SENET: TOWARD SIMPLE AND EFFICIENT LIDAR SEMANTIC SEGMENTATION FOR AUTONOMOUS DRIVING

Anonymous ICME submission

ABSTRACT

Accurate and fast scene understanding is one of the challenging task for autonomous driving, which requires to take full advantage of LiDAR point clouds for semantic segmentation. In this paper, we present a **simple** and **efficient** image-based semantic segmentation network, named **SENet**. In order to improve the descriptive power of learned features and reduce the computational as well as time complexity, our SENet integrates the convolution with larger kernel size instead of MLP, carefully-selected activation functions, and multiple auxiliary segmentation heads with corresponding loss functions into architecture. Quantitative and qualitative experiments conducted on publicly available benchmarks, SemanticKITTI and SemanticPOSS, demonstrate that our pipeline achieves much better mIoU and inference performance compared with state-of-the-art models. The code will be available in Github.

Index Terms— LiDAR Point Cloud, Autonomous Driving, Semantic Scene Understanding, Semantic Segmentation

1. INTRODUCTION

Recently, the rapid development of LiDAR sensors makes the 3D computer vision become an interesting research topic in applications, such as robotics and autonomous driving [1], where accurate, real-time and robust environment perception and understanding is a challenging task. In order to achieve this goal, LiDAR becomes the most popular and widely used choice, since 1) compared with visual cameras, LiDAR is much more robust to varying lighting and weather conditions. 2) and the acquired 3D point clouds provide rich geometric information. Therefore, LiDAR-based semantic scene perception, especially 3D point cloud semantic segmentation, has received increasing attention, aiming to perform point-wise classification.

In the past few years, the emergence of datasets, such as SemanticKITTI [2] and SemanticPOSS [3], benchmarks the LiDAR semantic segmentation. And they make it possible to apply deep learning techniques to this task. However, we cannot directly perform standard Convolutional Neural Networks (CNNs) on 3D point clouds due to their irregular and sparse structure. To address this problem, many recent new methods have been proposed, which can be classified into raw point-based [4, 5, 6], voxel-based [7, 8], and range image-based [9, 10].

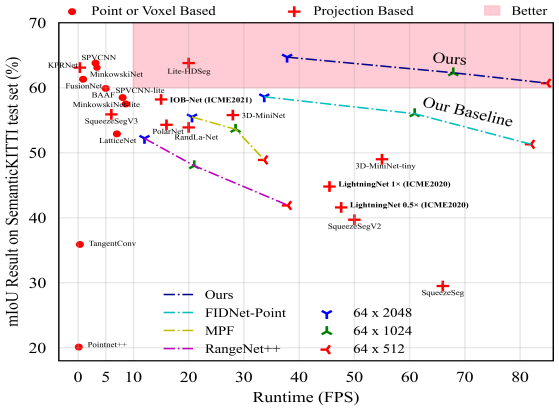


Fig. 1. LIDAR semantic segmentation **accuracy** vs **speed** on SemanticKITTI test set [2]. (The dotted lines represent the performance of the same method at different resolutions.) *Best viewed in color and zoomed in for more detail.*

Generally, point-based networks deal with raw 3D LiDAR point clouds directly, which can obtain much better performance with higher computational complexity [5] [6]. Voxel-based methods project unstructured point clouds into regular grid cells, and allows the usage of 3D convolutional neural networks. Although the state-of-the-art accuracy can be achieved, the higher model complexity makes it difficult for these methods to obtain real-time inference speed. For example, the inference speeds of SPVCNN [7] and Cylinder3D [8] on Tesla V100 are only 8.0 and 7.6 fps, respectively. In addition, range image-based approaches choose to represent the raw 3D point clouds as an ordered range image using spherical projection strategy, then the well-designed 2D CNNs can be utilized to carry out LiDAR semantic segmentation task. This kinds of methods actually can provide superior inference and accuracy performance, which drives more and more researchers to focus on this field [11], though they inevitably suffer from information loss during projection [12].

On the other hand, the current range image-based methods, such as KPRNet [13] and Lite-HDSeg [14] typically have an extremely large number of parameters, and much lower inference speed or higher model complexity, which limits their applications in autonomous driving to some extent. In order to relieve this issue, FIDNet [15] presents a new range image-based LiDAR semantic segmentation network that keeps the

solution as simple as possible while maintaining good performance.

Nevertheless, the performance of FIDNet is not comparable to the current state-of-the-art methods [16, 7, 14]. So based on the fundamental idea and performance of FIDNet, we attempt to rethink its design choices, and propose a simple and efficient LiDAR semantic segmentation model, termed SENet. Quantitatively experimental results demonstrate that our network performance can outperform current state-of-the-art methods with no increase in the number of effective parameters, and has a much higher inference speed (as shown in Fig. 1). Specifically, the main contributions of this paper are as follows:

- We present a newly-designed LiDAR point cloud segmentation architecture, named SENet, which can improve the inference speed at the cost of no parameters increment.
- In order to improve the nonlinear capability of the network, we use SiLU and Hardswish to adjust the activation functions.
- By introducing multiple auxiliary segmentation heads, we significantly improve the learning power of our network without introducing additional inference parameters.
- We conduct comprehensive experiments on the publicly available datasets, SemanticKITTI and SemanticPOSS. The results show that our method achieves state-of-the-art performance.

2. RELATED WORK

Point-based methods work directly on the raw point clouds. PointNet [17] and PointNet++[4] are pioneering studies to use shared MLPs to learn the properties of each point, which inspire the appearance of a series of point-based networks. KPConv [5] develops deformable convolutions that can use arbitrary number of kernel points to learn local representations. RandLA-Net [6] adopts a random sampling strategy to considerably improve the efficiency of point cloud processing and uses local feature aggregation to reduce the information loss caused by random operations. BAAF [16] makes full use of the geometric and semantic features of points to obtain more accurate semantic segmentation by using bilateral structures and adaptive fusion methods.

Voxel-based methods first discretize the point clouds into 3D voxel representations, then predict semantic labels for these voxels using 3D CNN frameworks. Minkowski [18] chose to use sparse convolution instead of standard 3D convolution to reduce the computational cost. SPVNAS [7] exploits neural structure search (NAS) to further improve the network's performance. Cylinder3D [8] transforms input point cloud into a specialized cylindrical grid and introduces an asymmetric 3D convolutional network to handle the sparsity and density variation of outdoor LiDAR point clouds.

Image-based methods project LiDAR point clouds onto 2D multimodal images and then apply the well-designed 2D CNNs for semantic segmentation. SqueeeSeg [9] and SqueezeSegV2 [19] use the lightweight model SqueezeNet and CRF for segmentation. RangeNet++ [10] integrates Darknet into SqueezeSeg and proposes an efficient KNN post-processing method to predict labels for point. SqueezeSegV3 [20] proposes Spatially-Adaptive Convolution (SAC) with different filters depending on the location of the input image. KPRNet [13] achieves promising results by using powerful as backbone together with KPConv as segmentation head. Lite-HDSeg [14] achieves state-of-the-art performance by introducing three different modules, Inception-like Context Module, Mutli-class Spatial Propagation Network, and a boundary loss. Although it gets a high inference speed, the higher number of parameters and model complexity make it less suitable for autonomous driving applications.

3. METHODOLOGY

In order to take full advantage of well-optimized conventional convolution functions for real-time LiDAR point cloud segmentation, we use the spherical projection approach to generate 2D multimodal range images by projecting the LiDAR point cloud into the spherical coordinate system, which is formulated as,

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\left[1 - arctan\left(y, x\right)\pi^{-1}\right] W \\ \left[1 - \left(arcsin\left(z, d^{-1}\right) + f_u\right)\frac{1}{f_u + f_d}\right] H \end{pmatrix} \quad (1)$$

where $f = f_u + f_d$ refers to the sensor's vertical field-of-view. The depth d of each point is calculated as $d = \sqrt{x^2 + y^2 + z^2}$. The final result is a projected range image of size $(H, W, 5)$, where each pixel contains 5 channels (x, y, z, d, r) and r is the intensity information of points. Based on this transformation, the point cloud segmentation problem is turned into an image segmentation task.

3.1. Network Architecture

Our SENet architecture is shown in Fig 2. We will detailedly introduce the core components in the following subsections.

Input Module, Classification Head and Activation Function. The range image is a special two-dimensional image with five channels, where each location actually can be considered as a point representation. Therefore, FIDNet [15] uses 1×1 conv layers in **Input Module** and **Classification Head** to process the input features, which, the authors think, can achieve the similar effect as MLP used in PointNet for point feature learning. However, it is more reasonable to use 3×3 conv layers instead of 1×1, due to the following reasons. 1) Unlike disordered and unstructured point cloud, the generated range images usually have structure features, which makes 3×3 conv much more suitable than MLP. 2) For 1×1 conv, the lower number of parameters and computational cost donnot mean faster inference speed. As shown in Table 1 from RepVGG [21], it can be concluded that under the same

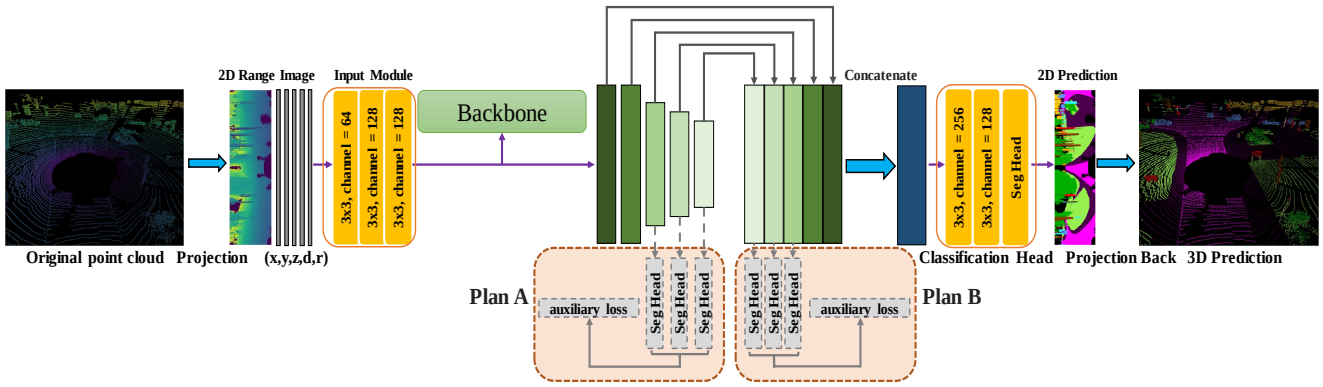


Fig. 2. The overall architecture of our pipeline. The backbone network can be any feature extraction structure such as BasicBlock in ResNet used in this paper. Plan A and Plan B are two different designs of auxiliary loss. As shown in the dotted line, it can be removed during inference thus not influencing the inference speed.

Table 1. Inferecne speed test with varying kernel size on NVIDIA 1080Ti. The batch size = 32, input channels = output channels = 2048, resolution = 56×56, stride = 1.

Kernel size	Theoretical FLOPs (B)	Time usage (ms)	Theoretical TFLOPS
1×1	420.9	84.5	9.96
3×3	3788.1	198.8	**38.10**

condition, the computational density (theoretical operations divided by time usage) of 3×3 conv can be up to $4\times$ that of 1×1 conv on a 1080Ti GPU. Therefore, we attempt to conduct a simple substitution to improve the inference speed, which will be verified in **Sec 4.3 Ablation Studies**. In addition, inspired by YoloV5 and MobileNetV3, we can see that the usage of a stronger nonlinear activation function can contribute to the improvement of network's expressive power without increasing the parameters. Hence, we adopt SiLU and Hardswish activation functions in our model. Experiments in **Sec 4.3** also show that both functions can enhance our model's performance with little influence on inference speed.

Loss Function. In order to solve problems **1)** Class imbalance, **2)** The problem of optimizing the intersection-over-union (IoU), **3)** Blurred segmentation boundaries, following [14], we use three differentloss functions, namely weighted cross-entropy loss L_{wce}, Lovász-Softmax loss L_{ls} and boundary loss L_{bd}, to supervise our model.

For segmentation task, the boundary-blurring problems between different objects generally arise from upsampling and downsampling operations. To address this issue, we introduce a boundary loss function, which can be formally defined as,

$$L_{bd}(\hat{y}, y) = 1 - \frac{2P^c R^c}{P^c + R^c} \tag{2}$$

Where P^c and R^c denote the precision and recall of the predicted boundary map y_{pd} with respect to the ground truth y_{gt} for class c. We give the definition of boundaries as,

$$\begin{cases} y^b_{gt} = pool(1 - y_{gt}, \theta_0) - (1 - y_{gt}) \\ y^b_{pd} = pool(1 - y_{pd}, \theta_0) - (1 - y_{pd}) \end{cases} \tag{3}$$

Here pool$(\cdot)$ refers to the max-pooling operation on a sliding window of size θ_0. Finally, our total loss is the weighted combination of these three loss functions. The formulation is

$$L = \alpha L_{wce} + \beta L_{ls} + \gamma L_{bd} \tag{4}$$

where α, β, and γ are the corresponding weights and are set to 1.0, 1.5, 1.0 respectively. In addition, we set θ_0 in L_{bd} to be 3.

Auxiliary Loss. In FIDNet, the authors integrate bilinear upsampling methods into FID module to interpolate the low-resolution feature maps, which generates five point-wise feature tensors with the same resolution but encoding different levels of information. Compared with conventional decoders used in other networks [22, 14, 13], FID is totally parameter-free and dramatically reduces the complexity and storage cost. However, this simple decoder make model's performance excessively depend on the low-dimensional and high-dimensional features. Additionally, unlike progressive upsampling decoder, the simple interpolation fusion decoding may results in feature maps at different scales not being fully aligned and decoded. To alleviate this problem, we introduce multiple auxiliary loss heads to refine the feature maps at different resolutions for learning capability improvement.

Specifically, we use the auxiliary segmentation head to predict the output of three feature maps with different resolutions, and compute the weighted loss together with the main loss to supervise our network to produce more semantic features. The final loss function can be defined as,

$$L_{total} = L_{main} + \lambda \sum_{i=1}^{3} L(y_i, \hat{y}_i) \tag{5}$$

where L_{main} is the main loss, y_i is the semantic output obtained from stage i, and $\hat{y}_i$ represents the corresponding semantic label. $L(\cdot)$ is computed according to Equation 4. As shown in Fig. 2, for Plan A, $\hat{y}_i$ is obtained by downsampling the GT labels with corresponding rate. For Plan B, since all feature maps are upsampled to the final output size, the GT labels are their $\hat{y}_i$.

Table 2. The performance comparison on SemanticKITTI test set.

Category	Methods	Size	FPS (Hz)	mean-IoU	car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign
Point-based	PointNet++ [4]	50K pts	0.1	20.1	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
	RandLa-Net [6]		20	53.9	94.2	26.0	25.8	40.1	38.9	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
	KPConv [5]		–	58.8	**96.0**	30.2	**42.5**	33.4	44.3	**61.5**	**61.6**	11.8	88.8	61.3	72.7	**31.6**	**90.5**	**64.2**	**84.8**	**69.2**	**69.1**	**56.4**	47.4
	BAAF [16]		5	59.9	95.4	**31.8**	35.5	**48.7**	**46.7**	49.5	55.7	**53.0**	**90.9**	**62.2**	**74.4**	23.6	89.8	60.8	82.7	63.4	67.9	53.7	**52.0**
Voxel-based	MinkowskiNet-lite [18]	Voxel	8.6	57.5	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	MinkowskiNet [18]		3.4	63.1	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	SPVCNN-lite [7]		8.1	58.5	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	SPVCNN [7]		3.2	63.8	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Image-based	SqueezeSeg-CRF [9]	64 × 2048	55	30.8	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
	SqueezeSegV2-CRF [19]		40	39.6	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
	SqueezeSegV3 [20]		6	55.9	92.5	38.7	36.5	29.6	33.0	45.6	46.2	20.1	91.7	63.4	74.8	26.4	89.0	59.4	82.0	58.7	65.4	49.6	58.9
	SalsaNext [22]		24	59.5	91.9	48.3	38.6	**38.9**	31.9	60.2	59.0	19.4	91.7	63.7	75.8	29.1	90.2	64.2	81.8	63.6	66.5	54.3	62.1
	Lite-HDSeg [15]		20	63.8	92.3	40.0	**55.4**	37.7	39.6	59.2	**71.6**	**54.3**	93.0	68.2	78.3	29.3	91.5	65.0	78.2	65.8	65.1	**59.5**	**67.7**
	KPRNet [15]		0.3	63.1	**95.5**	54.1	47.9	23.6	**42.6**	**65.9**	65.0	16.5	**93.2**	**73.9**	**80.6**	**30.2**	**91.7**	**68.4**	**85.7**	**69.8**	**71.2**	58.7	64.1
	RangeNet++ [10]	64 × 512	38.5	41.9	87.4	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
	MPF [23]		33.7	48.9	91.1	22.0	19.7	18.8	16.5	30.0	36.2	4.2	91.1	61.9	74.1	29.4	86.7	56.2	**82.3**	51.6	**68.9**	38.6	49.8
	FIDNet-Point [15]		82.0	51.3	90.4	28.6	30.9	34.3	27.0	43.9	48.9	16.8	90.1	58.7	71.4	19.9	84.2	51.2	78.2	51.9	64.5	32.7	50.3
	Ours		**84.9**	**60.7**	**92.1**	**45.4**	**42.9**	**43.9**	**46.8**	**56.4**	**63.8**	**29.7**	91.3	**66.0**	**75.3**	**31.1**	**88.9**	**60.4**	81.9	**60.5**	67.6	**49.5**	**59.1**
	RangeNet++ [10]	64 × 1024	23.3	48.0	90.3	20.6	27.1	25.2	17.6	29.6	34.2	7.1	90.4	52.3	72.7	22.8	83.9	53.3	77.7	52.5	63.7	43.8	47.2
	MPF [23]		28.5	53.6	92.7	28.2	30.5	26.9	25.2	42.5	45.5	9.5	90.5	64.7	**74.3**	**32.0**	88.3	59.0	**83.4**	56.6	**69.8**	46.0	54.9
	FIDNet-Point [15]		60.9	56.0	92.4	44.0	41.5	33.2	30.8	57.9	52.6	18.0	**91.0**	61.2	73.8	12.6	88.2	57.9	80.8	59.5	65.1	45.3	58.4
	Ours		**67.9**	**62.3**	**93.0**	**50.5**	**47.6**	**41.7**	**43.4**	**64.5**	**65.2**	**32.5**	90.5	**65.5**	74.1	29.2	**90.9**	**65.4**	81.6	**65.4**	65.6	**55.9**	**61.0**
	RangeNet++ [10]	64 × 2048	12.8	52.2	91.4	25.7	34.4	25.7	23.0	38.3	38.8	4.8	**91.8**	**65.0**	**75.2**	27.8	87.4	58.6	80.5	55.1	64.6	47.9	55.9
	MPF [23]		20.6	55.5	**93.4**	30.2	38.3	26.1	28.5	48.1	46.1	18.1	90.6	62.3	74.5	30.6	88.5	59.7	83.5	59.7	69.2	49.7	58.1
	FIDNet-Point [15]		33.7	58.6	93.0	45.7	42.0	27.9	32.6	62.6	58.1	30.5	90.8	58.3	74.9	20.1	88.5	59.5	83.1	64.3	67.8	52.6	60.0
	Ours		**37.8**	**64.7**	91.9	**58.6**	**50.3**	**40.6**	**42.3**	**68.9**	**65.9**	**43.5**	90.3	60.9	75.1	**31.5**	**91.0**	**66.2**	**84.5**	**69.7**	**70.0**	**61.5**	**67.6**

Table 3. Evaluation results on the SemanticPOSS test split.

	person	rider	car	truck	plants	traffic sign	pole	trashcan	building	cone/stone	fence	bike	ground	mIoU
SqueezeSeg [9]	14.2	1.0	13.2	10.4	28.0	5.1	5.7	2.3	43.6	0.2	15.6	31.0	75.0	18.9
SqueezeSeg + CRF [9]	6.8	0.6	6.7	4.0	2.5	9.1	1.3	0.4	37.1	0.2	8.4	18.5	72.1	12.9
SqueezeSegV2 [19]	48.0	9.4	48.5	11.3	50.1	6.7	6.2	14.8	60.4	5.2	22.1	36.1	71.3	30.0
SqueezeSegV2 + CRF [19]	43.9	7.1	47.9	18.4	40.9	4.8	2.8	7.4	57.5	0.6	12.0	35.3	71.3	26.9
RangeNet53 [10]	55.7	4.5	34.4	13.7	57.5	3.7	6.6	23.3	64.9	6.1	22.2	28.3	72.9	30.3
RangeNet53 + KNN [10]	57.3	4.6	35.0	14.1	58.3	3.9	6.9	24.1	66.1	6.6	23.4	28.6	73.5	30.9
MINet [24]	61.8	12.0	63.3	22.2	68.1	16.3	29.3	28.5	74.6	25.9	31.7	44.5	76.4	42.7
MINet + KNN [24]	62.4	12.1	63.8	22.3	68.6	16.7	30.1	28.9	75.1	28.6	32.2	44.9	76.3	43.2
FIDNet-Point [15]	71.6	22.7	71.7	22.9	67.7	21.8	27.5	15.8	72.7	31.3	40.4	50.3	79.5	45.8
FIDNet-Point + KNN [15]	72.2	**23.1**	72.7	23.0	68.0	**22.2**	28.6	16.3	73.1	**34.0**	40.9	50.3	79.1	46.4
Ours	74.9	21.8	77.0	25.3	72.0	18.0	30.9	46.9	75.9	26.1	47.5	51.7	**80.7**	49.9
Ours + KNN	**75.5**	22.0	**77.6**	**25.3**	**72.2**	18.2	**31.5**	**48.1**	**76.3**	27.7	**47.7**	**51.4**	80.3	**50.3**

4. EXPERIMENT

4.1. Dataset and Implementation details

SemanticKITTI is a large-scale dataset for the task of point cloud segmentation of autonomous driving scenes. It contains 43,551 LiDAR scans from 22 sequences collected from a city in Germany, where sequences 00 to 10 (19,130 scans) are used for training, 11 to 21 (20,351 scans) for testing, and sequence 08 (4,071 scans) for validation. All sequences are labeled with dense point-wise annotations.

SemanticPOSS is a much smaller, sparser, and more challenging benchmark collected by Peking University. It consists of 2,988 different, complex LiDAR scenes, each with a large number of sparse dynamic instances (e.g. pedestrians and bicycles). SemanticPOSS is divided into 6 parts, where we use part 2 as test set, and others as training set.

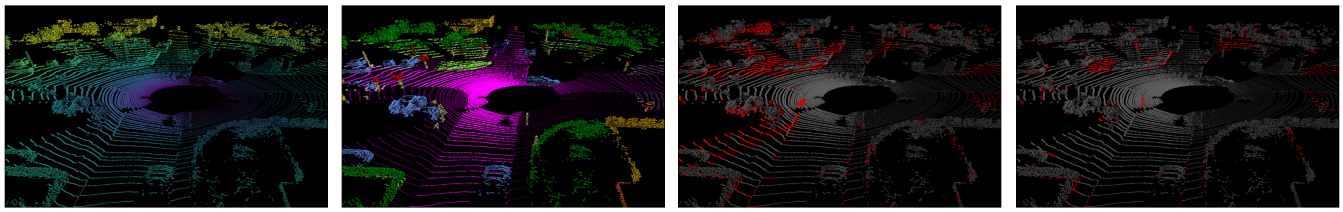
Implementation details. We conduct all the experiments on a single NVIDIA RTX3060 and RTX3090 GPU. The Stochastic Gradient Descent (SGD) optimizer with momentum of 0.9 is use for network optimization. During training, we adopt random rotation, random point dropout, and addition of random noise to X, Y, Z values to perform data augmentation. The wegiht decay is set to $1e^{-4}$. For SemanticsKITTI, we train network for 100 epochs with initial learning rate $1e^{-2}$, which is dynamically adjusted by a cosine annealing scheduler. For SemantciPOSS, the network is trained for 3 cycles with 45 epochs, the min and max learning rate are set to $1e^{-5}$ and $1e^{-3}$, respectively.

4.2. Results and Discussion

Quantitative Results on SemanticKITTI: Table 2 shows the quantitative results of several state-of-the-art models on SmanticKITTI benchmark. It reports the input size, frames per second (FPS), mean IoU and class-wise IoU. The best results are highlighted in bold. From these results, it can be seen that for input of size 64×2048, our model achieves the state-of-the-art performance (64.7%mIoU) compared to both point-based, voxel-based methods or image-based methods, while maintaining a higher FPS (37.8 FPS). For 64×1024 and 64×512 input, SENet obtain the superior results, 67.9 FPS, 62.3%mIoU and 84.9FPS, 60.7%mIoU, which outperforms the current methods by a large margin. It is worth noting that our SENet shows excellent performance when the input is 64×512, surpassing the performance of the baseline method FIDNet [15] and many other methods [20, 22, 23] under 64×2048 input.

Quantitative Results on SemanticPOSS: Table 3 shows the comparison of our proposed SENet with other related works. As can be reported from these results, all methods attain a little worse results due to the differences in sensors and environments, as well as the samll-scale and sparse structure of features. Nevertheless, it is worth noting that our method outperforms all models not only in overall mIoU, but also almost class-wise mIoU. This further verify the effectiveness and efficiency of our network. Specifically, we achieve a significant 7.1% and 3.9% mIoU improvements over the original best model MINet [24] and baseline method FIDNet [15].

Qualitative Result: To better visualize the enhancement of our model over the baseline, we provide qualitative compari-



(a) Input Point Cloud (b) Ground Truth (c) FIDNet-Point (d) SENet(ours)

Fig. 3. Qualitative analysis on SemanticKITTI validation set. Where **(a)** and **(b)** are input data of the LIDAR scan frame and corresponding segmentation ground truth, **(c)** and **(d)** are segmentation error maps in this scan frame for FIDNet and our method. *(With red indicating wrong prediction)*

Table 4. Ablation study evaluated on SemanticKITTI validation set.

Baseline	Row	KS	SiLU	H-swish	Plan A	Plan B	**mIoU**	**Params(M)**	**Latency(ms)**
FIDNet	1						55.4	6.053	11.956
Ours	2						58.9	6.053	11.956
	3	✓					59.2	6.774	11.576
	4		✓				60.0	6.053	11.981
	5			✓			60.5	6.053	12.039
	6	✓	✓				59.5	6.774	11.776
	7	✓		✓			60.6	6.774	11.562
	8				✓		62.5	6.061	11.956
	9					✓	63.0	6.061	11.956
	10	✓			✓		63.2	6.782	11.576
	11	✓				✓	64.3	6.782	11.576
	12	✓		✓		✓	65.3	6.782	11.562

Table 5. Impact of kenel size on time of model inference

Kernel Size	Input Resolution	Model Latency(ms)	FPS
1×1	64×2048	29.347	33.7
	64×1024	16.151	60.9
	64×512	11.956	82.0
3×3	64×2048	26.128	37.8
	64×1024	14.578	67.4
	64×512	11.576	84.8

son examples in Fig 3. As can be seen from the results, our approach demonstrates a significant improvement over the baseline and is closer to the ground truth.

4.3. Ablation Studies

To quantitatively analyze the effectiveness of different components, we conducted the following ablation experiments on the SemanticKITTI validation set. Here, we exploit a similar setup as SqueezeSegV3 [20] for high efficient training and evaluation. The size of input range image is set to 64×512. And we choose to list the accuracy evaluated directly on the projected 2D image instead of original 3D points.

Effects of Module Components. Table 4 demonstrates the experimental results of different design choices of network, which uses the mIoU, traning parameters required by models, and inference time as measures. Results in **the first row** are obtained by using the official FIDNet code. **The second line** reports the performance achieved using our network, in which the required normal vector has been removed. It still get 3.5% improvement over FIDNet. **The third row** validates the effectiveness when replacing the 1×1 convs with 3×3 convs. It can be concluded that although the 3×3 conv introduces more parameters than 1×1 conv, the former can further reduce the model's inference time, and slightly improve the performance by 0.3%. This is mainly because large kernel size brings much larger perceptual field, and the modern computational library is highly optimized. Table 5 further reports the difference between 3×3 conv and 1×1 conv in model latency and total FPS for different input range image resolutions. Overall, 3×3 conv can improve the model inference speed significantly. **The fourth to seventh rows** state the performance using different activation functions. From these results, we can see that the introduction of stronger nonlinearities contributes to the descriptiveness improvement of our model at little cost of inference speed. Results in **the eighth to eleventh lines** show that 1) both auxiliary segmentation heads can significantly improve the performance of our SENet. Although the integration of the auxiliary heads introduces some additional training parameters and increases the training time, we can remove these auxiliary heads in the inference phase. Therefore, they have no effect on the network's latency. 2) The combination of 3×3 conv and auxiliary loss outperforms the effect in the original 1×1 conv with auxiliary loss. And 3) since Plan B module calculates the loss directly based on the upsampled feature maps, which serves to refinement of features at different stages, Plan B brings a much better performance than Plan A.

Table 6. Impact of λ in auxiliary loss.

λ	0	0.1	0.5	1.0
mIoU	59.2	61.1	62.8	64.3

Table 7. Ablation studies for auxiliary loss module on different backbone.

Backbones	Params(M)	mIOU
HarDNet(vinilla)	2.735	55.1
HarDNet(ours)	3.138	58.7
HarDNet(ours) + Plan B	3.146	61.3

Effects of λ in auxiliary loss. The hyperparameter λ contained in Equation 5 play an important role in optimizing the model's performance. Hence, we evaluate the effectiveness of λ. As illustrated in Table 6, the introductions of additional suerpveise loss terms can help improve the segmentation performance. And the best results is achieved with $\lambda = 1.0$.

Effectiveness on Different Backbone. Here, we use a different feature extraction backbone, **HarDNet** (Harmonic DenseNet), to verify the generalization ability of the auxiliary loss modules. Compared with ResNet and DenseNet, HarDNet can achieve comparable accuracy in several tasks while

significantly reducing GPU running time. First, we use FC-HarDNet-70 to conduct experiment, and obtain 55.1% mIoU. Then, we carefully optimize the structure of HarDNet to construct a more powerful baseline with addition of fewer number of parameters. Finally, we integrate the auxiliary losses, and achieve a consistent model performance improvement.

5. CONCLUSION

In this paper, we propose a newly-designed networks,termed SENet, for LiDAR point cloud segmentation task. It is a simple and efficient model. Based on analysis of previous studies, we choose to use standard convolution with larger kernel size, and the SiLU as well as Hardswish activation functions, to improvement learning capability of our network. Then, we embed multiple auxiliary segmentation heads to further improve the power of learned features without introduction of parameters and efficiency cost. Experimental results on SemanticKITTI and SemanticPOSS demonstrate that our SENet can achieve state-of-the-art performance.

6. REFERENCES

[1] D. Feng, C. Haase-Schütz, L. Rosenbaum, H. Hertlein, C. Glaeser, F. Timm, W. Wiesbeck, and K. Dietmayer, "Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 3, pp. 1341–1360, 2020.

[2] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, and J. Stachniss, C.and Gall, "Semantickitti: A dataset for semantic scene understanding of lidar sequences," in *ICCV*, 2019.

[3] Y. Pan, B. Gao, J. Mei, S. Geng, C. Li, and H. Zhao, "Semanticposs: A point cloud dataset with large quantity of dynamic instances," in *IV*, 2020.

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

[5] H. Thomas, C. R. Qi, J. E. Deschaud, B. Marcotegui, and L. J Goulette, F.and Guibas, "Kpconv: Flexible and deformable convolution for point clouds," in *ICCV*, 2019.

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

[7] H. Tang, Z. Liu, S. Zhao, Y. Lin, J. Lin, H. Wang, and S. Han, "Searching efficient 3d architectures with sparse point-voxel convolution," in *ECCV*, 2020.

[8] H. Zhou, X. Zhu, X. Song, Y. Ma, Z. Wang, H. Li, and D. Lin, "Cylinder3d: An effective 3d framework for driving-scene lidar semantic segmentation," *arXiv:2008.01550*, 2020.

[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 *ICRA*, 2018.

[10] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "Rangenet++: Fast and accurate lidar semantic segmentation," in *IROS*, 2019.

[11] L. Fan, X. Xiong, F. Wang, N. Wang, and Z. Zhang, "Rangedet: In defense of range view for lidar-based 3d object detection," *arXiv:2103.10039*, 2021.

[12] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3d point clouds: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020.

[13] D. Kochanov, F. K. Nejadasl, and O. Booij, "Kprnet: Improving projection-based lidar semantic segmentation," *arXiv:2007.12668*, 2020.

[14] R. Razani, R. Cheng, E. Taghavi, and L. Bingbing, "Lite-hdseg: Lidar semantic segmentation using lite harmonic dense convolutions," in *ICRA*, 2021.

[15] Y. Zhao, L. Bai, and X. Huang, "Fidnet: Lidar point cloud semantic segmentation with fully interpolation decoding," in *IROS*, 2021.

[16] S. Qiu, S. Anwar, and N. Barnes, "Semantic segmentation for real point cloud scenes via bilateral augmentation and adaptive fusion," in *CVPR*, 2021.

[17] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *CVPR*, 2017.

[18] B. Graham, M. Engelcke, and L. Van D. M., "3d semantic segmentation with submanifold sparse convolutional networks," in *CVPR*, 2018.

[19] 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 *ICRA*, 2019.

[20] C. Xu, B. Wu, Z. Wang, W. Zhan, P. Vajda, K. Keutzer, and M. Tomizuka, "Squeezesegv3: Spatially-adaptive convolution for efficient point-cloud segmentation," in *ECCV*, 2020.

[21] X Ding, X Zhang, N Ma, J Han, G Ding, and J Sun, "Repvgg: Making vgg-style convnets great again," in *CVPR*, 2021.

[22] T. Cortinhal, G. Tzelepis, and E. E. Aksoy, "Salsanext: Fast semantic segmentation of lidar point clouds for autonomous driving," *arXiv:2003.03653*, 2020.

[23] Y. A. Alnaggar, M. Afifi, K. Amer, and M. ElHelw, "Multi projection fusion for real-time semantic segmentation of 3d lidar point clouds," in *WACV*, 2021.

[24] S. Li, X. Chen, Y. Liu, D. Dai, C. Stachniss, and J. Gall, "Multi-scale interaction for real-time lidar data segmentation on an embedded platform," *IEEE Robotics Autom. Lett.*, 2021.