3D-MiniNet: Learning a 2D Representation from Point Clouds for Fast and Efficient 3D LIDAR Semantic Segmentation

Iñigo Alonso[1] Luis Riazuelo[1] Luis Montesano[1,2] Ana C. Murillo[1]

Abstract— **LIDAR semantic segmentation is an essential task that provides 3D semantic information about the environment to robots. Fast and efficient semantic segmentation methods are needed to match the strong computational and temporal restrictions of many real-world robotic applications. This work presents 3D-MiniNet, a novel approach for LIDAR semantic segmentation that combines 3D and 2D learning layers. It first learns a 2D representation from the raw points through a novel projection which extracts local and global information from the 3D data. This representation is fed to an efficient 2D Fully Convolutional Neural Network (FCNN) that produces a 2D semantic segmentation. These 2D semantic labels are re-projected back to the 3D space and enhanced through a post-processing module. The main novelty in our strategy relies on the projection learning module. Our detailed ablation study shows how each component contributes to the final performance of 3D-MiniNet. We validate our approach on well known public benchmarks (SemanticKITTI and KITTI), where 3D-MiniNet gets state-of-the-art results while being faster and more parameter-efficient than previous methods.**

I. INTRODUCTION

Autonomous robotic systems use sensors to perceive the world around them. RGB cameras and LIDAR are very common due to the essential data they provide. One of the key building blocks of autonomous robots is semantic segmentation. Semantic segmentation assigns a class label to each LIDAR point or camera pixel. This detailed semantic information is essential for decision making in real-world dynamic scenarios. LIDAR semantic segmentation provides very useful information to autonomous robots when performing tasks such as Simultaneous Localization And Mapping (SLAM) [1], [2], autonomous driving [3] or inventory tasks [4], especially for identifying dynamic objects. In these scenarios, it is critical to have models that provide accurate semantic information in a fast and efficient manner, which is particularly challenging working with 3D LIDAR data. On one hand, the commonly called *point-based approaches* [5]–[7] tackle this problem directly executing 3D point-based operations, which is computationally expensive to operate at high frame rates. On the other hand, approaches that project the 3D information into a 2D image (*projection-based approaches*) are more efficient [3], [8]–[11] but do not exploit the raw 3D information. Recent results on fast [3] and parameter-efficient [12] semantic segmentation models are facilitating the adoption and use of semantic segmentation in real-world robotic applications [13], [14].

In this work, we present a novel fast and parameter-efficient approach for 3D LIDAR semantic segmentation that consists of three modules (as detailed later in Sec. III). The main contribution relies on our 3D-MiniNet module. 3D-MiniNet runs the following two steps: (1) First, it learns a 2D representation from the 3D point cloud (following previous works on 3D object detection [15]–[17]); (2) It computes the segmentation through a fast 2D fully convolutional neural network.

Our best configuration achieves state-of-the-art results in well known public benchmarks (SemanticKITTI benchmark [14] and KITTI dataset [18]) while being faster and more parameter efficient that prior work. Figure 1 shows how 3D-MiniNet achieves better precision-speed trade-off than previous methods on the SemanticKITTI benchmark. The main novelties, with respect to existing approaches, that facilitate these improvements are:

- An extension of MiniNet-v2 for 3D LIDAR semantic segmentation: 3D-MiniNet.
- Our novel projection module.
- A validation of 3D-MiniNet on the SemanticKITTI benchmark [14] and KITTI dataset [18].

Figure 1 shows how 3D-MiniNet achieves better precision-speed trade-off than previous methods on the SemanticKITTI benchmark.

The proposed projection module learns a rich 2D representation through different operations. It consists of four submodules: a context feature extractor, a local feature extractor, a spatial feature extractor and the feature fusion. We provide a detailed ablation study on this module showing how each of the proposed components contributes to improve the final performance of 3D-MiniNet. Besides, we implemented a fast

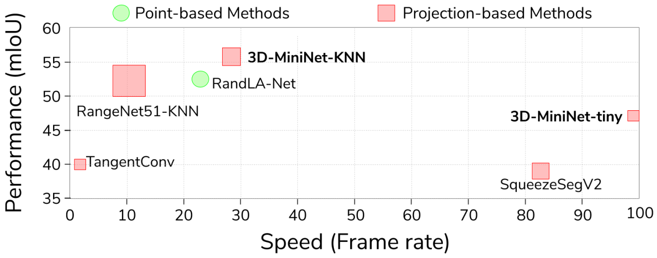

Fig. 1. **3D LIDAR semantic segmentation accuracy vs speed** on the SemanticKITTI [14] benchmark (test set). Point-based methods are drawn as green circles and projection-based methods as red squares. Areas of squares and circles depict the number of parameters used in each method. Our proposed 3D-MiniNet outperforms previous methods while being more parameter efficient and faster. Best viewed in color.

[1] RoPeRt group, at DIIS - I3A, Universidad de Zaragoza, Spain. {inigo, riazuelo, montesano, acm}@unizar.es [2] Bitbrain, Zaragoza, Spain

version of the point neighbor search based on a sliding-window on the spherical projection [19] in order to compute it at an acceptable frame-rate.

All the code and trained models will be made available to the community upon acceptance.

II. Related Work

A. 2D Semantic Segmentation

Current 2D semantic segmentation state-of-the-art methods are deep learning solutions [20]–[23]. Semantic segmentation architectures are evolved from convolutional neural networks (CNNs) architectures for classification tasks, adding a decoder on top of the CNN. Fully Convolutional Neural Networks for Semantic Segmentation (FCNN) [23] carved the path for modern semantic segmentation architectures. The authors of this work propose to upsample the learned features of classification CNNs using bilinear interpolation up to the input resolution and compute the cross-entropy loss per pixel. Another of the early approaches, SegNet [24], proposes a symmetric encoder-decoder structure using the unpooling operation as upsampling layer. More recent works improve these earlier segmentation architectures by adding novel operations or modules proposed initially within CNNs architectures for classification tasks. FC-DenseNet [22] follows DenseNet work [25] using dense modules. PSPNet [26] uses ResNet [27] as its encoder and introduces the Pyramid Pooling Module incorporated at the end of the CNN allowing to learn effective global contextual priors. Deeplab-v3+ [20] is one of the top-performing architectures for segmentation. Its encoder is based on Xception [28], which makes use of depthwise separable convolutions [29] and atrous (dilated) convolutions [30].

With respect to efficiency, ENet [31] set up certain basis which following works, such as ERFNet [32], ICNet [33], have built upon. The main idea is to work at low resolutions, i.e., quick downsampling, and to focus the computation on the encoder having a very light decoder. MiniNetV2 [34] uses a multi-dilation depthwise separable convolution, which efficiently learns both local and global spatial relationships. In this work, we take MiniNetV2 as our backbone and adapt it to capture information from raw LIDAR points.

B. 3D Semantic Segmentation

There are three main groups of strategies to approach this problem: point-based methods, 3D representations and projection-based methods.

1) Point-based Methods: Point-based methods work directly on raw point clouds. The order-less structure of the point clouds prevents standard CNNs to work on this data. The pioneer approach and base of the following point-based works is PointNet [6]. PointNet proposes to learn per-point features through shared MLP (multi-layer perceptron) followed by symmetrical pooling functions to be able to work on unordered data. Lots of works have been later proposed based on PointNet. Following with the point-wise MLP idea, PoinNet++ [5] groups points in an hierarchical manner and learns from larger local regions. The authors also propose a multi-scale grouping for coping with the non-uniformity nature of the data. In contrast, other approaches propose different types of operations following the convolution idea. Hua et al. [35] propose to bin neighboring points into kernel cells for being able to perform point-wise convolutions. Other works resort to graph networks to capture the underlying geometric structure of the point cloud. Loic et al. [36] use a directed graph to capture the structure and context information. For this, the authors represent the point cloud as a set of interconnected superpoints.

2) 3D representations: There are different kinds of representations of the raw point cloud data which have been used for 3D semantic segmentation. SegCloud [37] makes use of a *volumetric or voxel representation*, which is a very common way for encoding and discretizing the 3D space. This approach feeds the 3D voxels into a 3D-FCNN [23]. Then, the authors introduce a deterministic trilinear interpolation to map the coarse voxel predictions back to the original point cloud and apply a CRF as a final step. The main drawback of this voxel representation is that 3D-FCNN has very slow execution times for real-time applications. Su et al. [38] proposed SPLATNet, making use of another type of representation: *Permutohedral Lattice representation*. This approach interpolates the 3D point cloud to a permutohedral sparse lattice and then bilateral convolutional layers are applied to convolve on occupied parts of the representation. LatticeNet [39] was later proposed improving SPLATNet proposing its DeformSlice module for re-projecting the lattice feature back to the point cloud.

3) Projection-based Methods: This type of approaches rely on projections of the 3D data into a 2D space. For example, TangentConv [7] proposes to project the neighboring points into a common tangent plane where they perform convolutions. Another type of projection-based method is the *spherical representation*. This strategy consists of projecting the 3D points into a spherical projection and has been widely used for LIDAR semantic segmentation. This representation is a 2D projection that allows the application of 2D images operations, which are very fast and work very well on recognition tasks. SqueezeSeg [9] and its posterior improvement SqueezeSegV2 [8], based on SqueezeNet architecture [40], show that very efficient semantic segmentation can be done through this projection. The more recent work from Milioto et al. [3] combines the DarkNet [41] architecture with a GPU based post-processing method which obtains better results than Conditional random fields (CRF) for real-time semantic segmentation.

Projection-based approaches tend to be faster than the previous representations, but they lose the pontential of learning 3D features. LuNet [42] is a recent work which proposes to learn local features using point-based operations before projecting into the 2D space. Our novel projection module tackles with this issue by including a context feature extractor based on point-based operations. Besides, we build a faster and more parameter-efficient architecture and a faster implementation of LuNet's neighbor search method.

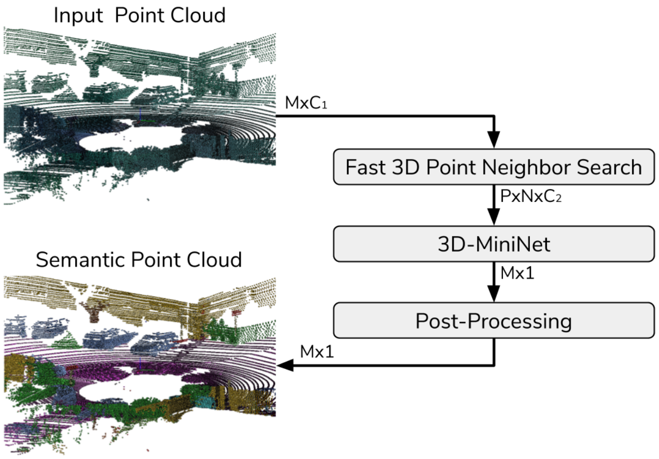

Fig. 2. Proposed approach overview. The M points from the input point cloud (with C_1 features) are split into P groups of N points with our fast 3D point neighbor search. Each point has a C_1 feature vector, which is extended to C_2 in this process with data relative to each group. The proposed 3D-MiniNet takes the point groups and predicts one semantic label per point. A post-processing method [3] is used to refine the final results.

III. 3D-MININET: LIDAR POINT CLOUD SEGMENTATION

Our novel approach for LIDAR semantic segmentation is summarized in Fig. 2. It consists of three modules: (A) fast 3D point neighbor search, (B) 3D-MiniNet, which takes P groups of N points and outputs the segmented point cloud and, (C) the KNN-based post-processing which refines the final segmentation.

There are two main issues that typically prevent point-based models to run at an acceptable frame-rate compared to projection-based methods: 3D point neighbor search is a required, but slow, operation and performing 3D operations is slower than using 2D convolutions. In order to alleviate these two issues, our approach includes a fast point neighbor search proxy (subsection III-A), and a module to minimize expensive point-based operations, which takes raw 3D points and outputs a 2D representation to be processed with a 2D CNN (subsection III-B.1).

A. Fast 3D Point Neighbor Search

We need to find the 3D neighbors because we want to learn features that encode the relationship of each point with their neighbors in order to learn information about the shape of the point-cloud. In order to perform the 3D neighbor search more efficiently, we first project the point cloud into a spherical projection of shape $W \times H$, mapping every 3D point (x, y, z) into a 2D coordinate (u, v), i.e., $\mathbb{R}^3 \rightarrow \mathbb{R}^2$:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\left[1-\arctan(y,x)\pi^{-1}\right] W \\ \left[1-\left(\arcsin\left(zr^{-1}\right)+\mathrm{f}_{\mathrm{up}}\right)\mathrm{f}^{-1}\right] H \end{pmatrix}, \tag{1}$$

where $f = f_{\mathrm{up}} + f_{\mathrm{down}}$ is the vertical field-of-view of the sensor and r is the depth of each point. We perform the projection of Eq. 1 following [3], where each pixel encodes one 3D point with five features: $C_1 = \{x, y, z, depth, remission\}$.

We perform the point neighbor search in the spherical projection space using a sliding-window approach. Similarly to a convolutional layer, we get groups of pixels, i.e., projected points, by sliding a $k \times k$ window across the image. The generated groups of points have no intersection, i.e., each point belongs only to one group. This step generates P point groups of N points each ($N = k^2$), where all points from the spherical projection are used ($P \times N = W \times H$).

Before feeding the actual segmentation module, 3D-MiniNet, with these point groups, the features of each point are augmented. For each group we compute the relative (r) feature values of each point with respect to the group mean of each feature in C_1 (similar to previous works which compute them relative to a center point [12], [19]). Besides, similar to [43], we compute the 3D euclidean distance of each point to the mean point. Therefore, each point has now eleven features: $C_2 = \{x, x_{\mathrm{r}}, y, y_{\mathrm{r}}, z, z_{\mathrm{r}}, depth, depth_{\mathrm{r}}, remission, remission_{\mathrm{r}}, d_{Euc}\}$.

mean values of the five features in C_1 and the relative (r) values for each point with respect to the group mean (similar to previous works which compute them relative to a center point [12], [19]).

B. 3D-MiniNet

3D-MiniNet consists of two modules, as represented in Fig. 3: the proposed projection module, which takes the raw point cloud and computes a 2D representation, and our efficient backbone network based on MiniNetV2 [34] to compute the semantic segmentation.

1) Projection Learning Module: The goal of this module is to transform raw 3D points to a 2D representation that can be used for efficient segmentation. The input of this module if the output of the point neighbor search described in the previous subsection. It is a set of P groups, where each group contains N points with C_2 features each, gathered through the sliding-window search on the spherical projection as explained in the previous subsection.

The following three kinds of features are extracted from the input data (see left part of Fig. 3 for a visual description of this proposed module) and fused in the final module step:

Local Feature Extractor: The first feature is a PointNet-like local feature extraction (see projection learning module (a) of Fig. 3). It runs four linear layers shared across the groups followed by a BatchNorm [44] and LeakyRelu [45].We follow PointPillars [15] implementation of these shared linear layers using 1x1 convolutions across the tensor resulting in very efficient computation when handling lots of point groups.

Context Feature Extractor: The second feature extraction (projection learning module (b) of Fig. 3) learns context information from the points.This is a very important module because although context information can be learned through the posterior CNN, point-based operations learn different features than convolutions. Therefore, this module helps learning a richer representation with information than might not be learned through the CNN.

The input of this context feature extractor is the output of the second linear layer of the local feature extractor

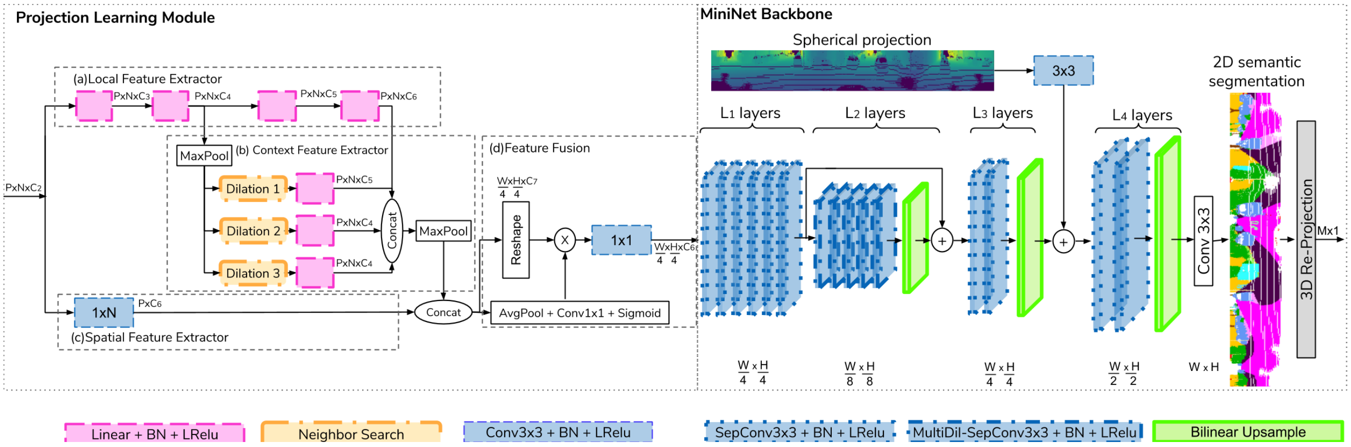

Fig. 3. 3D-MiniNet overview. It takes P groups of N points each and computes semantic segmentation of the M points of the point cloud where $P \times N = M$. It consists of two main modules: our proposed learning module (on the left) which learns a 2D tensor which is fed to the second module, an efficient FCNN backbone (on the right) which computes the 2D semantic segmentation. Each 3D point of the point cloud is given a semantic label based on the 2D segmentation. Best viewed in color.

(giving the last linear layer as input would drop significantly the frame-rate due to the high number of features). This tensor is maxpooled (in order to complete the PointNet-like operation which work on unordered points) and then, our fast neighbor search is run to get point groups. In this case, three different groupings (using our point neighbor search) are performed with a 3×3 sliding window with different dilation rates of 1, 2, 3 respectively. Dilation rates, as in convolutional kernels [30], keep the number of grouped points low while increasing the receptive field allowing a faster context learning. We use zero-padding and a stride of 1 for keeping the same size. After every grouping we perform a linear, BatchNorm and LeakyRelu. The outputs of these two feature extractor modules are concatenated and applied a maxpool operation over the N dimension. This maxpool operation keeps the feature with higher response along the neighbor dimension, being order-invariant with respect to the neighbor dimension. The maxpool operation also makes the learning robust to pixels with no point information (spherical projection coordinates with no point projected).

Spatial Feature Extractor: The last feature extraction operation is a convolutional layer of kernel $1 \times N$ (projection learning module (c) of Fig. 3). Convolutions can extract features of each point with respect to the neighbors when there is an underlying spatial structure which is the case, as the point groups are extracted from a 2D spherical projection. In the experiment section, we take this feature extractor as our baseline without the two others which is equivalent of performing only standard convolutions on the spherical projection.

Feature Fusion: Lastly, a feature fusion with self-attention module is applied. It learns to reduce the feature space into an specified number of features, learning which features are more important. It consists of three stages: (1) concatenation of the feature extraction outputs reshaping the resulting tensor to $(W/4 \times H/4 \times C_7)$, (2) a self-attention operation which multiplies the reshaped tensor by the output of a pooling, 1×1 convolution and sigmoid function which has the same concatenated tensor as its input and, (3) a 1×1 convolutional layer followed by a BatchNorm and LeakyRelu which acts as a bottleneck limiting the output to C_6 features.

All implementation details, such as the number of features of each layer, are specified in Sect. IV. The experiments in Sect. V show how each part of this learning module contributes to improve 3D-MiniNet's performance.

2) 2D Segmentation Module (MiniNet Backbone): Once the previous module has computed a $W/4 \times H/4 \times C_6$ tensor, the 2D semantic segmentation is computed with an efficient CNN (see MiniNet backbone of Fig. 3 for the detailed visual description). We mainly use a FCNN instead of performing more MLP operations because convolutional layers have lower inference time when working on high dimensional spaces.

We base our FCNN on MiniNetV2 architecture [34]. Our encoder performs L_1 depthwise separable convolutions and L_2 multi-dilation depthwise separable convolutions. For the decoder, we use bilinear interpolations as upsampling layers. It performs L_3 depthwise separable convolutions at $W/4 \times H/4$ resolution and L_4 at $W/2 \times H/2$ resolution. Finally, a convolution is performed at $W \times H$ resolution to get the 2D semantic segmentation prediction.

We also follow MiniNetV2 approach in having a second convolutional branch for extracting fine-grained information, i.e., high-resolution low-level features. The input of this second branch is the spherical projection.

The number of layers and features at each layer is specified in Sect. IV-B.

As a final step, the predicted 2D semantic segmentation has to be re-projected back again into the 3D space ($\mathbb{R}^2 \rightarrow \mathbb{R}^3$). For the points projected into the spherical representation is a straightforward step inasmuch we only have to assign the semantic label predicted in the spherical projection. Nevertheless, the points that had not been projected into the spherical projection (one 2D coordinate can have more than one 3D point), have no semantic label. For those points, the semantic label of its corresponding 2D coordinate is assigned. As this issue may lead to miss-predictions, a post-processing method is performed to refine the results.

C. Post-Processing

In order to cope with the miss-predictions of non-projected 3D points, we follow Milioto et al. [3] post-processing method. All 3D points get a new semantic label based on K Nearest Neighbors (KNN). The criteria for selecting the K nearest points is not based on the relative euclidean distances but on relative depth values. Besides, the search is narrowed down based on 2D spherical coordinate distances. Milioto et al. implementation is GPU-based and is able to run in 7ms keeping the frame-rate high.

IV. Experimental setup

This section details the experimental setup used to evaluate our method.

A. Datasets

SemanticKITTI Benchmark: The SemanticKITTI dataset [14] is a recent large-scale dataset that provides dense point-wise annotations for the entire KITTI Odometry Benchmark [18]. The dataset consists of over 43000 scans from which over 21000 are available for training (sequences 00 to 10) and the rest (sequences 11 to 21) are used as test set. The dataset distinguishes 22 different semantic classes from which 19 classes are evaluated on the test set via the official online platform of the benchmark. As this is the current most relevant and largest dataset of single-scan 3D LIDAR semantic segmentation, we perform our ablation study and our more thorough evaluation on this dataset.

KITTI Benchmark: SqueezeSeg [9] work provided semantic segmentation labels exported from the 3D object detection challenge of the KITTI dataset [18]. It is a medium-size dataset split into 8057 training and 2791 validation scans.

B. Settings

a) 3D Point Neighbor Search Parameters: We set the resolution of the spherical projection to 2048×64 for the SemanticKITTI dataset and 512×64 for the KITTI (same resolution than previous works to be able to make fair comparisons). We set a 4×4 window size with a stride of 4 and no zero-padding for our fast point neighbor search leading to 8192 groups of 3D points for the SemanticKITTI data and 2048 groups for the KITTI data. Our projection module is fed with these groups and generates a learned representation of resolution 512×16 for the SemanticKITTI configuration and 128×16 for the KITTI.

b) Network Parameters: The full architecture and all its parameters are described in Fig. 3. We considered three different configurations for evaluating the proposed approach: 3D-MiniNet, 3D-MiniNet-small, 3D-MiniNet-tiny. The number of features (C_3, C_4, C_5, C_6) for the projection module of the different 3D-MiniNet configurations are (24, 48, 96, 192) features for 3D-MiniNet, (16, 32, 64, 128) for 3D-MiniNet-small and (12, 24, 48, 96) for 3D-MiniNet-tiny. The number of layers (L_1, L_2, L_3, L_4) of the FCNN backbone network are (50, 30, 4, 2) features for 3D-MiniNet, (24, 20, 2, 1) for 3D-MiniNet-saml and (14, 10, 2, 1) for 3D-MiniNet-tiny. N_{c} is the number of semantic classes of the dataset.

TABLE I

Ablation study of the different parts of the projection module evaluated on the test set of SemantiKITTI.

Method	Data Aug.	Conv	Local MLP	Attention	Context MLP	Relative features	mIoU	FPS	Params (M)
3D-MiniNet Small		✓					44.4	73	0.93
	✓	✓					47.6	73	0.93
	✓		✓				48.7	69	0.93
	✓	✓	✓				49.5	66	0.96
	✓	✓	✓	✓			49.9	65	1.08
	✓	✓	✓	✓	✓		51.2	61	1.13
	✓	✓	✓	✓	✓	✓	51.8	61	1.13

c) Post-processing Parameters: For the K Nearest Neigbors post-process method [3], we set as 7×7 the windows size of the neighbor search on the 2D segmentation and we set K to 7.

d) Training protocol: We train the different 3D-MiniNet configurations for 500 epochs with batch size of 3, 6 and 8 for 3D-MiniNet, 3D-MiniNet-small, and 3D-MiniNet-tiny respectively (different due to memory constraints). We use Stochastic Gradient Descent (SGD) optimizer with an initial learning rate of $4 \cdot 10^{-3}$ and a decay of 0.99 every epoch. For the optimization, we use the cross-entropy loss function, see eq. 2.

$$\mathcal{L} = -\frac{1}{M} \sum_{m=1}^{M} \sum_{c=1}^{C} (\frac{f_t}{f_c})^i y_{c,m} \ln(\hat{y}_{c,m}), \tag{2}$$

where M is the number of labeled points and C is the number of classes. $Y_{c,m}$ is a binary indicator (0 or 1) of point m belonging to a certain class c and $\hat{y}_{c,m}$ is the CNN predicted probability of point m belonging to a certain class c. This probability is calculated by applying the soft-max function to the networks' output. To account for class imbalance, we use the median frequency class balancing, as applied in SegNet [24]. To smooth the resulting class weights, we propose to apply a power operation, $w_c = (\frac{f_t}{f_c})^i$, with f_c being the frequency of class c and f_t the median of all frequencies. We set i to 0.25.

e) Data augmentation: During the training, we randomly rotate and shift the whole 3D point cloud. We randomly invert the sign for X and Z values for all the point cloud. We also drop some points. The rotation angle is a Gaussian distribution with mean 0 and standard deviation (std) of 40. The shifts we perform are Gaussian distributions with mean 0 and std of 0.35, 0.35 and 0.01 (meters) for the X, Y, Z axis (being Z the height). The percentage of dropped points is a uniform distribution between 0 and 10.

V. Results

A. Ablation Study of the Projection Module

The projection module is the main novelty from our approach. This subsection shows how each part helps to

TABLE II

RESULTS ON SINGLE-SCAN TEST SET IN SEMANTICKITTI [14]. POINT-BASED METHODS: ROWS 1-4. 3D REPRESENTATIONS: ROW 5. PROJECTION-BASED METHODS: ROWS 6-11.

Methods	Size	**mIoU**	**Frame-rate (FPS)**	**Params(M)**	road IoU	sidewalk IoU	parking IoU	other-ground IoU	building IoU	car IoU	truck IoU	bicycle IoU	motorcycle IoU	other-vehicle IoU	vegetation IoU	trunk IoU	terrain IoU	person IoU	bicyclist IoU	motorcyclist IoU	fence IoU	pole IoU	traffic-sign IoU
PointNet [6]	50K pts	14.6	2	3	61.6	35.7	15.8	1.4	41.4	46.3	0.1	1.3	0.3	0.8	31.0	4.6	17.6	0.2	0.2	0.0	12.9	2.4	3.7
SPG [36]		17.4	0.2	**0.25**	45.0	28.5	0.6	0.6	64.3	49.3	0.1	0.2	0.2	0.8	48.9	27.2	24.6	0.3	2.7	0.1	20.8	15.9	0.8
PointNet++ [5]		20.1	0.1	6	72.0	41.8	18.7	5.6	62.3	53.7	0.9	1.9	0.2	0.2	46.5	13.8	30.0	0.9	1.0	0.0	16.9	6.0	8.9
RandLA-Net [46]		53.9	22	1.24	90.7	73.7	60.3	20.4	86.9	**94.2**	**40.1**	26.0	25.8	**38.9**	81.4	**61.3**	**66.8**	**49.2**	**48.2**	7.2	56.3	**49.2**	47.7
SPLATNet [38]	50K pts	18.4	1	0.8	64.6	39.1	0.4	0.0	58.3	58.2	0.0	0.0	0.0	0.0	71.1	9.9	19.3	0.0	0.0	0.0	23.1	5.6	0.0
SqueezeSeg [9]	64x2048 px	29.5	90	1	85.4	54.3	26.9	4.5	57.4	68.8	3.3	16.0	4.1	3.6	60.0	24.3	53.7	12.9	13.1	0.9	29.0	17.5	24.5
DBLiDARNet [11]		37.6	—	2.8	85.8	59.3	8.7	1.0	78.6	81.5	6.6	29.4	19.6	6.5	77.1	46.0	58.1	23.7	20.1	2.4	39.6	32.6	39.1
SqueezeSegV2 [8]		39.7	83	1	88.6	67.6	45.8	17.7	73.7	81.8	13.4	18.5	17.9	14.0	71.8	35.8	60.2	20.1	25.1	3.9	41.1	20.2	36.3
TangentConv [7]		40.9	0.3	0.4	83.9	63.9	33.4	15.4	83.4	90.8	15.2	2.7	16.5	12.1	79.5	49.3	58.1	23.0	28.4	8.1	49.0	35.8	28.5
RangeNet21 [3]		47.4	25	25	91.4	74.0	57.0	26.4	81.9	85.4	18.6	26.2	26.5	15.6	77.6	48.4	63.6	31.8	33.6	4.0	52.3	36.0	50.0
RangeNet53 [3]		49.9	13	50	91.7	74.0	65.1	28.2	82.9	85.3	25.8	22.7	33.6	22.2	77.3	50.0	64.6	36.8	31.4	4.7	54.8	39.1	52.3
RangeNet53-KNN [3]		52.2	12	50	**91.8**	**75.2**	**65.0**	**27.8**	87.4	91.4	25.7	25.7	34.4	23.0	80.5	55.1	64.6	38.3	38.8	4.8	58.6	47.9	55.9
3D-MiniNet-tiny (Ours)	64x2048 px	46.9	**98**	0.44	90.7	70.7	59.4	20.0	83.4	82.0	19.0	29.3	25.4	20.8	77.9	50.6	60.8	35.1	32.3	3.2	51.0	32.7	46.7
3D-MiniNet-small (Ours)		51.8	61	1.13	91.5	72.3	61.7	25.1	83.9	83.4	25.4	35.6	25.4	25.1	80.3	53.9	64.3	43.4	42.3	20.7	53.0	36.4	50.3
3D-MiniNet (Ours)		53.0	36	3.97	91.6	74.0	64.1	25.9	85.8	85.2	28.3	37.9	39.3	28.8	80.3	54.5	65.9	43.8	40.3	14.4	57.0	37.9	51.5
3D-MiniNet-tiny-KNN (Ours)		49.0	55	0.44	90.7	71.0	59.5	19.7	86.4	86.6	19.2	31.6	27.8	21.3	80.0	55.4	61.4	38.1	35.0	3.0	53.7	40.5	51.0
3D-MiniNet-small-KNN (Ours)		54.4	40	1.13	91.5	72.7	61.8	24.6	87.1	88.1	25.6	39.3	38.0	25.6	82.5	59.7	65.0	47.2	46.2	**22.4**	56.1	45.8	54.9
3D-MiniNet-KNN (Ours)		**55.8**	28	3.97	91.6	74.5	64.2	25.4	**89.4**	90.5	28.5	**42.3**	**42.1**	29.4	**82.8**	60.8	66.7	47.8	44.1	14.5	**60.8**	48.0	**56.6**

Scans per second have been measured using a Nvidia gtx 2080ti
— Not reported by the authors.

improve the learned representation. For this experiment, we use 3D-MiniNet-small configuration.

Table I shows the ablation study of our proposed module, measuring the mIoU, speed and learning parameters needed with each configuration. The first row and baseline is working on the spherical projection using a convolution as the *projection* method, i.e., just a downsampling in that case.

As the projection used is neither rotation nor shift invariant, performing this data augmentation helps to our network generalization as first row shows. Second row shows the performance using only $1 \times N$ convolutions in the learning layers with the 5-channel input (C_1) used in RangeNet [3] which we establish as our baseline, i.e, our spatial feature extractor. The third row shows the performance if we replace the $1 \times N$ convolution for point-based operations, i.e, our local feature extractor. These results point that MLP operations work better for 3D points but take more execution time. The fourth row combines both the convolution and local MLP operation. Combining convolutions and MLP operations increases performance due to the different type of features learned by each type of operation as explained in Sect. III-B.1.

The attention module also increases the performance with almost no extra computational effort. It reduces the feature space into a specified number of features, learning which features are more important. The sixth row shows the results adding our context feature extractor. Context is also learned later through the FCNN via convolutions but here, the context feature extractor learns different context through with MLP operations. Context information is often very useful in semantic tasks, e.g., for distinguishing between a bicyclist, a cyclist and a motorcyclist. This context information gives a boost higher than the other feature extractors showing its relevance. Finally, increasing the number of features of each point with features relative to the point group (C_2) also leads to better performance without decreasing the frame-rate and without adding any learning parameter.

TABLE III

RESULTS ON KITTI [18] VALIDATION SET.

Methods	Size	**mIoU**	**Frame-rate (fps)**	**Params(M)**	car IoU	pedestrian IoU	cyclist IoU
SqueezeSeg [9]	64x512 px	37.2	227	1	64.6	21.8	25.1
PointSeg [10]		39.7	160	—	67.4	19.2	32.7
SqueezeSegv2 [8]		44.9	143	1	73.2	27.8	33.6
LuNet [19]		55.4	67*	23.4	72.7	46.9	46.5
DBLiDARNet [11]		56.0	—	2.8	75.1	47.4	45.4
3D-MiniNet-tiny (Ours)	64x512 px	45.5	**245**	**0.44**	69.6	37.5	29.5
3D-MiniNet-small (Ours)		50.6	161	1.13	74.4	40.7	36.7
3D-MiniNet (Ours)		**58.0**	92	3.97	**75.5**	**49.6**	**48.9**

Scans per second have been measured using a Nvidia gtx 2080ti
* Offline neighboring point search is not taken into account.
— Not reported by the authors.

B. Benchmarks results

This subsection presents quantitative and qualitative results of 3D-MiniNet and comparisons with other relevant works.

a) Quantitative Analysis: Table II compares our method with several point-based approaches (rows 1-4), 3D representation methods (row 5) and projection-based approaches (rows 6-11) measuring the mIoU, the processing speed (FPS) and the number of parameters required by each method. As we can see, point-based methods for semantic segmentation of LIDAR scans tend to be slower than projection ones without providing better performance. As LIDAR sensors such as Velodyne usually work at 5-20 FPS, only RandLA-Net and projection-based approaches are currently able to process in real time the full amount of data made available by the sensor.

Looking at the different configurations of 3D-MiniNet, it gets state-of-the-art using fewer parameters and being faster (3D-MiniNet-small-KNN) beating both RandLANet

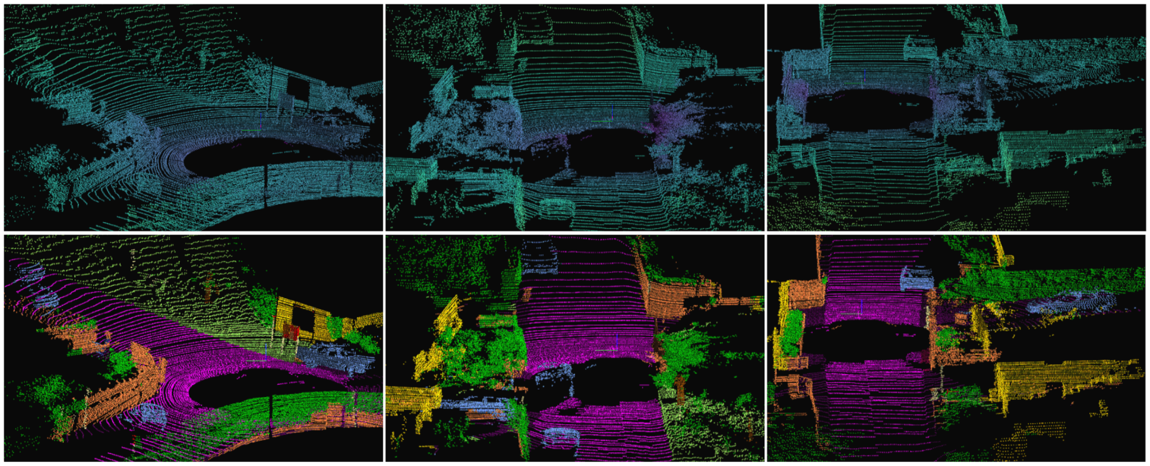

Fig. 4. 3D-MiniNet LIDAR semantic segmentation predictions on the SemanticKITTI benchmark (test sequence 11). LIDAR point cloud are on top where color represents depth. Predictions are on bottom where color represents semantic classes: cars in blue, road in purple, vegetation in green, fence in orange, building in yellow and traffic sign in red. For the full video sequence, go to https://www.youtube.com/watch?v=5ozNkgFQmSM. Best viewed in color.

(point-based method), SPLATNet (3D representation) and RangeNet53-KNN (projection-based). Besides, 3D-MiniNet-KNN configuration is able to get even better performance although it needs more parameters than RandLANet. If efficiency can be traded off for performance, smaller versions of Mininet also obtain better performance metrics at higher frame-rates. 3D-MiniNet-tiny is able to run at 98 fps and, with only a 9% drop in mIoU (46.9% compared to the 29% of SqueezeSeg version that runs at 90 fps).

The post-processing method applied [3] shows its effectiveness improving the results the same way it improved RangeNet. This step is crucial to correctly process points that were not included in the spherical projection, as discussed in more detail in Sect. III.

The scans of the KITTI dataset [18] have a lower resolution (64x512) as we can see in the evaluation reported in Table III. 3D-MiniNet also gets state-of-the-art performance on LIDAR semantic segmentation on this dataset. Our approach gets considerably better performance than SqueezeSeg versions (+10-20 mIoU). 3D-MiniNet also gets better performance than LuNet and DBLiDARNet which were the previous best methods on this dataset.

Note that in this case, we did not evaluate the KNN post-processing since this dataset only provides 2D labels.

The experiments show that projection-based methods are more suitable for the LIDAR semantic segmentation with a good speed-performance trade-off. Besides, better results are obtained when including point-based operations to extract both context and local information from the 3D raw points into the 2D projection.

b) Qualitative Analysis: Fig. 4 shows a few examples of 3D-MiniNet inference on test data. The supplementary video includes inference results on a full sequence[1]. As test ground-truth is not provided for the test set (evaluation is performed externally on the online platform), we can only show visual results with no label comparison.

Note the high quality results on our method in relevant classes such as cars, as well as in challenging classes such as traffic signs. In the supplementary video we can also appreciate some of the 3D-MiniNet failure cases. As it could be expected, the biggest difficulties happen distinguishing between classes with similar geometric shapes and structures like building and fences.

VI. CONCLUSIONS

In this work, we propose 3D-MiniNet, a fast and efficient approach for 3D LIDAR semantic segmentation. 3D-MiniNet projects the 3D point cloud into a 2-Dimensional space and then learns the semantic segmentation using a fully convolutional neural network. Differently from common projection-based approaches that perform a predefined projection, 3D-MiniNet learns this projection from the raw 3D points, learning both local and context information from point-based operations, showing very promising and effective results. Our ablation study shows how each part of the proposed approach contributes to the learning of the representation. We validate our approach on the SemanticKITTI and KITTI public benchmarks. 3D-MiniNet gets state-of-the-art results while being faster and more efficient than previous methods.

REFERENCES

[1] R. Jian, W. Su, R. Li, S. Zhang, J. Wei, B. Li, and R. Huang, “A semantic segmentation based lidar slam system towards dynamic environments,” in *International Conference on Intelligent Robotics and Applications*. Springer, 2019, pp. 582–590.

[2] Z. Zhao, W. Zhang, J. Gu, J. Yang, and K. Huang, “Lidar mapping optimization based on lightweight semantic segmentation,” *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 3, pp. 353–362, 2019.

[3] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, “Rangenet++: Fast and accurate lidar semantic segmentation,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[4] S. W. Chen, G. V. Nardari, E. S. Lee, C. Qu, X. Liu, R. A. F. Romero, and V. Kumar, “Sloam: Semantic lidar odometry and mapping for forest inventory,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 612–619, 2020.

[1] `https://www.youtube.com/watch?v=5ozNkgFQmSM`

[5] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in neural information processing systems*, 2017, pp. 5099–5108.

[6] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 652–660.

[7] M. Tatarchenko, J. Park, V. Koltun, and Q.-Y. Zhou, "Tangent convolutions for dense prediction in 3d," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3887–3896.

[8] B. Wu, X. Zhou, S. Zhao, X. Yue, and K. Keutzer, "Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 4376–4382.

[9] 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 *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1887–1893.

[10] Y. Wang, T. Shi, P. Yun, L. Tai, and M. Liu, "Pointseg: Real-time semantic segmentation based on 3d lidar point cloud," *arXiv preprint arXiv:1807.06288*, 2018.

[11] A. Dewan and W. Burgard, "Deeptemporalseg: Temporally consistent semantic segmentation of 3d lidar scans," *arXiv preprint arXiv:1906.06962*, 2019.

[12] Z. Zhang, B.-S. Hua, and S.-K. Yeung, "Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1607–1616.

[13] X. Li, S. Du, G. Li, and H. Li, "Integrate point-cloud segmentation with 3d lidar scan-matching for mobile robot localization and mapping," *Sensors*, vol. 20, no. 1, p. 237, 2020.

[14] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "SemanticKITTI: A dataset for semantic scene understanding of lidar sequences," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[15] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 697–12 705.

[16] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4490–4499.

[17] Y. Zhou, P. Sun, Y. Zhang, D. Anguelov, J. Gao, T. Ouyang, J. Guo, J. Ngiam, and V. Vasudevan, "End-to-end multi-view fusion for 3d object detection in lidar point clouds," *arXiv preprint arXiv:1910.06528*, 2019.

[18] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3354–3361.

[19] P. Biasutti, V. Lepetit, J.-F. Aujol, M. Brëdif, and A. Bugeau, "LU-Net: An efficient network for 3d lidar point cloud semantic segmentation based on end-to-end-learned 3d features and u-net," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[20] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," *arXiv:1802.02611*, 2018.

[21] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[22] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio, "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation," in *CVPR Workshops*. IEEE, 2017.

[23] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of IEEE Conference on CVPR*, 2015.

[24] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[25] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," in *Proceedings of IEEE CVPR*, 2017.

[26] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Computer Vision and Pattern Recognition*, 2017.

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

[28] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," *arXiv preprint*, 2016.

[29] L. Sifre and S. Mallat, "Rigid-motion scattering for image classification," Ph.D. dissertation, Citeseer, 2014.

[30] F. Y. and V. K., "Multi-scale context aggregation by dilated convolutions," in *International Conference on learning representations*, 2016.

[31] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "Enet: A deep neural network architecture for real-time semantic segmentation," *arXiv preprint arXiv:1606.02147*, 2016.

[32] E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo, "Erfnet: Efficient residual factorized convnet for real-time semantic segmentation," *IEEE Transactions on Intelligent Transportation Systems*, 2018.

[33] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "ICNet for real-time semantic segmentation on high-resolution images," in *European Computer Vision*, 2018.

[34] I. Alonso, L. Riazuelo, and A. C. Murillo, "Mininet: An efficient semantic segmentation convnet for real-time robotic applications," *IEEE Transactions on Robotics (T-RO)*, 2020.

[35] B.-S. Hua, M.-K. Tran, and S.-K. Yeung, "Pointwise convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 984–993.

[36] L. Landrieu and M. Simonovsky, "Large-scale point cloud semantic segmentation with superpoint graphs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4558–4567.

[37] L. Tchapmi, C. Choy, I. Armeni, J. Gwak, and S. Savarese, "Segcloud: Semantic segmentation of 3d point clouds," in *2017 International Conference on 3D Vision (3DV)*. IEEE, 2017, pp. 537–547.

[38] H. Su, V. Jampani, D. Sun, S. Maji, E. Kalogerakis, M.-H. Yang, and J. Kautz, "Splatnet: Sparse lattice networks for point cloud processing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2530–2539.

[39] R. A. Rosu, P. Schütt, J. Quenzel, and S. Behnke, "Latticenet: Fast point cloud segmentation using permutohedral lattices," *arXiv preprint arXiv:1912.05905*, 2019.

[40] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[41] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[42] P. Biasutti, V. Lepetit, J.-F. Aujol, M. Brédif, and A. Bugeau, "Lu-net: An efficient network for 3d lidar point cloud semantic segmentation based on end-to-end-learned 3d features and u-net," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[43] Z. Zhang, B.-S. Hua, D. W. Rosen, and S.-K. Yeung, "Rotation invariant convolutions for 3d point clouds deep learning," in *2019 International Conference on 3D Vision (3DV)*. IEEE, 2019, pp. 204–213.

[44] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[45] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proceedings icml*, vol. 30, no. 1, 2013, p. 3.

[46] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, "Randla-net: Efficient semantic segmentation of large-scale point clouds," 2020.