2018 IEEE Intelligent Vehicles Symposium (IV)
Changshu, Suzhou, China, June 26-30, 2018

LiSeg: Lightweight Road-object Semantic Segmentation In 3D LiDAR Scans For Autonomous Driving

Wenquan Zhang, Chancheng Zhou, Junjie Yang and Kai Huang*
Key Laboratory of Machine Intelligence and Advanced Computing
(Sun Yat-sen University), Ministry of Education, China
School of Data and Computer Science, Sun Yat-sen University, China
Email: {zhangwq35, zhouchch6, yangjj27}@mail2.sysu.edu.cn, huangk36@mail.sysu.edu.cn

***Abstract*— LiDAR based perception module plays an important role in autonomous driving. However, the present CNN models are designed for image processing but not LiDAR point clouds. The performances of such models are limited by the great memory consumption and heavy computation cost. In this work, a lightweight CNN model, LiSeg, is proposed to perform real-time road-object semantic segmentation on LiDAR point cloud scans for autonomous driving. The model size of LiSeg is several times smaller than others, while achieving high accuracy.**

I. INTRODUCTION

There is increasing popularity on Light Detection And Ranging(LiDAR) sensors equipped on autonomous driving vehicles. LiDAR is able to capture accurate 3D information of the environment as point clouds. LiDAR point clouds are used for object localization, object detection, and scene understanding in autonomous driving. For scene understanding on LiDAR point clouds, a semantic label is attached to every points in the 3D space, much like semantic segmentation in image processing (a.k.a. pixel-wise classification). These semantic labels are useful for follow-up fine-grained tasks, for instance finding accurate object and road boundaries for localization and navigation, which helps autonomous vehicles to drive safely.

Traditional LiDAR point cloud based semantic segmentation works in a pipelined fashion [1]. It consists of multiple stages, i.e., ground segmentation, object segmentation, and classification based on hand crafted features [2]. In such a multi-stage pipelined process, the accuracy of the final results heavily depends on the results from the segmentation stage. In addition, the accuracy of classification also depends on the hand crafted features.

To cope with the problems of using multi-stage processing and designing hand crafted features, convolutional neural network(CNN) has been proposed as an end-to-end solution for LiDAR point cloud feature extraction and classification. As the LiDAR point cloud is sparse due to the non-uniform sampling of the 3D space, there are two basic approaches for processing the sparse LiDAR point cloud in CNN: 1) 3D CNN is applied by encoding point cloud in 3D voxels [3], [4], [5], and 2) 2D image-style CNN is applied with point cloud projected into dense matrix [6], [7], [8]. For 3D CNN the accuracy is limited by the low resolution of voxelization, while high resolution voxels lead to huge amount of memory consumption. For those sparse and unordered points in LiDAR scans, 2D dense matrix is more memory-efficient at point-wise resolution compared to 3D voxels. However, in order to extract features within images, such 2D CNN model is designed with many convolution layers and convolution filters. It is too heavy for LiDAR point cloud processing in terms of model size and model complexity.

The huge model size and the consequent huge model complexity will introduce huge amount of runtime memory consumption as well as heavy computation. Although the model size can be reduced by weights quantization, deep compression, and pruning [9], [10], the accuracy however drops due to the low precision weights. In addition, the above techniques are not specifically targeting the reduction of the model complexity and the problem of complex models remains due to many convolution layers and many convolution filters.

In this paper, a lightweight FCN, namely LiSeg, is proposed to perform real-time road-object semantic segmentation on LiDAR point cloud. Our LiSeg is designed with fewer convolution filters. To further reduce convolution parameters, depth-wise separable convolution is applied. To reduce convolution operations, max-pooling is applied in early stage to reduce feature maps. In addition, dilated convolution is used in order to enhance the receptive field with less numbers of parameters. The model size of LiSeg is at least eight times smaller than the related work in the literature, while still achieving high accuracy. In the meanwhile, the proposed lightweight LiSeg network still achieves mean-IoU of 0.72 and 0.71 on KITTI data set and our Velodyne HDL-32E data set, respectively. The LiSeg performs point cloud semantic segmentation at 111 frames per second(fps) on a GTX 1070 GPU. The contributions of this paper are summarized as follows:

- A lightweight, fast and accurate FCN LiSeg for LiDAR point cloud semantic segmentation based on depth-wise separable convolution.
- A full-view point-wise annotated point cloud data set with high flow-rate pedestrians and vehicles captured by Velodyne 3D LiDAR HDL-32E.
- We implement our LiSeg with Tensorflow and compare

* Corresponding author

the performance with FASTNET [6] and SQUEEZESEG [8] in the literature.

The rest of this paper is organized as follows: Sec. II presents the related work. The data pre-processing step for point cloud encoding and the LiSeg network structure is proposed in Sec. III. The experiments and results are shown in Sec. IV. And Sec. V concludes the whole work.

II. RELATED WORK

The development of 3D LIDAR sensors has motivated researchers to propose efficient methods to point cloud encoding and segmentation in perception tasks.

A. Point Cloud Encoding

For data representations, Li et al. [11] and Chen et al. [12] project point cloud into perspective view as bird-eye view, and cylindrical coordinates front view. However, such perspective views are too sparse for segmentation. Maturana et al. [3] and Li et al. [4] proposed 3D CNN on point cloud voxel grids for classification and vehicle detection. Hackel et al. [5] proposed 3D CNN on point cloud voxels for semantic segmentation. Such voxel-based representation is memory-efficient for large organized point cloud. But it suffers from resolution due to data sparsity, and computation cost of 3D convolution, especially for LIDAR with less scan lines. Zhou et al. [13] proposed stacked Voxel Feature Encoding(VFE) layers to encode point cloud in 3D voxels, which encodes point cloud into voxels by deep learning. It balances each voxel by randomly picking and dropping points according to the sampling size, which still suffers from the problem of data sparsity within LIDAR point cloud. Dewan et al. [6], Velas et al. [7], and Wu et al. [8] encoded point cloud as dense matrix, which is more memory and time efficient for semantic segmentation on sparse LIDAR scan.

B. Segmentation

Dub et al. [14] proposed to detect objects using euclidean clustering and random-forest for classification after ground plane segmentation, based on about 21 hand-crafted features. Qi et al. [15] proposed point-set based semantic segmentation model for RGB point cloud, which is mainly generated by stereo cameras. Hackel et al. [5] proposed voxel based 3D CNN for semantic segmentation on large point cloud area, with 3D sliding windows. Such work is designed for large, dense, and organized point cloud, but not for the sparse unordered LIDAR scan in real-time scenarios. Velas et al. [7] proposed using CNN for point cloud ground segmentation. Dewan et al. [6] proposed using image style CNN structure FastNet for deep semantic classification for 3D LIDAR data. The two 1024 units fully-connected layers in FASTNET lead to the great amount of model parameters, and slow down the inference step. Wu et al. [8] proposed using image style CNN SqueezeSeg with Conditional Random Field(CRF) module. Though CRF module helps to improve the segmentation performances, especially on image semantic segmentation tasks, the CRF module is time-consuming for it needs several iterations. What's more, such schemes, which are designed for image-based feature extraction, are too complex for the sparse LIDAR point cloud, leading to long training time and inference time. In this work, a lightweight network with separable convolution is proposed.

III. APPROACH

A. Data Pre-processing

The point cloud in LIDAR scans is sparse, and point density varies a lot due to the distance and shape of objects. In order to process the sparse and unorganized LIDAR point cloud with deep convolution neural network, the point cloud is encoded in dense grid-based matrix in 2D format, as shown in Fig.1. The point cloud is transformed into Spherical coordinates $\mathcal{S}(\theta, \phi)$, and discriminated into grid-based matrix $\mathcal{S}(\tilde{\theta}, \tilde{\phi})$. Other properties of points are encoded as channels of the dense matrix. The shape of the matrix is thus defined as $[H, W, C]$, which represents the number of grids on [θ, ϕ, $channels$] respectively. Here, H is the total number of LIDAR scan lines, e.g., $H = 64$ and 32 for Velodyne HDL-64E and Velodyne HDL-32E, respectively. W is the number of grids on ϕ. In order to project every point into Spherical coordinates, W should be set to $FOV/\Delta\phi$, while $\Delta\phi$ is the horizontal resolution of LIDAR. C is the number of required channels. In order to enhance memory efficiency and generalization of representation, C here is set to 5, with only 5 channels: intensity, x, y, z, and range($range = \sqrt{x^2 + y^2 + z^2}$). These channels are the raw properties of LIDAR point clouds, which are enough to present the distribution of points within 3D space.

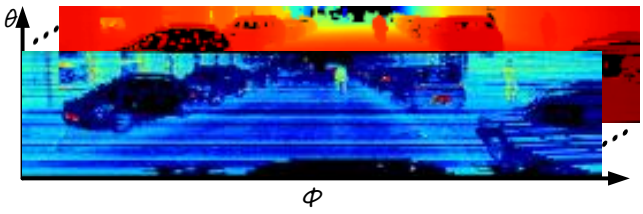


Fig. 1: Dense grid-based matrix in Spherical coordinates.

For each p_i in Cartesian coordinates point cloud $\mathcal{P}$, the corresponding transformation into Spherical coordinates $\mathcal{S}(\theta_i, \phi_i)$ and discrimination into grid-based Spherical coordinates $\mathcal{S}(\tilde{\theta}_i, \tilde{\phi}_i)$ is shown as Eq.1, Eq.2 and Eq.3. In this case, $\mathcal{RING}(p_i)$ is the scan line number of p_i, which can be achieved directly from the Velodyne driver.

$$\tilde{\theta}_i = \theta_i = \mathcal{RING}(p_i) \tag{1}$$

$$\phi_i = \arcsin \frac{p_i.y}{\sqrt{p_i.x^2 + p_i.y^2}} \tag{2}$$

$$\tilde{\phi}_i = \lfloor \frac{\phi_i}{\Delta\phi} \rfloor, \Delta\phi = 0.175^\circ \tag{3}$$

Here, the proposed LiSeg network is trained on KITTI data set, which used a Velodyne HDL-64E with 64 scan lines, 10Hz rotations frequency and 0.175° horizontal resolution($\Delta\phi$). In KITTI data set, only 90° of field of view

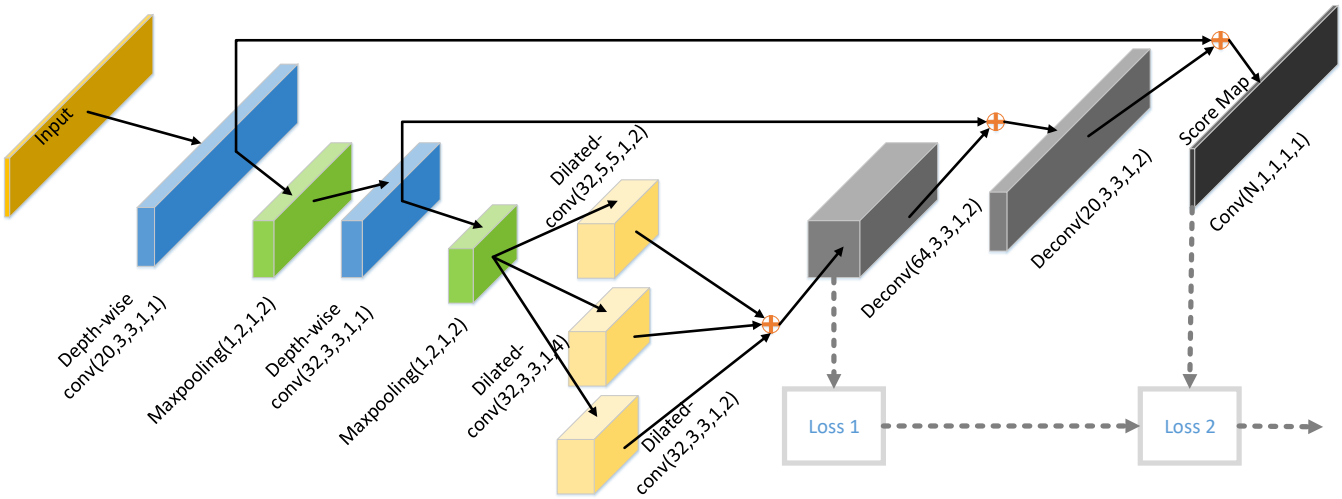


Fig. 2: LiSeg Structure. Depth-wise separable convolution layers are shown in blue. Max-pooling layers are in green. Dilated convolution layers are shown in yellow. Transpose convolution layers are shown in grey. Traditional convolution layer is shown in black.

is annotated. Thus, in this case, the H is set to 64 for KITTI data set and 32 for Velodyne HDL-32E data set, W is set to $90/0.175 = 512$, and C is set to 5.

B. Network Structure

In order to be deployed on mobile devices for autonomous driving, LiSeg is designed as a lightweight model according to the common encoder-decoder FCN style but with separable convolution and dilated convolution. The separable convolution performs a depth-wise convolution that acts separately on channels, followed by a point-wise convolution that mixes channel-wise features. In this case, depth-wise separable convolution is used for reducing parameters. In addition, due to the great differences of the distribution of intensity, x, y, z, and range, depth-wise convolution is more suitable. Dilated convolution is used in the encoder for enhancing the reception field. The whole LiSeg network encompasses only convolutional and deconvolutional layers followed by Batch Normalization(BN) and ReLu(RL) non-linearities.

The structure of LiSeg is shown in Fig.2. Depth-wise separable convolution layer is shown in blue. Max-pooling layer is shown in green. Dilated convolution layer is shown in yellow. Transpose convolution layer is shown in grey. Convolution is shown in black. The convolution operators are parameterized as $(F, K1, K2, S1, S2)$ and the max-pooling operators are parameterized as $(K1, K2, S1, S2)$, where F is the number of filters, $(K1, K2)$ is the kernel size respectively, and $(S1, S2)$ is the strides respectively except for dilated convolution layers. For dilated convolution layers, $(S1, S2)$ stands for the dilated rate.

As LiSeg is designed in lightweight manner, two depth-wise separable convolution layers are applied for reducing parameters and computation[16]. For the first depth-wise separable convolution layer with kernel size (3, 3), the number of input channels $\mathcal{C}_{in1}$ is 5, and number of output channels $\mathcal{C}_{out1}$ is 20. For the second depth-wise convolution layer with kernel size (3, 3), $\mathcal{C}_{in2}$ is 20 and $\mathcal{C}_{out2}$ is 32. After applying depth-wise separable convolution, the number of parameters in the first layer is reduced from 900($\mathcal{C}_{in1} * \mathcal{C}_{out1} * 3 * 3$) to 145($\mathcal{C}_{in1} * 3 * 3 + \mathcal{C}_{in1} * \mathcal{C}_{out1} * 1 * 1$) by 84%. The number of parameters in the second layer is reduced from 5760 to 820 by 86% in the same manner.

In order to reduce parameters of feature maps, and improve timing performance, two max pooling layers are used in early stage for sampling the feature maps by a half. However, due to the low resolution of the dense matrix and the imbalance of H and W, e.g., (H=64, W=512) and (H=32, W=512), sub-sampling is only carried out on columns W within the encoder module.

Three dilated convolution layers with dilated rate: (1, 2), (1, 4) and (1, 2) are used for feature extraction. Such three layers provide multi-scale features with reception field: 3*7, 3*15, and 5*11. In this case, spatial drop-out layers are applied for the above three dilated convolution layers, while training with Velodyne HDL-32E data set. The multi-scale feature maps from the three dilated convolution layers are then concatenated as the input of decoder for transpose convolution, similar to Inception module but dilated convolution is used rather than convolution.

Skip-connection techniques are used in the decoding step by providing features in early stages as enhance-concatenate component. After two transpose convolution layers, the final score map is generated by a convolution layer.

The whole LiSeg network is trained via end-to-end back-propagation guided by the following weighted multi-layer loss function:

$$\mathcal{L} = \sum_{i=1}^{2} \lambda_i \mathcal{L}oss_i{}^{WCE}(\hat{\mathcal{Y}}, \mathcal{Y}) \tag{4}$$

The above weighted multi-layer loss $\mathcal{L}$ is evaluated on two feature maps respectively, regarded as $\mathcal{L}oss_1$ and $\mathcal{L}oss_2$, as

shown in Fig.2. The hyper parameter λ_i is the corresponding regularization weights. $\hat{\mathcal{Y}}$ and $\mathcal{Y}$ are the predictions and ground truth respectively.

The number of points in the LiDAR point cloud are class-imbalance, because most of points belong to background. Thus the following multi-class Weighted Cross Entropy(WCE) loss is applied as regularization, because of the class-imbalance problem of the LiDAR point cloud. The corresponding class-imbalance regularization weights are computed according to the training set statistics.

$$\mathcal{L}oss_i^{WCE}(\hat{\mathcal{Y}}, \mathcal{Y}) = - \sum^{H_i, W_i, Classes} \mathcal{W}(\mathcal{Y}) \mathcal{ID}(\mathcal{Y}) \log(\hat{\mathcal{Y}}) \tag{5}$$

The $\mathcal{ID}(\mathcal{Y})$ is an index function to select the expected classes, and $\mathcal{W}(\mathcal{Y})$ is corresponding the class-imbalance weights.

IV. Experiment

A. Data Set

1) KITTI: The HDL-64 version model LiSeg-64 is trained with the KITTI Velodyne object detection benchmark training set containing 7,481 Velodyne scans. The provided 3D bounding box are transformed into point-wise label, encoded in dense matrix format. In this case, only points belong to $cars$ and $pedestrians$ classes are taken into consideration for LiSeg-64, others are regarded as $don't care$.

2) HDL-32E Beijing: The HDL-32E version model LiSeg-32 is trained with our annotated HDL-32E Beijing point cloud data set. The data set is captured in Beijing, China, by the Velodyne HDL-32E LiDAR on the roof of the vehicle at 10 Hz. It contains 433 full-view annotated point cloud frames, and includes point-wise annotations for 5 classes: $vehicle$, $pedestrian$, $ground$, $curb$, and $tree$. The point clouds are captured under four different scenes at different time, including traffic jam in the morning rush hour, normal traffic condition in the noon and so on. It shows the high density vehicles and pedestrians in China, and will soon be published. In this case, only points belong to $vehicle$, $pedestrian$, and $ground$ classes are taken into consideration for LiSeg-32, others are regarded as $don't care$.

In order to avoid over-fitting and audit the generalization capacities of the proposed architecture, a 5-fold cross-validation step is applied during training among both two data sets. In this case, spatial drop-out layers are applied after the above three dilated convolution layers, while training with Velodyne HDL-32E data set.

B. Data Augmentation

As mentioned above, the KITTI Velodyne point cloud is annotated only on the front $[-45°, 45°]$ (90°) field of view(FOV). An on-the-fly data augmentation is applied by rotation and flipping horizontally, in order to preserve the geometry properties of the LiDAR information. During training step, each point cloud is rotated to FOV $[-135°, -45°]$, $[45°, 135°]$, and $[-135°, -225°]$ plus a random angle shift δ. As the point clouds are calibrated into camera coordinates, the rotation is carried along y-axis. The corresponding rotation angle ω is shown in Eq.6. In this way, taking the random angle shift into account, the data set size was increased by a factor more than 8.

$$\omega = [0, -\frac{\pi}{2} + \delta, \frac{\pi}{2} + \delta, \pi + \delta] \tag{6}$$

In the Velodyne HDL-32E data set, the Velodyne point cloud is annotated on full 360° FOV, including 4 classes. For each point cloud frame, the 360° point cloud is divided into 71 sub point clouds each with 90° FOV, e.g. $[-180°, -90°]$, $[-175°, -85°]$ and so on. The rotation and randomly flipping online augmentation technique mentioned above is also applied during training step. In this way, the data set size was increased by a factor more than 568, regardless some overlapping situations of the sub point clouds.

C. Setup

The LiSeg is implemented with Tensorflow R1.4 estimator API. The multi-layer loss weights λ_1 is set to 0.1 and λ_2 is set to 0.9, which leads to the best performance in this case. The network is trained with Adam optimizer with standard parameters and learning rate 0.001, with batch size of 8, epochs of 8, due to the memory limitation. The data augmentation technique mentioned above is applied according to the data set.

The training process and experiments are conducted on i7-6700 with a single Nvidia GTX 1070 GPU. As KITTI object detection benchmark is not designed for point cloud semantic segmentation, the performance on KITTI data set is evaluated on the transformed validation set. For HDL-32E Beijing data set, the performance is evaluated on the test set with point-wise annotations.

D. Result

The performance of LiSeg are compared with other similar semantic segmentation models on LiDAR point cloud scans: SqueezeSeg and FastNet. The above models are trained and evaluated on both data sets. Models trained on KITTI data set with three predicted classes are marked with 64, while models trained on Velodyne HDL-32E data set with four predicted classes are marked with 32.

1) Model Size: The model size of LiSeg and other models are summarized in Table.I. For a fair comparison on the model size, all the models here are implemented with the same Tensorflow r1.4 estimator API with dense matrix input in the same shape $[64, 512]$ or $[32, 512]$. All models are exported in the same way, including variables and saved model in protocol buffer text format, without applying any model compression techniques.

TABLE I: Exported Model Size

Model	Model size(MB)
LiSeg	**1.1**
SqueezeSeg w/o CRF	8.5
FastNet	88.5

In Table.I, LiSeg is the most lightweight model, compared with SQUEEZESEG w/o CRF and FASTNET, with the help of small feature maps and the application of depth-wise separable convolution. The model size of FASTNET is about 80x larger, because of the two fully-connected layers. And the model size of SQUEEZESEG w/o CRF is eight times larger, and it will be larger if equipped with CRF module. Table.I shows that LiSeg is a lightweight LIDAR point cloud semantic segmentation model, and it's able to be deployed on mobile devices due to the small model size.

2) Runtime performance: In order to evaluate the timing performance by using LiSeg for autonomous driving perception tasks, the runtime is regarded as the inference time of the whole point cloud scan. So the inference time, i.e., the computation time on GPU, of all $four$ FOV: $[-45°, 45°]$, $[-135°, -45°]$, $[45°, 135°]$, and $[-135°, -225°]$ is regarded as runtime here.

The runtime performance of LiSeg and other models are summarized in Fig.3. The input dense matrix of LiSeg-64, SQUEEZESEG-64 and FASTNET-64 is in shape $[64, 512]$, while others are in shape $[32, 512]$. Fig.3 shows that LiSeg is faster than other models in inference time. LiSeg-32 can inference the whole Velodyne HDL-32E LIDAR point cloud scan of all $four$ FOV at 111 fps on average on the experiment environment, which is faster than other models. And it shows that LiSeg-64 runs at 45 fps, which is twice faster than SQUEEZESEG-64 and about four times faster than FASTNET-64. In addition, the proposed LiSeg-64 also runs faster than SQUEEZESEG-32 w/o CRF and FASTNET-32, while performing the semantic segmentation on the whole LIDAR scan frame. This is due to the lightweight design of LiSeg.

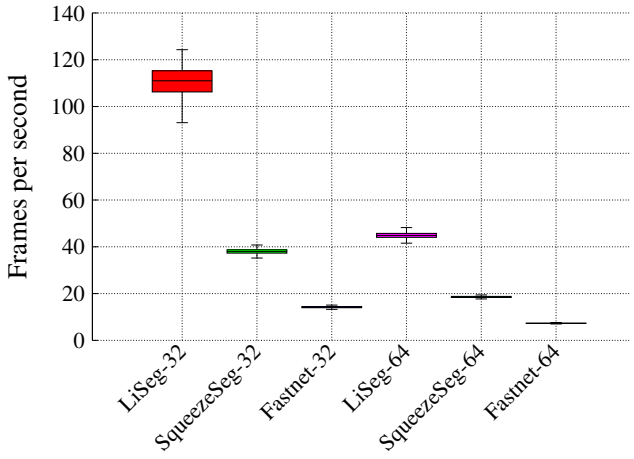


Fig. 3: Runtime performance in fps.

3) Segmentation Performance: The segmentation performance of LiSeg is evaluated in point-wise class labels. In this case, the standard mean Intersection-over-Union(mIoU), and the mean pixel accuracy(mPA) metrics are used to evaluate the segmentation performance. The performance is computed as following Eq.7 and Eq.8, while $\hat{\mathcal{Y}}_c$ is the predicted point set on class c and $\mathcal{Y}_c$ is the ground truth point set.

$$mIoU = \frac{1}{N} * \sum_{c}^{N} \frac{\left|\hat{\mathcal{Y}}_c \cap \mathcal{Y}_c\right|}{\left|\hat{\mathcal{Y}}_c \cup \mathcal{Y}_c\right|} \tag{7}$$

$$mPA = \frac{1}{N} * \sum_{c}^{N} \frac{\left|\hat{\mathcal{Y}}_c \cap \mathcal{Y}_c\right|}{\left|\hat{\mathcal{Y}}_c\right|} \tag{8}$$

For segmentation performance, LiSeg-64, SQUEEZESEG-64 w/o CRF, and FASTNET-64 are evaluated on KITTI data set, while LiSeg-32, SQUEEZESEG-32 w/o CRF, and FASTNET-32 are evaluated on Velodyne HDL-32E Beijing data set. The segmentation performance is summarized in Table.II.

TABLE II: Segmentation Performance

Model	mIoU	mPA
LiSeg-32	**0.71**	**0.76**
SQUEEZESEG-32 w/o CRF	0.70	0.73
FASTNET-32	0.68	0.69
LiSeg-64	0.72	0.78
SQUEEZESEG-64 w/o CRF	0.72	0.77
FASTNET-64	**0.75**	**0.82**

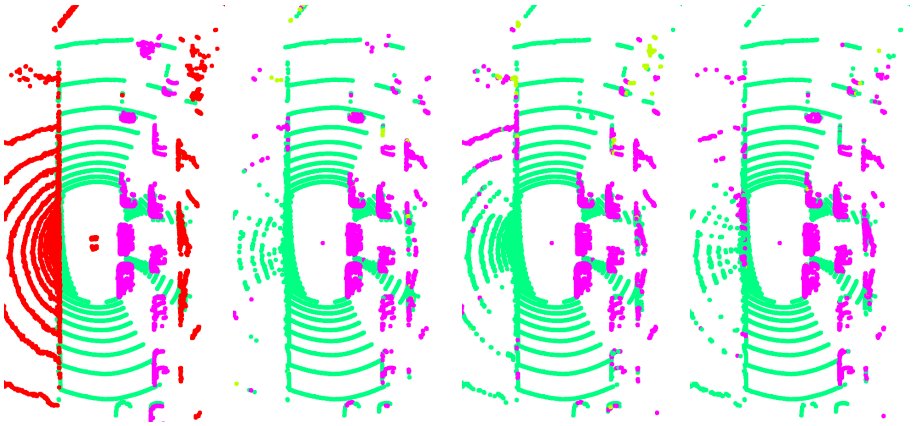
Table.II shows that the lightweight LiSeg performs well on road-object semantic segmentation in 3D LIDAR scans. Both evaluation metrics are the higher the better. It shows the the proposed LiSeg is not only lightweight and fast, but also accurate.

The qualitative results are shown in Fig.4. In this case, though FASTNET-32, and SQUEEZESEG-32 are able to predict the $ground$ in green and $vehicle$ in purple, LiSeg-32 is more accurate for fewer false positive points on $vehicle$ class, i.e., fewer purple points on trees and roads. And for LiSeg-64 and other two models, they are able to predict the point-wise labels in full-view for the partial annotated KITTI data set, due to the data augmentation technique. The $vehicles$ are shown in purple, and the $pedestrians$ are shown in green clearly. It shows that the proposed LiSeg is accurate for road-object semantic segmentation in LIDAR point cloud scans for autonomous driving perception tasks.

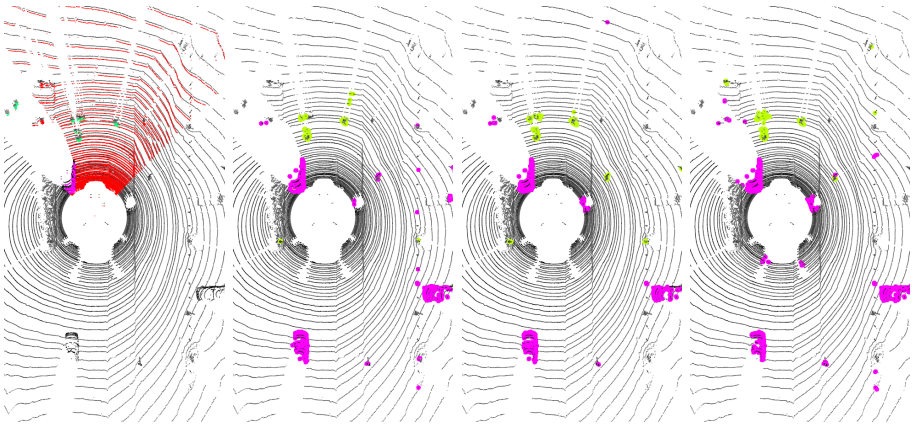
V. CONCLUSION

A lightweight but accurate FCN model LiSeg is proposed for road-object semantic segmentation in 3D LIDAR scans. The proposed model is training and evaluated at KITTI data set captured by Velodyne HDL-64E(LiSeg-64), and at our Velodyne HDL-32E Beijing data set[1] with point-wise annotation(LiSeg-32). The model size of LiSeg is eight times smaller than other models, while LiSeg still achieves high segmentation performance. LiSeg-64 achieves mIoU at 0.72 and runs at 45 fps on a single GTX 1070 GPU. LiSeg-32 achieves mIoU at 0.71, and runs at 111 fps for Velodyne HDL-32E point clouds.

[1]Can be found at https://drive.google.com/file/d/1LSy1UoHSLUFIlp8s3SFwY2JQvhdHTMuD/view?usp=sharing



(a) Results on Velodyne HDL-32E Beijing data set, from left to right: ground truth, LiSeg-32, SQUEEZESEG-32 w/o CRF, FASTNET-32. For Velodyne HDL-32E data set, $ground$ is shown in green, $vehicle$ is shown in purple, and $dontcare$ is shown in red.



(b) Results on KITTI data set, from left to right: ground truth, LiSeg-64, SQUEEZESEG-64 w/o CRF, FASTNET-64. Vehicles without annotation are also predicted correctly with the help of data augmentation. For KITTI data set, $vehicle$ is shown in purple, $pedestrian$ is shown in green, and $dontcare$ is shown in red. Points without annotation are shown in black.

Fig. 4: Full-view semantic segmentation results on test set after projected into Cartesian coordinates.

ACKNOWLEDGMENT

This work was supported in part by the National Key Research and Development Program of China under grant No.2017YFB1001703; the National Science Foundation of China under Grant No. U1711265; the Fundamental Research Funds for the Central Universities under grant No.17lgjc40; the Innovative R&D Team Introduction Program of Guangdong Province.

REFERENCES

[1] B. Douillard, J. Underwood, V. Vlaskine, A. Quadros, and S. Singh, "A pipeline for the segmentation and classification of 3d point clouds," in *Experimental robotics*. Springer, 2014, pp. 585–600.

[2] T. Hackel, J. D. Wegner, and K. Schindler, "Fast semantic segmentation of 3d point clouds with strongly varying density." *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, vol. 3, no. 3, 2016.

[3] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *Ieee/rsj International Conference on Intelligent Robots and Systems*, 2015, pp. 922–928.

[4] B. Li, "3d fully convolutional network for vehicle detection in point cloud," 2016.

[5] T. Hackel, N. Savinov, L. Ladicky, J. D. Wegner, K. Schindler, and M. Pollefeys, "Semantic3d.net: A new large-scale point cloud classification benchmark," vol. IV-1/W1, 2017.

[6] A. Dewan, G. L. Oliveira, and W. Burgard, "Deep semantic classification for 3d lidar data," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2017, pp. 3544–3549.

[7] M. Velas, M. Spanel, M. Hradis, and A. Herout, "Cnn for very fast ground segmentation in velodyne lidar data," 2017.

[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," 2017.

[9] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *ICLR*, 2016.

[10] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in neural information processing systems*, 2016, pp. 4107–4115.

[11] B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3d lidar using fully convolutional network," in *Robotics: Science and Systems*, 2016.

[12] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," 2016.

[13] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," 2017.

[14] R. Dub, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, and C. Cadena, "Segmatch: Segment based place recognition in 3d point clouds," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 5266–5272.

[15] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, vol. 1, no. 2, p. 4, 2017.

[16] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 1800–1807.