He *et al.*, "GeoNet: Deep geodesic networks for point cloud analysis," 2019, *arXiv:1901.00680*. [Online]. Available: http://arxiv.org/abs/1901.00680

[136] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3D reconstruction in function space," 2018, *arXiv:1812.03828*. [Online]. Available: http://arxiv.org/abs/1812.03828

[137] T. Le and Y. Duan, "PointGrid: A deep network for 3D shape understanding," in *Proc. IEEE CVPR*, Jun. 2018, pp. 9204–9214.

[138] J. Li, Y. Bi, and G. Hee Lee, "Discrete rotation equivariance for point cloud recognition," 2019, *arXiv:1904.00319*. [Online]. Available: http://arxiv.org/abs/1904.00319

[139] D. Worrall and G. Brostow, "Cubenet: Equivariance to 3D rotation and translation," in *Proc. ECCV*, 2018, pp. 567–584.

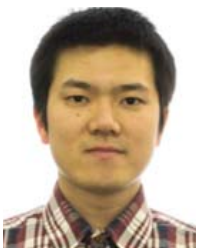
[140] K. Fujiwara, I. Sato, M. Ambai, Y. Yoshida, and Y. Sakakura, "Canonical and compact point cloud representation for shape classification," 2018, *arXiv:1809.04820*. [Online]. Available: http://arxiv.org/abs/1809.04820
[141] Z. Dong, B. Yang, F. Liang, R. Huang, and S. Scherer, "Hierarchical registration of unordered TLS point clouds based on binary shape context descriptor," *ISPRS J. Photogramm. Remote Sens.*, vol. 144, pp. 61–79, Oct. 2018.
[142] H. Deng, T. Birdal, and S. Ilic, "PPFNet: Global context aware local features for robust 3D point matching," in *Proc. IEEE CVPR*, Jun. 2018, pp. 195–205.
[143] S. Xie, S. Liu, Z. Chen, and Z. Tu, "Attentional ShapeContextNet for point cloud recognition," in *Proc. IEEE CVPR*, Jun. 2018, pp. 4606–4615.
[144] Z. J. Yew and G. H. Lee, "3DFeat-net: Weakly supervised local 3D features for point cloud registration," in *Proc. ECCV*, 2018, pp. 630–646.
[145] J. Sauder and B. Sievers, "Context prediction for unsupervised deep learning on point clouds," 2019, *arXiv:1901.08396*. [Online]. Available: http://arxiv.org/abs/1901.08396
[146] M. Shoef, S. Fogel, and D. Cohen-Or, "PointWise: An unsupervised point-wise feature learning network," 2019, *arXiv:1901.04544*. [Online]. Available: http://arxiv.org/abs/1901.04544

**Ying Li** (Graduate Student Member, IEEE) received the M.Sc. degree in remote sensing from Wuhan University, Wuhan, China, in 2017. She is currently pursuing the Ph.D. degree with the Mobile Sensing and Geodata Science Laboratory, Department of Geography and Environmental Management, University of Waterloo, Waterloo, ON, Canada.

Her research interests include autonomous driving, mobile laser scanning, intelligent processing of point clouds, geometric and semantic modeling, and augmented reality.

**Lingfei Ma** (Graduate Student Member, IEEE) received the B.Sc. and M.Sc. degrees in geomatics engineering from the University of Waterloo, Waterloo, ON, Canada, in 2015 and 2017, respectively, where he is currently pursuing the Ph.D. degree in photogrammetry and remote sensing with the Mobile Sensing and Geodata Science Laboratory, Department of Geography and Environmental Management.

His research interests include autonomous driving, mobile laser scanning, intelligent processing of point clouds, 3-D scene modeling, and machine learning.



**Zilong Zhong** (Member, IEEE) received the Ph.D. degree in systems design engineering specialized in machine learning and intelligence from the University of Waterloo, Waterloo, ON, Canada, in 2019.

He is a Post-Doctoral Fellow with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China. His research interests include computer vision, deep learning, and graph models and their applications involving large-scale image analysis.

**Fei Liu** received the B.Eng. degree from Yanshan University, Qinhuangdao, China, in 2011.

Since 2011, she has been working in the sectors of vehicular electronics and control, artificial intelligence, deep learning, field-programmable gate array (FPGA), advanced driver assistance systems, and automated driving. She is currently with Xilinx Technology Beijing, Ltd., Beijing, China, with a focus on the development of automated driving technologies and data centers.

**Michael A. Chapman** (Senior Member, IEEE) received the Ph.D. degree in photogrammetry from Laval University, Quebec City, Canada, in 1989.

He was a Professor with the Department of Geomatics Engineering, University of Calgary, Calgary, AB, Canada, for 18 years. He is currently a Professor of geomatics engineering with the Department of Civil Engineering, Ryerson University, Toronto, ON, Canada. He has authored or coauthored over 200 technical articles. His research interests include algorithms and processing methodologies for airborne sensors using global navigation satellite system (GNSS)/inertial measurement unit (IMU), geometric processing of digital imagery in industrial environments, terrestrial imaging systems for transportation infrastructure mapping, and algorithms and processing strategies for biometrology applications.

**Dongpu Cao** (Senior Member, IEEE) received the Ph.D. degree from Concordia University, Montreal, QC, Canada, in 2008.

He is currently the Canada Research Chair of driver cognition and automated driving, and also an Associate Professor and the Director of Waterloo Cognitive Autonomous Driving Laboratory, University of Waterloo, Waterloo, ON, Canada. His current research focuses on driver cognition, automated driving and cognitive autonomous driving. He has contributed more than 200 articles and 3 books.

Dr. Cao received the Society of Automotive Engineers (SAE) Arch T. Colwell Merit Award in 2012 and three best paper awards from the ASME and the IEEE conferences. He serves on the SAE Vehicle Dynamics Standards Committee and acts as the Co-Chair of the IEEE ITSS Technical Committee on Cooperative Driving. He serves as an Associate Editor for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, the IEEE/ASME TRANSACTIONS ON MECHATRONICS, the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, the IEEE/CAA JOURNAL OF AUTOMATICA SINICA, and the *ASME Journal of Dynamic Systems, Measurement, and Control*. He was a Guest Editor of *Vehicle System Dynamics* and the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS.

**Jonathan Li** (Senior Member, IEEE) received the Ph.D. degree in geomatics engineering from the University of Cape Town, Cape Town, South Africa, in 2000.

He is currently a Professor with the Department of Geography and Environmental Management and cross-appointed with the Department of Systems Design Engineering, University of Waterloo, Waterloo, ON, Canada. He has coauthored more than 400 publications, more than 230 of which were published in refereed journals, including the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING, the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, the IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING, the *International Society for Photogrammetry and Remote Sensing (ISPRS) Journal of Photogrammetry and Remote Sensing*, *Remote Sensing of Environment*, and leading artificial intelligence and remote sensing conferences, including the Computer Vision and Pattern Recognition (CVPR), the Association for the Advancement of Artificial Intelligence (AAAI), the International Joint Conferences on Artificial Intelligence (IJCAI), International Geoscience and Remote Sensing Symposium (IGARSS), and ISPRS. His research interests include information extraction from LiDAR point clouds and from Earth observation images.

Dr. Li was a recipient of the Outstanding Achievement in Mobile Mapping Technology Award in 2019 and the ISPRS Samuel Gamble Award in 2020.