LIGHTNINGNET : FAST AND ACCURATE SEMANTIC SEGMENTATION FOR AUTONOMOUS DRIVING BASED ON 3D LIDAR POINT CLOUD

Kaihong Yang, Sheng Bi, Min Dong*

South China University of Technology, China

ABSTRACT

Semantic segmentation is A series of earlier works have demonstrated that 3D LiDAR point cloud segmentation is a promising approach. However, these methods usually rely on pretrained models. And their requiring improvements in either speed or accuracy prevent them to be applied to mobile platforms. To address these problems, a smaller Convolutional Neural Network (CNN) architecture, namely, LightningNet is presented. Furthermore, we propose a lightweight pipeline with Feature Refinement Modules to boost the performance of real-time 3D LiDAR point cloud segmentation. Our model is trained on spherical images projected from LiDAR point clouds on KITTI [1] dataset. Experiments show accuracy improvements of 4.1-8.2% in small objects and 4% in terms of mIoU over the state-of-the-art of spherical-image based method with 111 fps on a single 1080Ti GPU and 14fps on a Jetson TX2, which enables deployment for real-time semantic segmentation in autonomous driving. We achieve superior performance on another large dataset called SemanticKITTI (illustrated in Figure 1) both in speed and accuracy also.

***Index Terms*—** LiDAR Point clouds, Light-weight Network, Semantic Segmentation, Autonomous Driving

1. INTRODUCTION

Interest in autonomous driving systems has motivated many computer vision studies over the past few years. The safety of autonomous driving systems depends on accurate, real-time and robust perception of the environment. Thus, different sensor combinations are used for different applications to help systems perceive their surroundings. And the 3D LiDAR sensor is one of the most popular components of such sensors. Unlike cameras, 3D LiDAR is robust regardless of lighting conditions, both during the day and at night and both with or without glare and shadows [2]. As the price of 3D LiDAR sensors drops, an increasing number of studies and industrial applications have been focused on LiDAR based perception tasks, such as semantic segmentation.

There exists much work dealt with 3D point clouds using heavy pipelines; however, such methods require many parameters and are difficult to tune and we focus on deep learning approaches here. Deep learning work on 3D LiDAR segmentation processes data in the following ways: as a raw point cloud [4], a voxelized 3D space [5] and a spherical image [2].

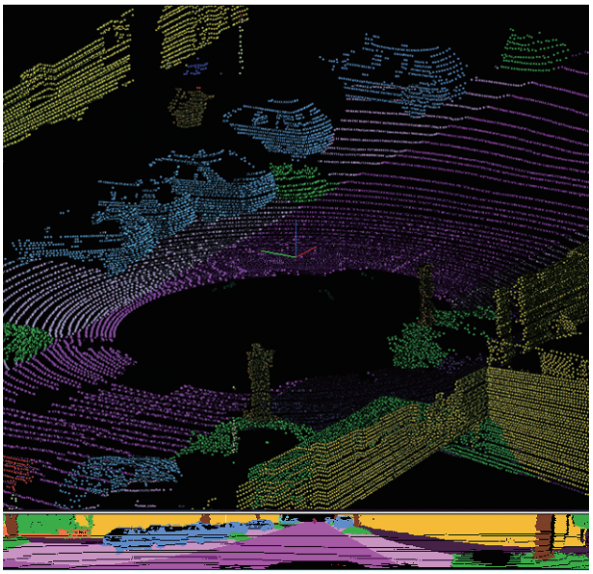

Fig. 1. Colored laser scan and corresponding spherical image from SemanticKITTI [3] with semantic information.

However, state-of-the-art (SOTA) segmentation methods require workstations that have high computational abilities. Embedded computing devices such as the Jetson TX2 and FPGAs cannot meet this requirement. Therefore, finding a robust, accurate and real-time perception method suitable for these devices has become a crucial problem.

Recently, some progress has been made in light-weight image classification [6, 7, 8]. These works have facilitated real-time semantic segmentation on GPUs and mobile devices. However, there are some differences between these two tasks; for example, to obtain localization information, semantic segmentation requires larger receptive fields and low-scale features, which are less important to image classification. Thus, we need to take the gap between these tasks into consideration when applying the above mentioned frameworks to semantic segmentation.

Since SqueezeSeg [2], derived from the light-weight net-

*Corresponding author: Sheng Bi (Email:picy@scut.edu.cn)

978-1-7281-1331-9/20/$31.00 2020 IEEE

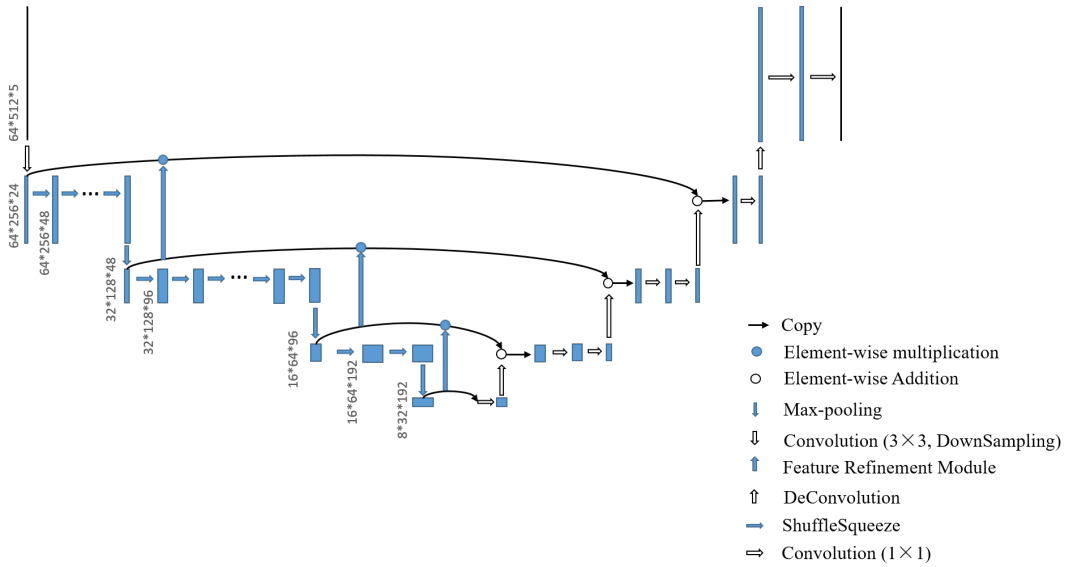

Fig. 2. Illustration of our pipeline based on LightningNet, for 3D LiDAR point cloud semantic segmentation. The baseline does not contain the Feature Refinement Module. The first ShuffleSqueeze in each Stage is for doubling the feature map's channels. Ellipses indicate that the same steps are omitted.

work SqueezeNet [6], was proposed to find the optimal trade-off between accuracy and computational efficiency, a series of its network extensions have been reported. In these methods, sparse, irregular 3D point cloud data are projected into dense, regular 2D spherical images and are segmented by Encoder-Decoder convolutional network architectures. These methods have succeeded in improving the speed of processing data, enabling these methods to run accurately on embedded devices, through the introduction of techniques such as Conditional Random Field (CRF) [9] and SENet [10]. It has been experimentally shown that these methods achieve fast and stable runtime (8.7 ms per frame without CRF and 13.6 ms per frame with CRF in SqueezeSeg). However, there is still room for improvement in finding the optimal trade-off.

One of challenges carrying out segmentation on the transformed data is that height of the data is at least 8 times less than width. SqueezeSeg sets the stride to (1,2) to not downsample the height to keep small objects' information, which increases the network's FLOPs a lot. In this work, we propose an end-to-end, train-from-scratch, light-weight pipeline based on our CNN architecture LightningNet. It can make better use of point cloud's information even if downsampling height and width at the same time and thus our pipeline based on it has fewer parameters and lower memory requirement. Then, we introduce connections with spatial attention to refine the low-level features while fusing features between the Encoder and the Decoder to boost the performance furthermore.

With these improvements, we propose a faster (111 fps), better (our method outperforms the most recent work by 4% in terms of mean intersection over union (mIoU)), lighter (the parameters of the network is 0.785M and the FLOPs is 1.216G), and train-from-scratch network.

Our contributions are summarized as follows:

We propose a light-weight convolution architecture to decrease FLOPs and parameters of our pipeline. We propose an efficient new baseline, based on our new convolution architecture, which is the fastest to date in segmenting the LiDAR point clouds from the KITTI dataset [1]. We propose a Feature Refinement Module (FRM) to increase the baseline's robustness and performance. With all of the proposed improvements, our method achieves SOTA performance in efficiency and accuracy.

2. OUR APPROACH

2.1. An Effective Baseline

Here we introduce a simple yet effective and fast pipeline based on the LightningNet, shown in Figure2 and Figure 3. We adopt the U-Net design, which includes an Encoder and a Decoder, to design our basic segmentation framework. Since the point clouds are collected from a Velodyne HDL-64E LiDAR, they have only 64 vertical channels. Therefore, height of spherical images is set to 64. The SqueezeSeg's dataset is filtered to only consider the front view area within 90 degrees and is divided into 512 grids which is also the width. With such a large gap between width and height dimensions, downsampling both of them at the same time without a special strategy will severely degrade small objects' information in point cloud. As stated in [11], feature maps with higher resolution contain more spatial information which helps to

segment small object. Thus we delay downsampling to the end of each stage. Although this would also bring extra cost in memory, our lightweight CNN architecture LightningNet alleviates the problem. The Decoder is derived from U-Net. For feature fusion, we do not concatenate upsampled feature maps with the corresponding feature maps in the Encoder but add them element-wisely. After each feature fusion, a 1×1 convolutional layer is used to help obtain more global information, and another 1×1 convolutional layer is used to gradually decrease the features to the desired number of possible classes.

2.2. LightningNet

With delayed downsampling and higher resolution, we are looking for a lighter CNN architecture to do the convolution.

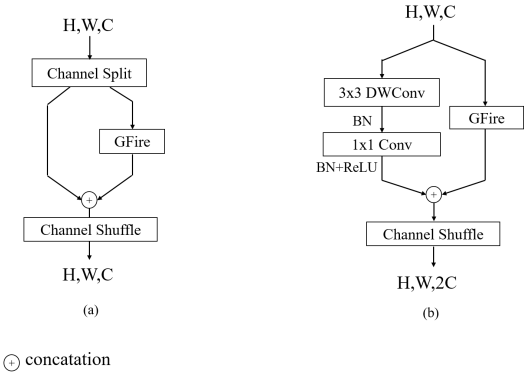

Fig. 3. LightningNet: (a) our basic LightningNet; (b) our LightningNet for feature channel expansion (2X). **DWConv**: depthwise convolution; **BN**: batch normalization; **GFire**: illustrated in Figure 4.

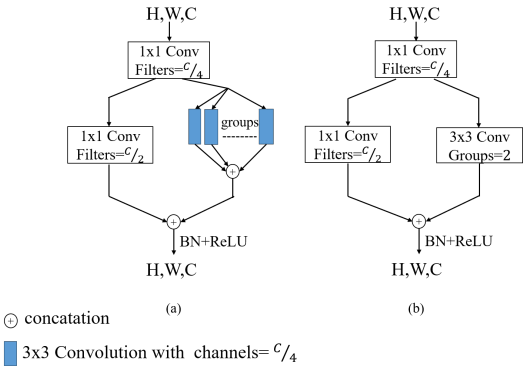

Fig. 4. GFire: (a) GFire in LightningNet; (b) a block equivalent to (a). **BN**: batch normalization.

Existing module like ShuffleNet V2 is not sufficiently light in weight. The right branch (1×1conv-3×3DWconv-1×1conv) of its basic unit requires $2C^2+9C$ parameters, and the downsampling unit needs $3C^2+9C$. When we delay the downsampling until each Stage is executed, parameters and computations increase due to higher resolution. Therefore, we propose LightningNet. We add the channel shuffle operation to our design and use the main framework of ShuffleNet V2 [8]. However, to delay the downsampling, we do not make the first block in each Stage a downsampling block but only a feature channel expansion block. Therefore, stride is set 1 and channels are double via concatenation of two branches. After each Stage, max pooling is used to downsample and to obtain a larger receptive field. As mentioned above, the original design of the ShuffleNet V2 unit strongly depends on depthwise separable convolution, and most of the computational cost is due to pointwise convolution which takes up $2C^2$ of $2C^2+9C$. Here we introduce our GFire module, which is inspired from SqueezeNet, to replace the right branch. As illustrated in Figure 4, we reduce the input channels to a quarter using a 1×1 convolution. After that we use a 1×1 conv layer to replace part of the 3×3 convolution and separate the 3×3 conv into 2 groups to further more reduce the parameters like Figure 4(a). And output of 1x1 conv is concatenated with output of the group convolution. Given matching input and output dimensions H$\times$W$\times$C, a normal 3×3 convolutional layer requires $9C^2$ parameters and $9HWC^2$ FLOPs, the right branch of a basic Shuffle Unit requires $2C^2+9C$ and $2HWC^2+9HWC$ respectively, while the GFire module requires only $\frac{3g+9}{8g}C^2$ parameters and $\frac{3g+9}{8g}HWC^2$ FLOPs, where g denotes the number of groups for 3×3 convolutions. Group convolution does not yet perform well on some embedded devices; therefore, to be cautious, we use only two groups. The left branch remains unchanged. The LightningNet is illustrated in Figure 3.

2.3. Feature Refinement Module

Feature fusion is crucial in semantic segmentation. Usually, high-level features contain more semantic information but have much lower resolution, meaning that they contain less spatial information. This lack of resolution limits the Decoder in reconstructing a precise segmentation map. U-Net was proposed to address this problem by adding links between low-level but high-resolution features from the bottom layers and high-level, low-resolution features from the top layers. However, there is a gap between these two types of features. Such direct fusion cannot help high-level features make full use of low-level features' spatial details. This is because low-level features contain little semantic information. Hence, we are inspired by SENet[10] and SqueezeSegV2[12] to refine the connections between the Encoder and Decoder as follows:

$$y_i = d(y_{i+1}) + F(x_i, x_{i+1}) \tag{1}$$

$$F(x_i, x_{i+1}) = U(A(c^{3,C}(x_{i+1}))) \times x_i \tag{2}$$

$$X_{i+1} = c^{3,C}(x_{i+1}) \tag{3}$$

$$\begin{aligned} A(X_{i+1}) = X_{i+1} \times \sigma(c^{1,C}(c^{1,\frac{C}{16}}(\\ [AP^7(X_{i+1}), MP^7(X_{i+1})]))) \end{aligned} \tag{4}$$

where y_i and x_i are the i-th level features in the Decoder and Encoder, respectively; d denotes deconvolution; $\times$ denotes element-wise multiplication; $c^{3,C}$ denotes convolution with a filter size of 3×3 and a number of output filters equal to C; and U denotes upsampling. Here we use Sub-pixel convolution from a previously proposed Super-Resolution technique [13]. F denotes the Feature Refinement Module (FRM); A denotes a Spatial Attention Module (SAM). As shown in Figure 5, the SAM starts with two parallel max pooling and average pooling layers with a large kernel of 7×7: $MP^7 \in \mathbb{R}^{C\times H\times W}$ and $AP^7 \in \mathbb{R}^{C\times H\times W}$. These are used to focus on the spatially informative part of the data. To seek a balance between efficiency and accuracy, we only consider a spatial attention module without a channel attention module. With the knowledge from high-level features of which areas are more important, the low-level features can be semantically refined better.

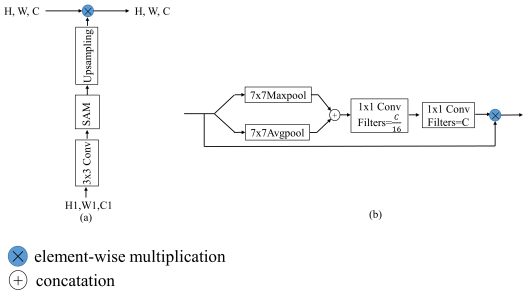

Fig. 5. Feature refinement module (FRM): (a) our FRM; (b) spatial attention module (**SAM**) in (a).

3. EXPERIMENTS

3.1. Training Details

All experiments are performed on an Nvidia Jetson TX2 and a workstation with a single 1080ti GPU in a CUDA 9 and cnDNN v7 environment. The model is developed in Tensorflow and Pytorch. We use the same hyperparameters provided in papers we compared. For the converted KITTI dataset, we use cross-entropy as the loss function and set learning rate to 0.005. We set batch size to 32 for training the whole network over 60000 steps in approximately four hours. And for the SemanticKITTI [3] dataset, we use a learning rate of 0.001 with a decay of 0.99 every epoch and train the same network for 150 epochs.

3.2. Dataset and Evaluation Metrics

We evaluate our model on a published dataset, which we call KITTI below, containing 10,848 images with point-wise labels from SqueezeSeg. The initial data are Velodyne data organized in time sequences from KITTI raw dataset. Similar to SqueezeSeg, these images are split into two parts: the first 8,057 frames in sequence are used for training and the last 2,791 are used for validation. We also test our method on another new largescale dataset – SemanticKITTI that provides point-wise semantic labels for the entire KITTI Odometry Benchmark. The dataset is comprised of over 43000 scans and is split into three set for training, validating and testing respectively.

We use the same evaluation metric, intersection-over-union (IoU), as in SqueezeSeg. The IoU is defined as follows:

$$IoUc = \frac{|\rho_c \cap \varrho_c|}{|\rho_c \cup \varrho_c|} \tag{5}$$

where ρ_c and ϱ_c denote the predicted and ground-truth point sets, respectively, belonging to class-c and $|\cdot|$ denotes the cardinality of a set. As shown in Equation (5), the IoU is the ratio of the overlap area between the predicted and ground-truth sets to the area of their union.

3.3. Ablation Studies

We improve the baseline in several respects. The mIoU and runtime performance comparisons on KITTI between similar works and our proposed LightningNet are shown in Table 1 and performance comparisons on SemanticKITTI with more point cloud segmentation works are listed in Table 2. We run SqueezeSegV2 ourselves to get its runtime on our workstation while the other methods' performance are taken from their corresponding papers. We perform ablation studies on the effect of delay downsampling and our other improvements on KITTI and show them in Table 3 and 4. The comparisons of runtime performance on TX2 are shown in Figure 7. Notice that the original SqueezeSeg and SqueezeSegV2 papers do not offer their runtime performance on TX2. Therefore, we run them on an NVIDIA Jetson TX2 ourselves to get their performance.

As shown in Table 1 and 2, our baseline achieves an mIoU performance that is comparable to that of the SOTA network and surpasses it in speed. We apply a scale factor on the number of channels to customize the network for different complexity. For example, the KITTI benchmark only has 3 categories to segment while the SemanticKITTI has 19 to work. We denote the network for the KITTI "LightningNet 0.5X"

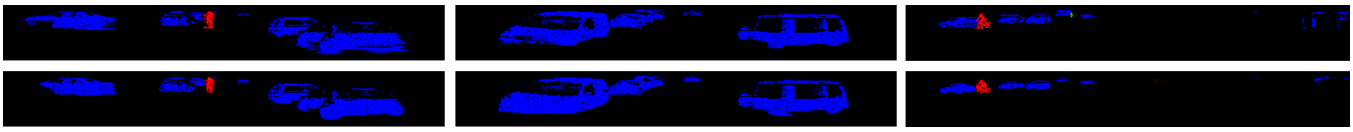

Fig. 6. Comparison of ground-truths (top) and predictions (bottom) on KITTI. Blue: car; red: cyclist; yellow: pedestrian.

Table 2. IoU (%) and runtime performance (ms) on SemanticKITTI. The input size of PointNets is 50000 points and the input size of the others is 64*2048 px.

Method	parameters	FLOPs	car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign	mIoU	Time(ms)
PointNet [14]	-	-	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	14.6	500
PointNet++ [4]	-	-	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	20.1	10000
SqueezeSegV2-CRF [12]	0.928M	3.433G	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	39.6	25
RangeNet53 [15]	50.377M	94.411G	86.4	24.5	32.7	25.5	22.6	36.2	33.6	4.7	91.8	64.8	74.6	27.9	74.1	55	78.3	50.1	64	38.9	52.2	**49.9**	77
LightningNet 0.5×(Ours)	**0.775M**	**1.25G**	85.3	18.9	24.5	30.7	23	29.2	38.7	0	90.4	28.5	74.9	1.1	77.7	42.1	77.1	35.2	72.2	25.3	16.4	41.6	**21**
LightningNet 1×(Ours)	4.763M	4.797G	86.3	22	29.4	39.6	25.7	37.8	44.3	0	92	33.3	77.5	0.6	79.9	39.6	77.7	38.1	70.7	32.5	24.3	44.8	22

Table 1. Comparisons with similar works on KITTI (IoU, %; Runtime, ms).

Method	Car	Pedestrian	Cyclist	mIoU	Time (ms)
SqueezeSeg	60.9	22.8	26.4	36.7	8.7
SqueezeSeg w/CRF	64.6	21.8	25.1	37.2	13.5
PointSeg	67.4	19.2	32.7	39.7	12
PointSeg w/RANSAC	67.3	23.9	38.7	43.3	14
SqueezeSegV2	73.2	27.8	33.6	44.9	12
LightningNet 0.5× (vanilla)	71.2	31.3	34.9	45.8	**8**
LightningNet 0.5×	73	36	37.7	**48.9**	**9**

Table 3. Ablation study for delayed downsamplings on KITTI(IoU, %; Runtime, ms).

	Car	Pedestrian	Cyclist	mIoU	Time (ms)
ShuffleNet V2	62.2	13.1	18.1	31.1	6
Early Downsampling	68.5	23.7	33.3	41.8	8
Delayed Downsampling	69.9	29.8	31.6	43.8	8.3

and that for the more complicated SemanticKITTI "LightningNet 1×". As we can see in Table 2, the overall complexity of LightningNet 1× is roughly 2 times of LightningNet 0.5×. Although our method does not beat RangeNet53 on SemanticKITTI, but our performance of parameters, FLOPs as well as inference speed outrun it a lot, which makes our method more suitable for embedding devices. In Figure 7, it is illustrated that our method improves the speed on embedded device, TX2, significantly and is very compatible with robot applications. Comparison of ground truth labels and segmentation results by LightningNet can be found in Figure 6.

Ablation study for delayed downsamplings. This experiment is designed to show the advantage of LightningNet comparing with ShuffleNet V2 and to show the influence of delayed downsampling in our pipeline without the FRM module. In this experiment, stride is (2,2) for the three stages. Table 3 shows that LightningNet is more powerful to obtain information than ShuffleNet V2 by having better performance in all classes. By comparing stats between Early Downsampling and Delayed Downsampling, we can observe an increase of performance when we delay downsampling. This supports our statement that LightningNet is lighter and more powerful and delayed downsampling helps our pipeline to gain better performance.

Table 4. Ablation study for FRM on KITTI(IoU, %; Runtime, ms).

	Car	Pedestrian	Cyclist	mIoU	Time (ms)
+Module in ExFuse [11]	70.5	32.9	33.8	45.7	8.7
+FRM(w/o SAM)	72.6	33	33	46.2	8.7
+FRM(w SAM)	73	36	37.7	48.9	9

Ablation study for FRM. From the results of Table 4, we observe that feature refinement is necessary for restoring semantic information in low-scale features. Additionally, a SAM can help to produce better segmentation results. Together, the FRM significantly improves the performance of all the classes by introducing more useful information to high-scale features when feature fusion. As illustrated in Table 4, our FRM works better than the similar module proposed in ExFuse. Meanwhile, the runtime slightly increases by 1 ms, which is still acceptable.

Finally, we compare our results with those obtained using SqueezeSegV2, as summarized in Table 1. Our results are comparable to those obtained with the SOTA method, and our method outperforms the SOTA method for mIoU, pedestrians and cyclists on KITTI. Our method's inference time is 9 ms per frame on our workstation and 71 ms per frame on the

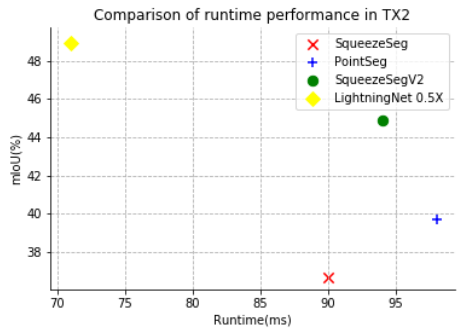

Fig. 7. Comparsion of runtime performance on TX2.

Jetson TX2 embedded device, with 0.78M parameters and a memory cost of 1.25GB, while the former methods require double of it or way more.

4. CONCLUSION

In this paper, we propose a new pipeline to do segmentation on LiDAR point cloud, which is more accurate and faster than SqueezeSegV2. We design a novel lightweight CNN architecture LightningNet which enables the proposed pipeline to handle feature maps with higher resolution. LightningNet is much lighter in weight and better at obtaining information in feature maps. The baseline based on this design achieves an improvement of 0.9% in terms of mIoU compared to the SOTA network on KITTI, with an extremely fast runtime of 8 ms per frame (125 fps). We also design a Spatial Attention Module and connections to help refine the low-level features. With all of these improvements and corresponding ablation experiments, we find an optimal trade-off between speed and accuracy. Finally, we achieve accuracy improvements of 4.1% and 8.2% on small object categories and 4% on average, while the speed of 14 fps achieved on the Jetson TX2 proves that our network can be compatible with embedded platforms. We also test our method on a new dataset SemanticKITTI and obtain the similar performance to KITTI.

5. ACKNOWLEDGMENTS

This research work is supported by National Natural Science Foundation of China (61703168, 61703284). Guangdong province science and technology plan projects (2017A030310163, 2017A010101031). Guangdong science and technology innovation strategy project (2018B010108002).

6. REFERENCES

[1] Andreas Geiger, Philip Lenz, and Raquel Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *CVPR*, 2012.

[2] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer, "Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud," in *ICRA*, 2018.

[3] 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 *ICCV*, 2019.

[4] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in neural information processing systems*, 2017.

[5] Gregory P Meyer, Jake Charland, Darshan Hegde, Ankit Laddha, and Carlos Vallespi-Gonzalez, "Sensor fusion for joint 3d object detection and semantic segmentation," in *CVPR*, 2019.

[6] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[7] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *CVPR*, 2018.

[8] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," in *ECCV*, 2018.

[9] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr, "Conditional random fields as recurrent neural networks," in *ICCV*, 2015.

[10] Jie Hu, Li Shen, and Gang Sun, "Squeeze-and-excitation networks," in *CVPR*, 2018.

[11] Zhenli Zhang, Xiangyu Zhang, Chao Peng, Xiangyang Xue, and Jian Sun, "Exfuse: Enhancing feature fusion for semantic segmentation," in *ECCV*, 2018.

[12] Bichen Wu, Xuanyu Zhou, Sicheng Zhao, Xiangyu Yue, and Kurt Keutzer, "Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud," *arXiv preprint arXiv:1809.08495*, 2018.

[13] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *CVPR*, 2016.

[14] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *CVPR*, 2017.

[15] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "RangeNet++: Fast and Accurate LiDAR Semantic Segmentation," in *IROS*, 2019.