

# Multi-scale Interaction for Real-time LiDAR Data Segmentation on an Embedded Platform

Shijie Li, Xieyuanli Chen, Yun Liu, Dengxin Dai, Cyrill Stachniss, Juergen Gall, *Member, IEEE*

**Abstract**—Real-time semantic segmentation of LiDAR data is crucial for autonomously driving vehicles and robots, which are usually equipped with an embedded platform and have limited computational resources. Approaches that operate directly on the point cloud use complex spatial aggregation operations, which are very expensive and difficult to deploy on embedded platforms. As an alternative, projection-based methods are more efficient and can run on embedded hardware. However, current projection-based methods either have a low accuracy or require millions of parameters. In this paper, we therefore propose a projection-based method, called Multi-scale Interaction Network (MINet), which is very efficient and accurate. The network uses multiple paths with different scales and balances the computational resources between the scales. Additional dense interactions between the scales avoid redundant computations and make the network highly efficient. The proposed network outperforms point-based, image-based, and projection-based methods in terms of accuracy, number of parameters, and runtime. Moreover, the network processes more than 24 scans, captured by a high-resolution LiDAR sensor with 64 beams, per second on an embedded platform, which is higher than the framerate of the sensor. The network is therefore suitable for robotics applications.

## I. INTRODUCTION

Environment perception and understanding are key to realize self-driving vehicles and robots. For the full-view perception of the environment, autonomously driving vehicles are usually equipped with multi-sensor systems, among which light detection and ranging (LiDAR) sensors play a key role due to their precise distance measurements. The large point clouds that are generated by the LiDAR sensors, however, need to be interpreted in order to understand the environment.

Although convolution neural networks (CNNs) perform well for semantic image segmentation [1], [2], [3], [4], they cannot be applied directly to 3D point clouds. This is because standard convolutions require a regular grid structure, whereas a raw point cloud is an unordered structure. To address this problem, some methods [5], [6], [7] directly process point clouds using some spatial aggregation operations like grouping and gathering. Although these methods work well in indoor scenarios, it is difficult to apply them to large outdoor scenarios since the computational cost of the aggregation operation increases with the number of points. Another issue is that these methods are inefficient on embedded platforms since they use operations

Manuscript received: June 9, 2021; Revised October 6, 2021; Accepted November 18, 2021. This paper was recommended for publication by Editor Markus Vincze upon evaluation of the Associate Editor and Reviewers' comments.

S. Li, X. Chen, C. Stachniss, and J. Gall are with the University of Bonn, Germany. D. Dai is with MPI for Informatics, Germany. Y. Liu is with ETH Zurich, Switzerland. This work was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2070 - 390732324 and GA1927/5-2 (FOR 2535 Anticipating Human Behavior).

that cannot be efficiently mapped on embedded hardware like Jetson AGX using TensorRT. However, runtime efficiency is of vital importance for real-world applications, especially for autonomously driving vehicles and robots.

Wu *et al.* [8], [9] thus proposed to represent point clouds produced by a LiDAR sensor as an ordered projection map, such that CNNs can then be applied. However, projected LiDAR data and RGB images are different modalities and applying directly 2D image-based methods does not yield a high segmentation accuracy. For this reason, some specific CNNs have been designed for LiDAR-based depth images, named as projection-based methods. Recent projection-based methods like [10], however, are very large with more than 50M parameters, making them not suitable for embedded platforms.

In this work, we therefore propose a lightweight projection-based model for semantic segmentation of LiDAR data that runs in real-time on an embedded platform. To this end, we revisit common multi-scale approaches like U-Net [11] that have one path for each scale, *i.e.*, each scale is processed independently and then fused at the end of the network. These networks, however, use the same operations for each path, which makes them either too expensive for embedded platforms or the accuracy is very low depending how complex the used operations are. In order to achieve a good balance between effectiveness and efficiency, we therefore adapt the computational operations for each path. While the top path extracts low-level clues, which can be easily detected with shallow layers operating on high-resolution feature maps, the bottom path extracts high-level semantic information, which requires more complex operations but on low-resolution feature maps. In order to avoid redundant computations across the paths, we furthermore propose a dense top-to-bottom interaction strategy where feature maps from a path are passed to all lower paths. We term the network, which is shown in Fig. 1, Multi-scale Interaction Network (MINet).

In addition, we show that the accuracy can be increased if additional supervision is added. While this is consistent with [2], [12], [13], [14], we demonstrate that it is important to use the right type of supervision for the right part of the network. In fact, we use semantic supervision only for the two top paths but not for the bottom path and edge supervision for the fusion of the multiple paths. The latter is important to obtain accurate segment boundaries after upsampling the paths with lower resolution. Finally, we process the multi-modal data consisting of 3D coordinates, remission, and depth information first independently and then fuse them in the feature space. This is in contrast to previous works for LiDAR data that just concatenate the modalities and therefore ignore that the characteristics of each modality are different.

In summary, our contributions include:

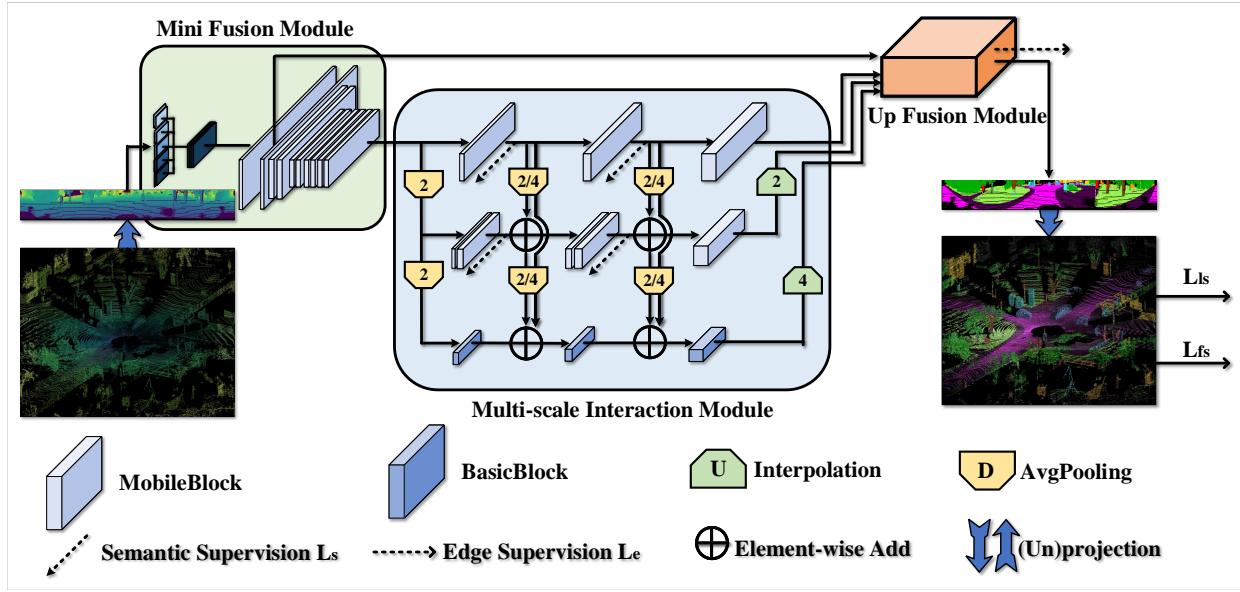


Fig. 1. Illustration of the MINet architecture with three paths in the Multi-scale Interaction Module. The numbers 2 and 4 for interpolation (U) and average pooling (D) indicate the upsampling and downsampling factor. The dashed arrows indicate the supervision type. The detailed description of the architecture is given in Tab. I where the different blocks are illustrated in Fig. 2 and the Up Fusion Module is illustrated in Fig. 3.

- We propose a multi-scale approach where the computational operations are balanced across the different scales and a top-to-bottom interaction strategy avoids redundant computations.
- We exploit different types of additional supervision to improve the accuracy without increasing the inference time.
- Different from previous methods, we process each modality independently and fuse them in the feature space, which improves the overall segmentation performance.
- By incorporating the above design decisions, we propose a lightweight projection-based model for semantic segmentation of LiDAR data that runs in real-time on an embedded platform.

The experimental results demonstrate that our method reduces the number of parameters by about 98% and is about  $4\times$  faster than the state-of-the-art projection-based method [10], while achieving a higher accuracy. We also evaluate our model on an embedded platform and demonstrate that our method can be deployed for autonomous driving.

## II. RELATED WORK

### A. Point-based Semantic Segmentation

Although CNNs [1], [2], [3], [4] are successful for 2D image-based semantic segmentation, they cannot handle unstructured data like point clouds. To address this problem, tangent convolutions [15] project local points to a tangent plane and vanilla convolutions are then applied to it. PointNet [5] is the first method that directly processes the point cloud. It applies a convolution operation for each point and uses a permutation invariant operation to aggregate information. However, PointNet does not take local information into consideration, which is realized by PointNet++ [6]. SPGraph [16] tackles semantic segmentation of large-scale point clouds by defining a super point graph (SPG). Because point-based

methods are inefficient for large point clouds, RandLA [17] addresses this problem by adopting random sampling and designing a better grouping strategy to maintain a better performance. P<sup>2</sup>Net [18] applies point-based methods on projected LiDAR data. However, these methods are too expensive for many applications, especially for embedded platforms.

### B. Projection-based Semantic Segmentation

Projection-based segmentation methods project LiDAR point clouds onto 2D multi-modal images and use 2D CNNs for semantic segmentation. SqueezeSeg [8] and SqueezeSegV2 [9] use a lightweight network called SqueezeNet [19] for semantic segmentation and a CRF for post-processing. Based on SqueezeSeg, RangeNet++ [10] adopts Darknet [20] and replaces the CRF with a  $k$ -NN for post-processing. It has also been successfully used to improve LiDAR-based odometry [21] and loop closure detection [22]. Current projection-based methods, however, do not achieve the same segmentation accuracy as point-based methods and the best performing approaches use very large networks. In this paper, we propose a novel lightweight model that can run in real-time on an embedded platform while achieving state-of-the-art performance.

## III. MULTI-SCALE INTERACTION NETWORK

The proposed Multi-scale Interaction Network (MINet) operates on projection maps generated from LiDAR point clouds. To associate a LiDAR point  $\mathbf{a} = (x, y, z)$  to a pixel  $(u, v)$  in the projection map of size  $(h, w)$ , we compute yaw (1) and pitch (2) and map it to pixel coordinates by translation and scaling [10]:

$$u = \frac{1}{2}[1 - \arctan2(y, x)\pi^{-1}]w, \quad (1)$$

$$v = [1 - (\arcsin(zd^{-1}) + o_{up})o^{-1}]h. \quad (2)$$

TABLE I  
INSTANTIATION OF THE PROPOSED MINET.

Module	Operation	$k$	$c$	$s$	$t$	Output size
MFM	Conv2d	3	4	1	5	$64 \times 2048$
	MobileBlock	3	20	1	1	$64 \times 2048$
	MobileBlock	3	24	2	1	$32 \times 512$
	MobileBlock	3	24	1	1	$32 \times 512$
	MobileBlock	5	40	2	1	$16 \times 256$
	MobileBlock	5	40	1	1	$16 \times 256$
	MobileBlock	5	40	1	1	$16 \times 256$
	MobileBlock	3	80	1	1	$16 \times 256$
	MobileBlock	3	80	1	1	$16 \times 256$
	MobileBlock	3	80	1	1	$16 \times 256$
	Conv2d	1	32	0	1	$16 \times 256$
	MobileBlock	3	64	1	1	$16 \times 256$
	MobileBlock	3	128	1	1	$16 \times 256$
MIM	MobileBlock	3	128	1	1	$16 \times 256$
	MobileBlock	3	32	1	1	$8 \times 128$
	MobileBlock	3	64	1	1	$8 \times 128$
	MobileBlock	3	64	1	1	$8 \times 128$
	MobileBlock	3	128	1	1	$8 \times 128$
	MobileBlock	3	128	1	1	$8 \times 128$
	BasicBlock	3	64	1	1	$4 \times 64$
	BasicBlock	3	128	1	1	$4 \times 64$
	BasicBlock	3	128	1	1	$4 \times 64$
	Conv2d	3	32	1	1	$16 \times 512$
UFM	Conv2d	3	32	1	1	$64 \times 2048$
	Conv2d	1	32	1	1	$64 \times 2048$

\* Each module contains several components: Conv2d, MobileBlock, and BasicBlock. Conv2d denotes a convolutional layer followed by one batch normalization layer and ReLU activation. MobileBlock and BasicBlock are illustrated in Fig. 2. Each operation has a kernel size  $k$ , stride  $s$ , and  $c$  output channels, repeated  $t$  times. The three sections of MIM denote the three paths.

The vertical field-of-view of the LiDAR sensor is  $o = o_{up} + o_{down}$ , where  $o_{up}$  and  $o_{down}$  represent the above and below horizon of the field-of-view, respectively.  $d = \|\mathbf{a}\|$  denotes the depth of a point. After this transformation, we obtain a projection map of size  $(h, w, 5)$  where the 5 channels correspond to the coordinates  $(x, y, z)$ , the depth, and the remission of the corresponding 3D point. The remission value indicates the proportion of the light that is diffusely reflected. It provides therefore information of the surface, which is helpful for distinguishing different classes. While depth can be computed from the coordinates, the network operations do not compute the depth explicitly. Adding depth in addition to the coordinates thus improves the accuracy as we show in our experiments. Each channel is normalized by the mean and standard deviation computed over the training and validation set.

The architecture of MINet is shown in Fig. 1. The projection map is first processed by the Mini Fusion Module (MFM) (Sec. III-A) to fuse the multi-modal information in the feature space. In the Multi-scale Interaction Module (MIM) (Sec. III-B), the data is processed at three different scales where the resolution is reduced by factor two for each path. As it is shown in Tab. I, the computation differs for each path where we use two basic components, namely MobileBlock and BasicBlock. The MobileBlock [23], [24] utilizes depthwise convolutions and has thus fewer parameters, but its learning capacity is also limited. The BasicBlock [25] is stronger, but also more expensive. MobileBlock and BasicBlock are shown in Fig. 2. We therefore balance the computational resources across the three paths as it is shown in Tab. I. While

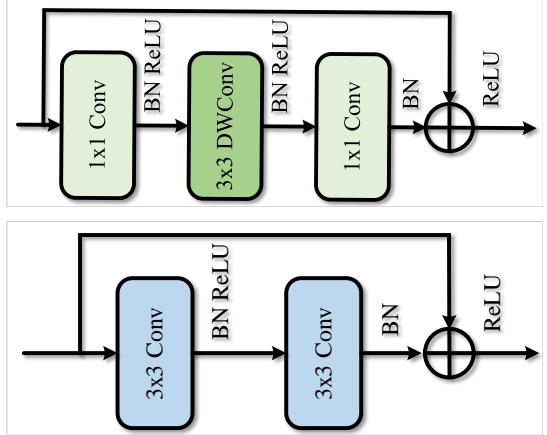


Fig. 2. Illustration of the MobileBlock (top) and the BasicBlock (bottom). DWConv means depth-wise convolution.

we use the expensive BasicBlock for the bottom path with lowest resolution, we decrease the computational cost as the resolution increases using five MobileBlocks for the middle path and three for the top path. The connections from each path to lower paths avoid redundant computations at lower paths and make the network more efficient. Finally, the 2D predictions for the original resolution are produced by the Up Fusion Module (UFM) (Sec. III-C), which are then mapped back to the 3D space. In the remainder of this section, we describe each module of MINet.

#### A. Mini Fusion Module (MFM)

Different from an RGB image, the projection map contains channels of different modalities. Previous projection-based segmentation methods [8], [9], [10] treat such different modalities equally, but we show that processing each channel independently using MFM is more efficient. Specifically, each channel of the multi-modal image is mapped to an independent feature space using five convolution blocks, including normalization and activation. This corresponds to the first row of Tab. I. This step can be considered as a feature calibration step for each modality before fusing them. It needs to be noted that we also treat the  $x$ ,  $y$ , and  $z$  coordinates separately. After the first five convolutional layers, these features are concatenated and fed into several MobileBlocks for fusing them. Since a small resolution leads to less computation, the information of the feature maps are gradually aggregated by average pooling.

#### B. Multi-scale Interaction Module (MIM)

After the fusion module, the data is processed by three paths where each path corresponds to a different scale as shown in Fig. 1. From top to bottom, the resolution of the feature maps is decreased by factor two using average pooling and the receptive field is accordingly increased. For the top path, we use the highest resolution. Since processing high resolution feature maps is very expensive, we use only three MobileBlocks as shown in Tab. I. The bottom path, which has the largest receptive field and lowest resolution, can offer more abstract semantic clues if we use more expensive operations. Hence, it uses three BasicBlocks. The middle path is a compromise between the top and bottom path and consists of five

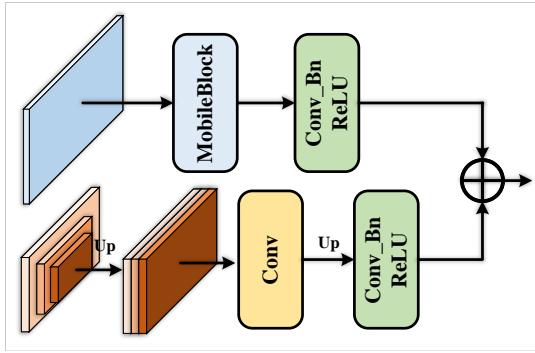


Fig. 3. Illustration of the Up Fusion Module (UFM).

TABLE II  
IMPACT OF THE THREE MODULES.

No.	MFM	Interaction	UFM	mIoU
1		✓	✓	50.9
2	✓		✓	50.7
3	✓	✓		50.6
4	✓	✓	✓	51.8
5	w/o depth	✓	✓	49.6

MobileBlocks. In our experiments, we show that increasing the computational operations as the resolution decreases leads to a higher efficiency compared to using the same blocks for all paths. While the number of parameters doubles compared to the proposed architecture if we use the BasicBlocks for all paths, the accuracy drops if only MobileBlocks are used.

A second important design choice is to allow interactions among the paths. Since the computational complexity of the paths increases for lower paths, we use a dense top-to-bottom fusion design for efficient multi-scale feature interaction. Especially, feature maps of the first and second path will be resized by average pooling and passed to all lower paths. To avoid a mismatch of the number of channels, the number of channels is increased gradually for each path and kept the same at each interaction position. Hence, no other operations are used to adjust the number of channels as shown in Tab. I. Due to the interaction, the lower paths benefit from the features computed from higher paths. The lower paths can therefore focus on information that has not been extracted by higher paths due to limited computational resources.

### C. Up Fusion Module (UFM)

To obtain the semantic labels of each pixel in the projection map, UFM shown in Fig. 3 combines the features from different scales and upsamples them to the input resolution. In addition, features after the first MobileBlock of the Mini Fusion Module are used to recover detailed spatial information as shown in Fig. 1. The lower part of Fig. 3 shows the feature maps of the three different paths that are first resized to the same size, concatenated together, and fused by a  $1 \times 1$  convolution. The fused feature maps are then upsampled to the original resolution and processed by a convolution block including a  $3 \times 3$  convolution, batch normalization, and ReLU activation. The upper part shows the feature maps from the Mini Fusion Module that have already the original resolution. They are processed by a MobileBlock and a convolution

TABLE III  
IMPACT OF ADDITIONAL SUPERVISION.

No.	MIM			UFM	mIoU
	Top	Middle	Bottom		
1	S	S		E	51.8
2		S		E	50.3
3	S			E	50.2
4	S	S			50.5
5				E	49.0
6					48.4
7	S	S	S	E	50.9
8	S	S		S	50.8
9	S	S		E(FL)	49.4

\* “S” denotes semantic supervision. “E” denotes edge supervision. (“FL”) indicates that the focal loss is used for edge supervision.

TABLE IV  
IMPACT OF  $\lambda$  IN (7).

$\lambda$	0	0.01	0.1	1.0
mIoU	49.0	49.3	51.8	51.4

block. Finally, the processed features from both modules are added together. Although the spatial information of the original feature maps already helps to sharpen segment boundaries, which can be fuzzy due to the upsampling, adding additional supervision for the segment boundaries emphasizes this effect as we will explain in the next section.

### D. Booster Training Strategy

Adding supervision to intermediate parts of a network [26] has been shown to be useful for network optimization [2], [12], [13], [14]. In this work, we also use intermediate supervision, however, we propose two different types of supervision. Similar to balancing the computational resources across scales, it is very important to use the right supervision for the right part of the network.

We use the standard weighted cross-entropy loss as semantic supervision

$$\mathcal{L}_s = -\frac{1}{|I|} \sum_{i \in I} \sum_{n=1}^N w_n p_i^n \log(\hat{p}_i^n), \quad (3)$$

where  $N$  is the number of classes,  $|I|$  is the total number of image pixels,  $p_i^n$  is the ground-truth semantic label for pixel  $i$  and class  $n$  ( $p_i^n \in \{0, 1\}$ ), and  $\hat{p}_i^n$  is the predicted class probability. The weight  $w_n$  for class  $n$  is inversely proportional to its occurrence frequency as in [10].

Besides at the end of the network, we use the weighted cross-entropy loss  $\mathcal{L}_s$  for the top and middle path as indicated by the dashed arrows in Fig. 1. As we will show in the experiments, this intermediate supervision improves the training and boosts the accuracy. Adding this semantic supervision to the bottom path, however, does not help since the resolution of the lower path is too low and downsampling of the ground-truth introduces too many artifacts.

As discussed in Sec. III-C, obtaining accurate segment boundaries after upsampling is an issue. Inspired by [27], [28], we extract the semantic boundaries from the ground-

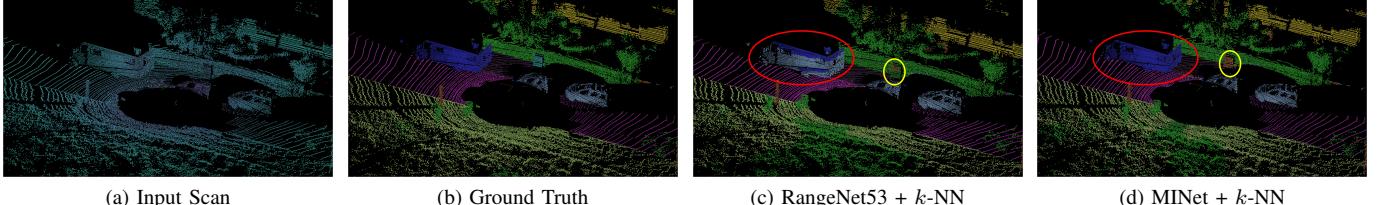


Fig. 4. Qualitative results. RangeNet segments most points of the truck as a car, while MINet segments the truck correctly (red circle). In some cases, both approaches fail (yellow circle). Although MINet segments the object correctly, it misclassifies it.

TABLE V  
ABLATION STUDY FOR DIFFERENT BLOCKS OF EACH PATH.

Top	Middle	Bottom	Param (M)	GFLOP	SPS	mIoU
3×MB	5×MB	3×BB	1.0	6.20	59	51.8
5×MB	5×MB	3×BB	1.1	6.43	52	50.1
7×MB	5×MB	3×BB	1.1	6.62	51	49.7
9×MB	5×MB	3×BB	1.2	7.77	48	48.9
3×MB	3×MB	3×BB	1.0	6.20	60	50.8
3×MB	7×MB	3×BB	1.0	6.29	55	49.0
3×MB	9×MB	3×BB	1.2	6.57	51	50.2
3×MB	5×MB	5×BB	1.4	6.43	54	50.3
3×MB	5×MB	7×BB	1.8	6.62	52	51.5
3×MB	5×MB	9×BB	2.4	6.92	50	49.8
3×MB	3×MB	3×MB	0.6	6.00	61	49.7
5×MB	5×MB	5×MB	0.6	6.30	58	50.2
3×BB	3×BB	3×BB	2.0	11.04	33	51.2

\* “MB” denotes MobileBlock. “BB” denotes BasicBlock. “SPS” represents the number of processed scans per second.

truth labels and compare it to the semantic boundaries after upsampling. The semantic edge loss is then obtained by

$$\mathcal{L}_e = -\frac{1}{|I|} \sum_{i \in I} (e_i \log(\hat{e}_i) + (1 - e_i) \log(1 - \hat{e}_i)), \quad (4)$$

where  $e_i$  is the ground-truth edge label at pixel  $i$  ( $e_i \in \{0, 1\}$ ) and  $\hat{e}_i$  is the predicted edge probability at pixel  $i$ .

Besides of the weighted cross entropy loss, which we denote by  $\mathcal{L}_{fs}$  and which is computed in the same way as  $\mathcal{L}_s$ , we use the Lovász-Softmax loss  $\mathcal{L}_{ls}$  [29] at the end of the network, which maximizes the intersection-over-union (IoU) score:

$$\mathcal{L}_{ls} = \frac{1}{N} \sum_{n=1}^N \overline{\Delta_{J_n}}(m(n)), \quad (5)$$

$$m_i(n) = \begin{cases} 1 - \hat{p}_i^n & \text{if } p_i^n = 1 \\ \hat{p}_i^n & \text{otherwise,} \end{cases} \quad (6)$$

where  $\overline{\Delta_{J_n}}$  defines the Lovász extension of the Jaccard index,  $\hat{p}_i^n \in [0, 1]$  and  $p_i^n \in \{0, 1\}$  denote for class  $n$  at pixel  $i$  the predicted probability and ground-truth label, respectively. In summary, the combined loss is given by

$$\mathcal{L} = \mathcal{L}_{fs} + \mathcal{L}_{ls} + \mathcal{L}_e + \lambda \sum_s \mathcal{L}_s \quad (7)$$

where  $\lambda = 0.1$  and  $\mathcal{L}_s$  are the loss functions for the top and middle path. The arrows in Fig. 1 show where each type of supervision is applied.

#### IV. EXPERIMENTS

##### A. Experimental Settings

We use two challenging datasets to evaluate our method, namely SemanticKITTI [30], [31] and SemanticPOSS [32].

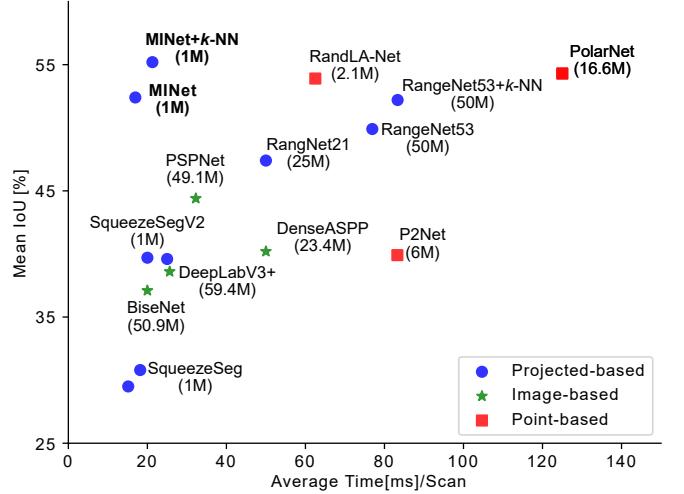


Fig. 5. Visualization of Tab. VI. We omit point-based methods that need more than 140ms per scan. The proposed MINet is not only the most accurate method, it also achieves the best trade-off between accuracy and runtime.

Based on the KITTI Odometry Benchmark [33], SemanticKITTI provides a semantic label for each point in all scans. It includes over 43,000 scans from 21 sequences, among which sequences 00 to 10 with over 21,000 scans are available for training and the remaining scans from sequences 11 to 21 are used for testing. Sequence 08 is used as the validation set, and we train our approach on the remaining training set. We report the results on the validation set for the ablation study. For the test set, which we use to compare to the state-of-the-art, the ground-truth is withheld and the results are evaluated by an on-line server.

SemanticPOSS is a smaller dataset with 2988 LiDAR scans, captured at the Peking University. The point clouds of SemanticPOSS are more sparse compared to SemanticKITTI due to the lower resolution of the LiDAR sensor. SemanticPOSS is split into six subsets equally, among which we use the 3<sup>rd</sup> subset for validation and the others for training. We report the results on the validation set. Since these two datasets differ in size, LiDAR sensor, and environment, they provide an ideal testbed for evaluating the proposed approach. As for the evaluation metric, we calculate the standard mean intersection over union (mIoU) [40] over all classes:

$$\text{mIoU} = \frac{1}{N} \sum_{n=1}^N \frac{\text{TP}_n}{\text{TP}_n + \text{FP}_n + \text{FN}_n} \quad (8)$$

where  $\text{TP}_n$ ,  $\text{FP}_n$ , and  $\text{FN}_n$  denote the numbers of true positive, false positive, and false negative predictions for class  $n$ , respectively.  $N$  is the number of classes.

TABLE VI  
EVALUATION RESULTS ON THE SEMANTICKITTI TEST SET FOR POINT-BASED, IMAGE-BASED, AND PROJECTION-BASED METHODS.

Methods \ Classes																					SPS	Param (M)	mIoU
	car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign				
Pointnet [5]	46.3	1.3	0.3	0.1	0.8	0.2	0.2	0.0	61.6	15.8	35.7	1.4	41.4	12.9	31.0	4.6	17.6	2.4	3.7	2	3.0	14.6	
Pointnet++ [6]	53.7	1.9	0.2	0.9	0.2	0.9	1.0	0.0	72.0	18.7	41.8	5.6	62.3	16.9	46.5	13.8	30.0	6.0	8.9	0.1	6.0	20.1	
SPGraph [16]	68.3	0.9	4.5	0.9	0.8	1.0	6.0	0.0	49.5	1.7	24.2	0.3	68.2	22.5	59.2	27.2	17.0	18.3	10.5	0.2	0.3	20.0	
SPLATNet [34]	66.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	70.4	0.8	41.5	0.0	68.7	27.8	72.3	35.9	35.8	13.8	0.0	1	0.8	22.8	
TangentConv [15]	86.8	1.3	12.7	11.6	10.2	17.1	20.2	0.5	82.9	15.2	61.7	9.0	82.8	44.2	75.5	42.5	55.5	30.2	22.2	0.3	0.4	35.9	
P <sup>2</sup> Net [18]	85.6	20.4	14.4	14.4	11.5	16.9	24.9	5.9	87.8	47.5	67.3	7.3	77.9	43.4	72.5	36.5	60.8	22.8	38.2	12	6.0	39.8	
RandLA-Net [17]	<b>94.2</b>	26.0	25.8	<b>40.1</b>	<b>38.9</b>	49.2	48.2	7.2	90.7	60.3	73.7	20.4	86.9	56.3	81.4	61.3	66.8	49.2	47.7	16	2.1	53.9	
PolarNet [35]	93.8	40.3	30.1	22.9	28.5	43.2	40.2	5.6	90.8	61.7	74.4	21.7	<b>90.0</b>	<b>61.3</b>	<b>84.0</b>	<b>65.5</b>	<b>67.8</b>	<b>51.8</b>	57.5	8	16.6	54.3	
DeepLabV3+ [36]	78.4	13.6	9.5	9.5	10.4	17.5	22.0	0.4	88.5	54.5	66.7	9.7	77.9	39.1	72.0	39.9	60.0	23.4	36.1	39	59.4	38.4	
PSPNet [37]	79.6	25.0	26.4	17.5	24.0	34.1	28.4	7.3	90.2	58.2	70.2	19.9	79.7	43.5	74.2	43.2	61.2	23.1	37.5	31	49.1	44.4	
BiSeNet [38]	76.0	13.4	11.9	18.3	7.6	16.4	26.0	0.5	87.6	49.9	64.2	6.5	74.7	34.7	69.7	36.8	58.0	19.6	32.3	50	50.9	37.1	
DenseASPP [39]	78.1	20.5	18.2	20.0	16.6	27.8	28.9	5.7	88.5	53.3	67.5	9.3	76.3	39.6	70.0	36.8	57.7	15.9	32.4	20	23.4	40.2	
SqueezeSeg [8]	68.8	16.0	4.1	3.3	3.6	12.9	13.1	0.9	85.4	26.9	54.3	4.5	57.4	29.0	60.0	24.3	53.7	17.5	24.5	<b>66</b>	1.0	29.5	
SqueezeSeg + CRF [8]	68.3	18.1	5.1	4.1	4.8	16.5	17.3	1.2	84.9	28.4	54.7	4.6	61.5	29.2	59.6	25.5	54.7	11.2	36.3	55	1.0	30.8	
SqueezeSegV2 [9]	81.8	18.5	17.9	13.4	14.0	20.1	25.1	3.9	88.6	45.8	67.6	17.7	73.7	41.1	71.8	35.8	60.2	20.2	36.3	50	1.0	39.7	
SqueezeSegV2 + CRF [9]	82.7	21.0	22.6	14.5	15.9	20.2	24.3	2.9	88.5	42.4	65.5	18.7	73.8	41.0	68.5	36.9	58.9	12.9	41.0	40	1.0	39.6	
RangeNet21 [10]	85.4	26.2	26.5	18.6	15.6	31.8	33.6	4.0	91.4	57.0	74.0	26.4	81.9	52.3	77.6	48.4	63.6	36.0	50.0	20	25.0	47.4	
RangeNet53 [10]	86.4	24.5	32.7	25.5	22.6	36.2	33.6	4.7	<b>91.8</b>	64.8	74.6	<b>27.9</b>	84.1	55.0	78.3	50.1	64.0	38.9	52.2	13	50.4	49.9	
RangeNet53 + k-NN [10]	91.4	25.7	<b>34.4</b>	25.7	23.0	38.3	38.8	4.8	<b>91.8</b>	<b>65.0</b>	<b>75.2</b>	27.8	87.4	58.6	80.5	55.1	64.6	47.9	55.9	12	50.4	52.2	
MINet	85.2	38.2	32.1	29.3	23.1	47.6	46.8	24.5	90.5	58.8	72.1	25.9	82.2	49.5	78.8	52.5	65.4	37.7	55.5	59	1.0	52.4	
MINet + k-NN	90.1	<b>41.8</b>	34.0	29.9	23.6	<b>51.4</b>	<b>52.4</b>	<b>25.0</b>	90.5	59.0	72.6	25.8	85.6	52.3	81.1	58.1	66.1	49.0	<b>59.9</b>	47	1.0	<b>55.2</b>	

\* The proposed MINet performs well for small objects.

## B. Ablation Study

1) *Modules*: As illustrated in Fig. 1, our network consists of three modules. In Tab. II we evaluate some design choices for the Mini Fusion Module (MFM) (Sec. III-A), the Multi-scale Interaction Module (MIM) (Sec. III-B), and the Up Fusion Module (UFM) (Sec. III-C). While the proposed approach achieves 51.8% mIoU (Row 4), we evaluate the impact of processing each modality separately before fusing them in the features space in the first row. To keep the number of parameters the same, we use  $3 \times 3$  convolutions to process the input multi-modal image instead of processing each modality separately in MFM. In this case, the accuracy drops by 0.9% (Row 1). This shows the benefit of processing each modality separately at the beginning. In the last row, we also report the result when we omit the depth from the input. In this case, the accuracy drops to 2.2%. In Fig. 1, we have connections between the three paths. If we remove the top-to-down interactions, the accuracy is reduced by 1.1% (Row 2). This demonstrates the benefit of allowing interactions between the multi-scale features. If the multi-resolution features are just resized, concatenated, and processed by convolutional layers, instead of using UFM, the accuracy is reduced by 1.2% (Row 3).

2) *Supervision Setting*: As discussed in Sec. III-D, we use additional supervision for MIM and UFM. For MIM, we use the semantic labels (S) to add the loss (3) to the top and middle path. For UFM, we add the loss (4) for the segment boundaries (E). In Tab. III, we report the mIoU for different settings. The best setting is achieved by adding semantic supervision to the top and middle path in MIM and applying edge supervision to UFM (Row 1). Removing any of this additional supervision leads to an accuracy loss by more than 1.3% (Row 2-4). If the additional supervision is only used for UFM, the accuracy even decreases further (Row 5). If we do not use any additional supervision, the accuracy is lowest and 3.4% below the proposed setting (Row 6). While we removed so far additional supervision, the last two rows in the table

show results when we add or change the type of supervision. If we add additional supervision to the bottom path, the accuracy decreases by 0.9%. This is due to the large difference between the ground-truth resolution and the resolution of the bottom path, which results in sampling artifacts that have a negative impact. Instead of using the edge loss (4) for UFM, we also replaced it by the semantic loss that is used for MIM. While adding the semantic loss (Row 8) is better than using no additional loss for UFM (Row 4), the edge loss (Row 1) achieves a 1% higher accuracy than the semantic loss. This is expected since the purpose of the edge loss is to improve the segment boundaries after the upscaling, which is done by the Up Fusion Module. We also investigated what happens if the focal loss [41] is used for the edge loss instead of (4) (Row 9). In this case, the accuracy decreases.

3) *Impact of  $\lambda$* : Our loss function (7) contains only one hyper-parameter, namely  $\lambda$ . We evaluate the impact of  $\lambda$  in Tab. IV. The setting  $\lambda = 0$  corresponds to row 5 of Tab. III where no additional supervision is added to the paths. Setting  $\lambda$  between 0.1 and 1.0 performs well.

4) *Path Settings*: As shown in Tab. I, we balance the computational resources across the three paths where we increase the complexity as the resolution decreases. In rows 1-10 of Tab. V, we report the results when we vary the number of blocks for the top, middle, and bottom path. The results show that increasing the parameters only for one path does not result in an improvement. For instance, using 9 instead of 3 MobileBlocks for the top path (Row 4) decreases the accuracy by 2.9%. This shows that a good computational balance between the paths is required. In rows 11-13, we report the results when we use the same operations for all paths, *i.e.*, either 3 or 5 MobileBlocks or 3 BasicBlocks. Using only MobileBlocks reduces the number of parameters, but it improves the runtime only slightly and this is only the case for 3 MobileBlocks. This, however, comes at a substantially lower accuracy. In terms of runtime and accuracy, the proposed setting provides a much better trade-off. If the computational

TABLE VII  
EVALUATION RESULTS ON THE SEMANTICPOSS DATASET.

	person	rider	car	trunk	plants	traffic sign	pole	building	fence	bike	road	mIoU
SqueezeSegV1 [8]	5.5	0.0	8.7	3.4	39.1	2.4	2.5	34.5	7.6	18.4	62.5	16.8
SqueezeSegV1 [8] + CRF	14.2	1.4	11.6	18.1	5.9	11.1	1.9	37.9	5.6	18.9	78.7	18.7
SqueezeSegV2 [9]	18.4	11.2	34.9	15.8	56.3	11.0	4.5	47.0	25.5	32.4	71.3	29.8
SqueezeSegV2 [9] + CRF	<b>23.9</b>	<b>22.6</b>	29.7	15.3	37.3	11.1	<b>5.3</b>	45.9	18.2	34.7	<b>73.4</b>	28.9
RangeNet53 [10]	10.0	6.2	33.4	7.3	54.2	5.5	2.6	49.9	18.4	28.6	63.5	25.4
RangeNet53 + k-NN	14.2	8.2	35.4	9.2	58.1	6.8	2.8	55.5	<b>28.8</b>	32.2	66.3	28.9
MINet	13.3	11.3	34.0	18.8	62.9	11.8	4.1	55.5	20.4	34.7	69.2	30.5
MINet + k-NN	20.1	15.1	<b>36.0</b>	<b>23.4</b>	<b>67.4</b>	<b>15.5</b>	5.1	<b>61.6</b>	28.2	<b>40.2</b>	72.9	<b>35.1</b>

TABLE VIII  
PERFORMANCE ON AN EMBEDDED PLATFORM (JETSON AGX).

Methods	Resolutions	GFLOPs	SPS	mIoU
RangeNet53 [10]	64 × 2048	360.5	7	49.9
	64 × 1024	180.3	11	45.4
	64 × 512	90.1	22	39.3
RangeNet53 + k-NN [10]	64 × 2048	360.5	5	52.2
	64 × 1024	180.3	8	48.0
	64 × 512	90.1	13	41.9
MINet	64 × 2048	6.2	24	52.4
	64 × 1024	3.2	47	49.1
	64 × 512	1.7	80	45.0
MINet + k-NN	64 × 2048	6.2	13	55.2
	64 × 1024	3.2	18	52.4
	64 × 512	1.7	21	48.5

\* The number of GFLOPs does not include the post-processing.

expensive BasicBlocks are used for all paths, the number of parameters and runtime nearly doubles while the accuracy is nearly the same. This shows that using the same operations for all resolutions is highly inefficient and that the proposed approach achieves a good balance between efficiency and effectiveness.

### C. Comparison with other Methods

We first compare the proposed approach (MINet) with other methods on the SemanticKITTI test set in terms of both accuracy and efficiency. For a fair comparison, all methods including image-based methods are trained from scratch. The results are shown in Tab. VI. In the first rows, we show the results for point-based methods. Most of the approaches are very slow and cannot process more than 2 scans per second since spatial aggregation operations are usually very time-consuming for large point clouds. The very recent works [17], [18], [35] are faster and process up to 16 scans per second, but it is difficult to deploy these networks on embedded systems due to their complex operations. The highest accuracy is achieved by [35], but the network is very large with over 16M parameters. Our proposed approach outperforms all point-based methods in terms of runtime, number of parameters, and accuracy.

As for image-based methods, we use four widely used methods, namely PSPNet [2], DeepLabV3+ [36], DenseASPP [39], and the lightweight model BiseNet [38]. We adjust the input channels so that these methods can be applied to the projection map. While these methods are faster than point-based methods, they have by far more parameters and the accuracy is significantly lower. This shows that projected LiDAR data cannot be directly processed by image-based segmentation methods since the modality differs from RGB images.

We also compare our approach to other projection-based methods. While SqueezeSeg [8], [9] has the same amount of

parameters and depending on the setting runs slightly faster, the accuracy is very low. The state-of-the-art projection-based method RangeNet [10] outperforms SqueezeSeg in terms of accuracy, but this is achieved by increasing the number of parameters to over 50M and decreasing the runtime. Our approach is much more efficient. It uses only 2% of the number of parameters compared to RangeNet53 and it is about 4× faster while achieving a higher accuracy. Fig. 5 visualizes the accuracy and runtime of the methods of Tab. VI and shows the effectiveness and efficiency of the proposed approach.

We also evaluate our method on SemanticPOSS [32] and compare our approach to other methods in Tab. VII. Since the dataset is smaller and the point clouds are more sparse compared to SemanticKITTI, the mIoU is lower for all methods. However, our approach still outperforms other methods with a large margin. This proves the effectiveness of our method.

### D. Performance on an Embedded Platform

We finally compare our method with RangeNet on an embedded platform. Here, we use a Jetson AGX that is an AI module for embedded systems, as it is usually used for autonomous driving. We optimize our method and RangeNet using TensorRT. The results are summarized in Tab. VIII. Since the input resolution can be decreased to reduce the runtime, we report the results for three different input resolutions. We can see that at each resolution, the proposed MINet outperforms RangeNet53 [10] with or without post-processing. Furthermore, MINet is also much faster than RangeNet53. Specifically, MINet is about 4× faster than RangeNet53 without post-processing and 2× faster with post-processing. Such high efficiency makes MINet quite suitable for robotics applications. Even with full resolution and post-processing, MINet runs at real-time since the LiDAR scan frequency is 10Hz. Compared to Tab. VI where the runtime is measured on a workstation with a single Quadro P6000, the post-processing has a higher impact on the runtime for the embedded platform since the post-processing is not optimized by TensorRT. Moreover, the post-processing is applied to the point cloud, so it cannot benefit from reducing the resolution of the projection map.

## V. CONCLUSION

In this work, we proposed a novel lightweight projection-based method, called Multi-scale Interaction Network (MINet), for semantic segmentation of LiDAR data. The network is highly efficient and runs in real-time on GPUs and embedded platforms. It outperforms point-based, image-based, and projection-based methods in terms of accuracy,

number of parameters, and runtime. This is achieved by using a multi-scale approach where the computational resources are balanced between the scales and by introducing interactions between the scales. By processing the modalities separately before fusing them and adding additional different types of supervision, we could further improve the accuracy without decreasing the runtime. Compared to the state-of-the-art projection-based method RangeNet, MINet reduces the number of parameters by 98% and is 4× faster while achieving a higher accuracy. Since MINet processes more than 24 scans per second on an embedded platform, it can be used for autonomous vehicles and robots. Our method also achieves good performance on other tasks, like moving object segmentation [42]. Projection-based methods, however, have some limitations. For instance, it is not straightforward to integrate temporal information from multiple views. Finally, we expect that the design principles of the network are also valuable for other tasks like 3D car detection. The source code is available at <https://github.com/sj-li/MINet>.

## REFERENCES

- [1] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, “Rethinking atrous convolution for semantic image segmentation,” *arXiv preprint arXiv:1706.05587*, 2017.
- [2] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, “Pyramid scene parsing network,” in *IEEE CVPR*, 2017, pp. 2881–2890.
- [3] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, “Searching for MobileNetV3,” in *IEEE CVPR*, 2019, pp. 1314–1324.
- [4] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, “ShuffleNet V2: Practical guidelines for efficient CNN architecture design,” in *ECCV*, 2018, pp. 116–131.
- [5] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “PointNet: Deep learning on point sets for 3D classification and segmentation,” in *IEEE CVPR*, 2017, pp. 652–660.
- [6] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “PointNet++: Deep hierarchical feature learning on point sets in a metric space,” in *NIPS*, 2017, pp. 5099–5108.
- [7] W. Wu, Z. Qi, and L. Fuxin, “PointConv: Deep convolutional networks on 3D point clouds,” in *IEEE CVPR*, 2019, pp. 9621–9630.
- [8] 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 *IEEE ICRA*, 2018, pp. 1887–1893.
- [9] 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 *IEEE ICRA*, 2019, pp. 4376–4382.
- [10] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, “RangeNet++: Fast and accurate LiDAR semantic segmentation,” in *IEEE/RSJ IROS*, 2019, pp. 4213–4220.
- [11] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional networks for biomedical image segmentation,” in *MICCAI*, 2015, pp. 234–241.
- [12] H. Zhang, K. Dana, J. Shi, Z. Zhang, X. Wang, A. Tyagi, and A. Agrawal, “Context encoding for semantic segmentation,” in *IEEE CVPR*, 2018, pp. 7151–7160.
- [13] Y. Liu, M.-M. Cheng, X. Hu, J.-W. Bian, L. Zhang, X. Bai, and J. Tang, “Richer convolutional features for edge detection,” *IEEE TPAMI*, vol. 41, no. 8, pp. 1939–1946, 2019.
- [14] Y. Liu, M.-M. Cheng, D.-P. Fan, L. Zhang, J. Bian, and D. Tao, “Semantic edge detection with diverse deep supervision,” *arXiv preprint arXiv:1804.02864*, 2018.
- [15] M. Tatarchenko, J. Park, V. Koltun, and Q.-Y. Zhou, “Tangent convolutions for dense prediction in 3D,” in *IEEE CVPR*, 2018, pp. 3887–3896.
- [16] L. Landrieu and M. Simonovsky, “Large-scale point cloud semantic segmentation with superpoint graphs,” in *IEEE CVPR*, 2018, pp. 4558–4567.
- [17] 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,” *arXiv preprint arXiv:1911.11236*, 2019.
- [18] S. Li, Y. Liu, and J. Gall, “Projected-point-based segmentation: A new paradigm for LiDAR point cloud segmentation,” *arXiv preprint arXiv:2008.03928*, 2020.
- [19] 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 <0.5MB model size,” *arXiv preprint arXiv:1602.07360*, 2016.
- [20] J. Redmon and A. Farhadi, “YOLOv3: An incremental improvement,” *arXiv preprint arXiv:1804.02767*, 2018.
- [21] X. Chen, A. Milioto, E. Palazzolo, P. Giguère, J. Behley, and C. Stachniss, “SuMa++: Efficient LiDAR-based semantic SLAM,” in *IEEE/RSJ IROS*, 2019, pp. 4530–4537.
- [22] X. Chen, T. Läbe, A. Milioto, T. Röhling, O. Vysotska, A. Haag, J. Behley, C. Stachniss, and F. Fraunhofer, “OverlapNet: Loop closing for LiDAR-based SLAM,” in *RSS*, 2020.
- [23] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “MobileNets: Efficient convolutional neural networks for mobile vision applications,” *arXiv preprint arXiv:1704.04861*, 2017.
- [24] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted residuals and linear bottlenecks,” in *IEEE CVPR*, 2018, pp. 4510–4520.
- [25] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *IEEE CVPR*, 2016, pp. 770–778.
- [26] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, “Deeply-supervised nets,” in *Artificial Intelligence and Statistics*, 2015, pp. 562–570.
- [27] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, “Learning a discriminative feature network for semantic segmentation,” in *IEEE CVPR*, 2018, pp. 1857–1866.
- [28] A. Kirillov, E. Levinkov, B. Andres, B. Savchynskyy, and C. Rother, “InstanceCut: from edges to instances with MultiCut,” in *IEEE CVPR*, 2017, pp. 5008–5017.
- [29] M. Berman, A. Rannen Triki, and M. B. Blaschko, “The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks,” in *IEEE CVPR*, 2018, pp. 4413–4421.
- [30] 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 *IEEE ICCV*, 2019, pp. 9297–9307.
- [31] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, J. Gall, and C. Stachniss, “Towards 3d lidar-based semantic scene understanding of 3d point cloud sequences: The SemanticKITTI dataset,” *The International Journal of Robotics Research*, 2021.
- [32] Y. Pan, B. Gao, J. Mei, S. Geng, C. Li, and H. Zhao, “Semanticpos: A point cloud dataset with large quantity of dynamic instances,” *arXiv preprint arXiv:2002.09147*, 2020.
- [33] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? The KITTI vision benchmark suite,” in *IEEE CVPR*, 2012, pp. 3354–3361.
- [34] 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 *IEEE CVPR*, 2018, pp. 2530–2539.
- [35] Y. Zhang, Z. Zhou, P. David, X. Yue, Z. Xi, B. Gong, and H. Foroosh, “PolarNet: An improved grid representation for online LiDAR point clouds semantic segmentation,” in *IEEE CVPR*, 2020, pp. 9601–9610.
- [36] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, “Encoder-decoder with atrous separable convolution for semantic image segmentation,” in *ECCV*, 2018, pp. 801–818.
- [37] H. Li, P. Xiong, J. An, and L. Wang, “Pyramid attention network for semantic segmentation,” *arXiv preprint arXiv:1805.10180*, 2018.
- [38] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, “BiSeNet: Bilateral segmentation network for real-time semantic segmentation,” in *ECCV*, 2018, pp. 325–341.
- [39] M. Yang, K. Yu, C. Zhang, Z. Li, and K. Yang, “DenseASPP for semantic segmentation in street scenes,” in *IEEE CVPR*, 2018, pp. 3684–3692.
- [40] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes challenge: A retrospective,” *IJCV*, vol. 111, no. 1, pp. 98–136, 2015.
- [41] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” in *IEEE ICCV*, 2017, pp. 2980–2988.
- [42] X. Chen, S. Li, B. Mersch, L. Wiesmann, J. Gall, J. Behley, and C. Stachniss, “Moving Object Segmentation in 3D LiDAR Data: A Learning-based Approach Exploiting Sequential Data,” vol. 6, pp. 6529–6536, 2021. [Online]. Available: <http://www.ipb.uni-bonn.de/pdfs/chen2021ral-iros.pdf>