DeepTemporalSeg: Temporally Consistent Semantic Segmentation of 3D LiDAR Scans

Ayush Dewan Wolfram Burgard

***Abstract*— Understanding the semantic characteristics of the environment is a key enabler for autonomous robot operation. In this paper, we propose a deep convolutional neural network (DCNN) for the semantic segmentation of a LiDAR scan into the classes *car*, *pedestrian* or *bicyclist*. This architecture is based on dense blocks and efficiently utilizes depth separable convolutions to limit the number of parameters while still maintaining state-of-the-art performance. To make the predictions from the DCNN temporally consistent, we propose a Bayes filter based method. This method uses the predictions from the neural network to recursively estimate the current semantic state of a point in a scan. This recursive estimation uses the knowledge gained from previous scans, thereby making the predictions temporally consistent and robust towards isolated erroneous predictions. We compare the performance of our proposed architecture with other state-of-the-art neural network architectures and report substantial improvement. For the proposed Bayes filter approach, we show results on various sequences in the KITTI tracking benchmark.**

I. INTRODUCTION

In the last decade, the research towards self-driving cars has picked up a staggering pace. The main objective of this technology is to make our roads safer than ever before [1]. A key ingredient to realize the goals of autonomous vehicles is a robust perception system, where the main objective is to understand the environment in which the robot is operating, through a variety of sensors that a robot is endowed with. In this paper we focus of semantic scene understanding of urban outdoor environment using 3D LiDAR scans. Understanding the semantics is necessary, as it paves the way for robust visual localization [18, 22], efficient mapping [26], among several other tasks.

In this paper we propose a deep convolutional neural network (DCNN) architecture for the task of semantic segmentation of a 3D LiDAR scan into the following semantic categories: *car*, *pedestrian* and *bicyclist*. Recently, deep neural network based methods have lead to breakthroughs in several vision tasks, such as classification [27, 9, 11], detection [24, 23, 16, 31, 20] and segmentation [17, 4, 25, 3, 13, 29, 30]. Majority of these methods are based on camera images [27, 9, 11, 24, 23, 16, 17, 4, 25, 3] and few methods have focused on using 3D LiDAR scans [29, 30, 31, 20]. Our proposed architecture is based on dense blocks [11]. To reduce the number of parameters, we replace the standard convolution layers with depth separable convolution layers [6] for dense blocks in the decoder. This allows us to reduce the number of parameters by a significant amount while still having competitive performance. Standard DCNN architectures treat each example independently and do not use any previous or prior information. Especially in the case of perception in robotics, the data is sequential. To leverage over this sequential nature of information, we propose a Bayes filter approach for making our segmentation results temporally consistent. More concretely, we use a Bayes filter with a *static* state, where static in this context means that transition between different states is unlikely, which is true for semantic classes. This approach neatly combines the current prediction of the neural network, with the information accumulated from previous scans. In our previous work [8], such an approach was part of our method to classify points in a 3D LiDAR scan as non-movable, movable and dynamic. We illustrated its advantages through qualitative results. In this paper, we thoroughly analyze our method by evaluating our approach on various sequences of KITTI tracking benchmark and report both qualitative and quantitative results.

All authors are with the Department of Computer Science at the University of Freiburg, Germany.

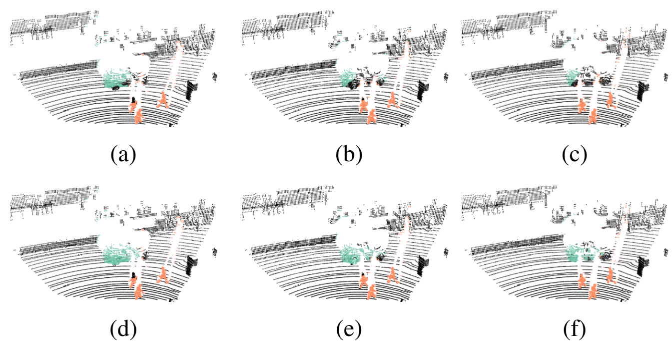


Fig. 1: Illustration of semantic segmentation with our proposed methods. In the top row ((a)-(c)), we show the output of our proposed DCNN, for three consecutive scans. In the top left image, points on a car (color green) are correctly classified, but in subsequent scans, points on the same car are partially ((b)-(c)) misclassified as background. In the bottom row, we show the output of our proposed binary Bayes filter. For all the scans, points on the same car are correctly classified.

The main contributions of this paper include a DCNN for semantic segmentation of LiDAR scans into the classes: *car*, *pedestrian* and *bicyclist*. We compare our DCNN with state-of-the-art DCNNs [29, 30, 28], proposed to solve the same task. To justify different architecture design choices and gain further insight towards them, we also present an ablation study. Our next contribution is a Bayes filter approach for making the predictions of the neural network temporally consistent. This approach leverages over the sequential nature of the input data stream and makes our segmentation system robust towards sporadic erroneous prediction. For comparison, we use our proposed architecture as a baseline method. The dataset, code and learned models is available here. [1]

II. Related Work

With the advent of deep neural networks, a significant progress has been made towards solving a variety of tasks, including the task of semantic segmentation. Regarding 2D images, a plethora of research has been done in last few years [17, 25, 3, 13, 4], pushing the boundary of state-of-the-art results to the limit. A similar progress has not been in the field of semantic segmentation of 3D pointcloud data due to inherent differences in the two data modalities. In the case of 2D images, the input data to the network is fixed but in the case of 3D data, multiple representations are possible. Regarding the current task, the most commonly used representation are either a collection of 3D points or projecting the pointcloud on a 2D image. For the first representation, the PointNet architecture proposed by Qi et al. [19] is a popular choice for learning from unordered pointcloud. They propose to use a multi layer perceptron, for learning features from individual points and then use a symmetric function to combine features learned from points, as a *global* representation. A symmetric function is necessary in this case, in order to make the learned representation invariant to the permutations of the input point set. For the task of classification, the learned global representation is sufficient but for segmentation they propose to combine the global representation with learned local features. Extending PointNet, they proposed PointNet++ [21]. The extension include hierarchical learning, where a set of points (centroids) are sampled from the input point set and then points in the neighborhood of the centroids are grouped together, which is then followed by the PointNet architecture. The grouping of points in the metric space, enable learning of local contextual information. They have shown results primarily on indoor sequence for the data collected from RGB-D sensors. In our case, we use a LiDAR scanner for segmentation of urban outdoor environments. The data from LiDAR scanner is sparser in comparison to the RGB-D sensor and the outdoor environment is more spread out in comparison to confined indoor spaces. In our case, we use the second representation i.e. projecting the 3D LiDAR scan on to a 2D image. This allows us to represent a LiDAR scan in a compact fashion and furthermore the advancements made in the field of semantic segmentation using 2D images can be used as well.

Focusing on the task of semantic segmentation using 2D images, one of the initial architectures was proposed by Long et al. [17]. They proposed an encoder-decoder style, fully convolutional network (FCN) architecture and other architectures since then have followed the same paradigm. Jégou et al. [13] proposed a dense block based DCNN for the task of semantic segmentation. The main differences between our DCNN and theirs is that we use depth separable convolution layers for dense blocks in the decoder. To down-sample the feature maps they proposed a transition down block comprising of a composite function implementing different operations. We replace this block with a single max-pooling operation and show that instead of a composite function, this single operation is sufficient. In the presented ablation study, we justify these proposed changes.

We compare our proposed architecture with the architectures proposed in [29, 30, 28]. The first architecture proposed by Wu et al. [29] is based on the SqueezeNet [12] architecture. They use *fire* modules, which first involves squeezing the feature maps using 1×1 filters and then expanding these squeezed feature maps in parallel using filters of size 1×1 and 3×3 and concatenating their outputs at the end. Using three max-pool layers they down-sample the feature maps only along the width dimension and to up-sample the feature maps they again use *fire* modules in the decoder. Last layer of their neural network architecture is a recurrent CRF and the complete architecture is trained end-to-end. In our ablation study, we compare with their proposed down-sampling technique.

They further improve this architecture in [30] by using a binary mask as an additional input channel. This mask indicates existence of a LiDAR measurement corresponding to a pixel location. Along this they also introduce a novel context aggregation module to limit the error introduced by missing LiDAR measurements and furthermore in order to tackle the class imbalancing problem they use focal loss [15] for training their DCNN. The last method we compare with is the DCNN proposed by Wang et al. [28]. Similar to the neural network architectures proposed by Wu et al. [29], their network architecture is also based on SqueezeNet. They also Squeeze Excitation blocks [10] after the initial *fire* modules and at the end of the encoder use an *enlargement* layer which is based on the Atrous Spatial Pyramid Pooling [5].

III. Network Architecture

In Fig.2 we illustrate the complete framework for semantic segmentation of a LiDAR scan. The first step is to project the scan onto different 2D images and each such image encodes a specific modality. These images are then stacked together and are passed through our proposed DCNN for semantic segmentation. The segmentation mask predicted by the DCNN is then projected back to the LiDAR scan to infer pointwise semantic labels.

For the task of semantic segmentation we a propose a novel fully convolutional DCNN architecture called DBLiDARNet. Our architecture is based on dense blocks and is shown in Fig.2. Similar to other DCNN architecture proposed for the task of semantic segmentation [13, 17, 25], our network is also comprised of an encoder for learning the features required for the task while down-sampling the feature map size and a decoder to up-sample the feature maps so that the last hidden layer has the same spatial resolution as the input image. In the encoder, we have two convolution layers (conv_0 and conv_1), three dense blocks (db_0, db_1

[1] http://deep-temporal-seg.informatik.uni-freiburg.de

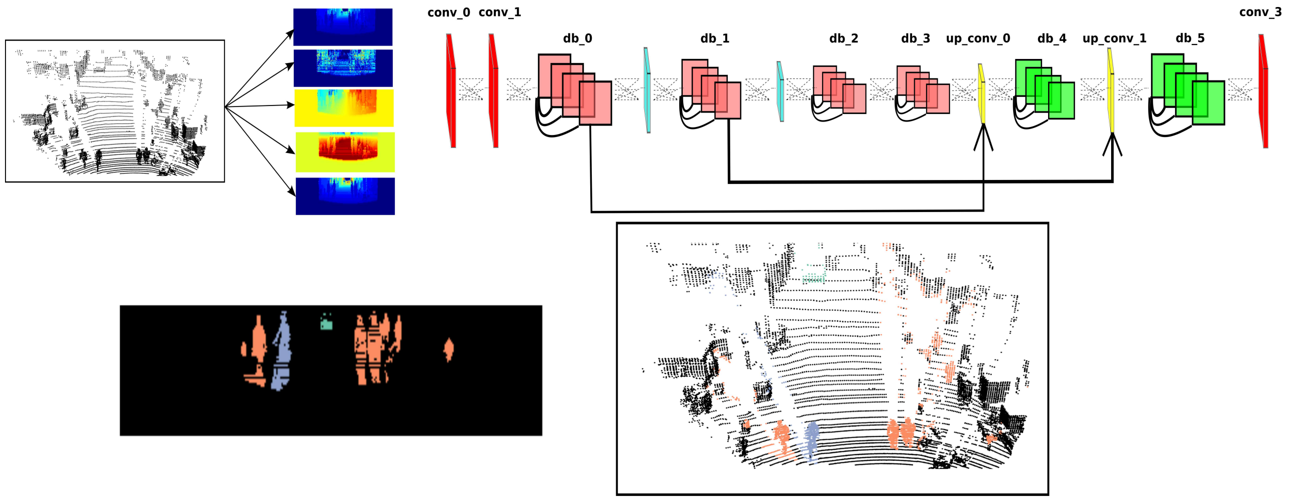


Fig. 2: Our proposed semantic segmentation framework. In the first step we project a LiDAR scan onto five 2D images and encode the following modalities: depth, surface reflectance intensity, 3D coordinates(x, y, z). These images are then stacked together, fed into the proposed CNN architecture, and the output is the predicted segmentation mask (bottom left). The segmentation information is then projected back to the scan to infer the semantic labels for each point in the scan. Our proposed architecture for semantic segmentation. In the encoder we have two convolution layers (conv_0 and conv_1), two max-pooling layers and three dense blocks (db_0, db_1 and db_2). In the decoder we use two up-convolution layers to up-sample the feature maps, two dense blocks (db_3, db_4) with depth separable convolution and one convolution layer (conv_2). We use skip connection to concatenate feature maps from the encoder, in order to recapture the details lost due to up-sampling.

and db_2) and two max-pooling layers to down-sample the feature maps $4\times$ in comparison to the spatial resolution of the input image. In the decoder we use two up-convolution layers to up-sample the feature maps and use two more dense blocks with depth separable convolution layers. To limit the number of learnable parameters in the decoder, similar to the architecture proposed by Jégou et al. [13], in our proposed architecture the input to the up-convolution layers is the feature maps learned by the dense block prior to the up-convolution layer instead of all the features maps learned till that point. For instance the input to the layer up_conv_0 is only the feature maps learned by the dense block db_2. To recapture the information lost during up-sampling we use skip connections to concatenate the feature maps from the encoder to the output of the up-convolution layers.

The input to our proposed architecture is a five channel 2D image, which is generated after projecting the 3D LiDAR scan onto a spherical plane. This channels encode following modalities: depth, surface reflectance intensity and, 3D coordinates(x, y, z). The resolution of the image after the projection is skewed i.e. the height of the image is smaller than the width since the vertical field-of-view of the scanner is lower than the horizontal field-of-view. In comparison to the data captured from standard RGB cameras, where the height and the width of the image are similar, in our the case the height (64) is $8\times$ smaller than the width (512) of the image.

To limit the number of operations within different layers, make the architecture memory efficient and increase the receptive field, a common practice is to down-sample the feature maps, where the down-sampling rate varies from $16\times$ to $32\times$, depending on the task. For the task of semantic segmentation where dense pixel wise prediction is required, a large down-sampling rate can lead to decrease in performance [5]. Since height of our input image is 64, reducing the feature map size 16 or $8\times$ will result in a feature map with height dimension as 8 or 4, thereby resulting in a significant loss of information. Therefore, to arrest the information lost due to down-sampling operation we only reduce the feature map size $4\times$. Other methods learning from images of same resolution have proposed down-sampling the feature maps $8\times$ ([29, 30]) but only along the width dimension and keeping the height dimension unchanged or down-sampling $4\times$ and use dilated convolutions in the last layers of the encoder [28]. In the ablation study, we report results for a network where we down-sample $8\times$ along the both spatial dimensions to justify our decision of down-sampling $4\times$. We also train a network where we down-sample only $4\times$ but only along the width dimension, to compare with the down-sampling method proposed in [29, 30].

The complete details regarding the dimensions of each layer or block and different associated hyper-parameters is reported in Tab.I. The kernel size of the filter for all the convolution and up-convolution layers except conv_2 is 3×3. In the last layer we use a filter of size 1×1 to reduce the number feature maps to the number of classes. The *stride* for each convolution layer is set to 1 and the *stride* for up-convolution layer is set to 2. For all dense blocks the *growth rate* parameter is set to 16. The number of features learned within a dense block is *growth rate* times the *repetition*, where *repetition* is the number of times the

TABLE I: Architecture

Layer name	Dimension (H×W×C)	Repetition	Depth separable
conv_0	$64 \times 512 \times 48$	-	No
conv_1	$64 \times 512 \times 48$	-	No
db_0	$64 \times 512 \times 144$	6	No
db_1	$32 \times 256 \times 272$	8	No
db_2	$16 \times 128 \times 432$	10	No
db_3	$16 \times 128 \times 240$	15	Yes
up_conv_0	$32 \times 256 \times 240$	-	No
db_4	$32 \times 256 \times 128$	8	Yes
up_conv_1	$64 \times 512 \times 128$	-	No
db_5	$64 \times 512 \times 96$	6	Yes
conv_3	$64 \times 512 \times 4$	-	No

composite function within a block is repeated. As mentioned before, the input to an up-convolution layer is only the number of feature maps learned in the previous dense block and therefore the output of the db_3 only contains the feature maps learned within the block (16×15), in contrast to output of db_2 which consists of feature maps learned within the block (16×10) concatenated with the number of input feature maps (272). We use skip connection as showed in Fig.2. The input to the dense blocks in the decoder (db_4 and db_5) is concatenation of the feature maps learned by the previous up-convolution layer and the output of the dense block (in the encoder) with same spatial resolution. In this case the input to db_4 is the output of up_conv_0 concatenated with the output of db_1.

A. *Training*

Our complete network architecture is implemented in TensorFlow [2]. We use the dataset provided by Wu et al. [29], consisting of 8057 images for training and 2791 images for testing. We use softmax cross-entropy loss and use the Adam optimizer [14] with a learning rate of $1e^{-4}$, weight decay of $5e^{-4}$ and batch size of 2. Among the three classes, the point measurements from cars is significantly more than the measurements from either pedestrians or bicyclists, mainly because of the inherent difference in the size of the geometrical structure. This leads to the problem of class imbalancing, where some classes in the training data overwhelm the classes which are under represented. To tackle this we use a weight balancing technique and assign larger weights to points belonging to the class *pedestrians* and *bicyclists* in comparison to points belonging to the class *cars* and *background*.

IV. Bayes Filter Method

In the Sec.III, we proposed a novel DCNN architecture for semantic segmentation of a LiDAR scan into different categories. The output of the network is the predicted softmax probabilities of a point in a scan belonging to different categories. Since this prediction is performed independently for different scans, in this section we introduce a novel Bayes Filter approach to make our pointwise prediction temporally consistent. This approach assumes the scans are sequential with significant overlap and the objective is to leverage over this sequential nature of information and make our prediction robust to isolated erroneous predictions from the neural network.

The semantic state of a point is *static*, i.e. it remains same over time and transition between these states is unlikely. For each point, we use three separate binary Bayes filters with *static* state, to estimate the belief for each class independently. To estimate the belief for a class c, for a point $\mathbf{p}^t \in \mathbb{R}^3$ in a scan at time t , we first define a binary random state variale $O_c^t = \{0, 1\}$, where $O_c^t = 1$ indicates that the point belongs to the class c and $O_c^t = 0$ indicates the opposite. Without loss of generality, from now on, we would write $Bel(O_c^t = 1)$ as $Bel(O_c^t)$ and $Bel(O_c^t = 0)$ as $Bel(\neg O_c^t)$. The current belief $Bel(O_c^t)$ depends only on the predictions of the neural network, $\xi_c^{1:t}$, for the class c as shown in Eq.(1).

$$Bel(O_c^t) = P(O_c^t \mid \xi_c^{1:t}), \tag{1}$$

where, $\xi_c^{1:t}$ are softmax scores for the class c. We define such binary random variables for each class and estimate the belief for each class independently.

Using Bayes rule and Markov assumption we can rewrite the Eq.(1) as following,

$$P(O_c^t \mid \xi_c^{1:t}) = \frac{P(\xi_c^t \mid O_c^t) P(O_c^t \mid \xi^{1:t-1})}{P(\xi^t \mid \xi^{1:t-1})}. \tag{2}$$

Using Bayes rule for the term $P(\xi_c^t \mid O_c^t)$, Eq.(2) can be modified as following,

$$P(O_c^t \mid \xi_c^{1:t}) = \frac{P(O_c^t \mid \xi_c^t) P(\xi_c^t) P(O_c^t \mid \xi^{1:t-1})}{P(O_c^t) P(\xi^t \mid \xi^{1:t-1})}. \tag{3}$$

Similarly, $P(\neg O_c^t \mid \xi_c^{1:t})$ can be written as,

$$P(\neg O_c^t \mid \xi_c^{1:t}) = \frac{P(\neg O_c^t \mid \xi_c^t) P(\xi_c^t) P(\neg O_c^t \mid \xi^{1:t-1})}{P(\neg O_c^t) P(\xi^t \mid \xi^{1:t-1})}. \tag{4}$$

We now introduce the log odds notation, where odds of an event x is defined in Eq.(5) and the log odds are defined in Eq.(6)

$$\frac{p(x)}{\neg p(x)} = \frac{p(x)}{1 - p(x)}, \tag{5}$$

$$l(x) = \log \frac{p(x)}{1 - p(x)}. \tag{6}$$

The odds for a point $\mathbf{p}^t$ having the semantic class c can be estimated by dividing Eq.(3) with Eq.(4). The odds is defined in Eq.(7) and the log odds are defined in Eq.(9),

$$\frac{P(O_c^t \mid \xi_c^{1:t})}{P(\neg O_c^t \mid \xi_c^{1:t})} = \frac{P(O_c^t \mid \xi_c^t)}{P(\neg O_c^t \mid \xi_c^t)} \frac{P(O_c^t \mid \xi^{1:t-1})}{\neg P(O_c^t \mid \xi^{1:t-1})} \frac{P(\neg O_c^t)}{P(O_c^t)}, \tag{7}$$

$$= \frac{P(O_c^t \mid \xi_c^t)}{1 - P(O_c^t \mid \xi_c^t)} \frac{P(O_c^t \mid \xi^{1:t-1})}{1 - P(O_c^t \mid \xi^{1:t-1})} \frac{1 - P(O_c^t)}{P(O_c^t)}, \tag{8}$$

$$l_t(O_c^t) = \log \frac{P(O_c^t \mid \xi_c^t)}{1 - P(O_c^t \mid \xi_c^t)} + l_{t-1}(O_c^t) - l_0(O_c^t), \tag{9}$$

where, the current measurement is defined as following,

$$P(O_c^t \mid \xi_c^t) = \xi_c^t. \tag{10}$$

In Eq.(9), $l_t(O_c^t)$ are the log odds for the belief at time t, the first term on the right side in Eq.(9) are the log odds for the current measurement, $l_{t-1}(O_c^t)$ are the log odds for the previous belief and $l_0(O_c^t)$ are the log odds for the initial belief. Through this formulation, our inference not only depends on the current measurement ($P(O_c^t \mid \xi_c^t)$), but also on the previous measurements, incorporated through the recursive term $l_{t-1}(O_c^t)$. To enable this recursive behavior, data association between points in consecutive scans is required and for this we use our method of estimating pointwise motion proposed in [7]. We performed data association by aligning scans using the estimated motion and choosing the nearest point on the basis of Euclidean distance as the corresponding point. As mentioned before, we estimate $l_t(O_c^t)$ for each class separately and for the inference we choose the class with the largest odds.

V. Results

A. Network Architecture

To evaluate our proposed DCNN, we use the test set from the dataset provided by Wu et al. [29]. We report class wise IoU and compare our results with two DCNN proposed by Wu et al. ([29],[30]) and the network architecture proposed by Wang et al. [28]. In Fig.3, we show qualitative semantic segmentation results. Our proposed DCNN is able to segment objects of different classes successfully (top row) and is able to tackle cases where an object is heavily occluded (middle row). We also illustrate a case where our method sometimes over segments a bicyclist into classes *pedestrians* and *bicyclists*. This primarily happens because a person is part of both classes, in one case a person is walking and in the other case a person is riding a bicycle.

In Tab.II we report the class wise IoU and mean IoU for different methods. Our proposed DCNN outperforms the existing state-of-the art DCNNs proposed for the same task and has a better IoU for all the three classes. In the case of *pedestrian*, the increase in IoU is around 70%, for the class *bicyclist* the increase is around 17%, with an overall increase in mean IoU by 16%. These results indicate a remarkable improvement over the existing DCNNs proposed to solve the same task. Comparing the runtime, our DCNN has the largest runtime but considering that scan rate for LiDAR scanners is around 10Hz, our method still provides real time performance.

Comparing the inter class performance, the highest IoU is achieved for the class *car*, whereas the performance for *pedestrian* and *bicyclist* are comparable. Similar trend is evident for other methods as well. This difference in performance has three main reasons, firstly the number of instances of *pedestrian* and *bicyclist* is lesser in comparison to *car*. Secondly, object in both these classes have a smaller size in comparison to cars and therefore the number of points sampled from their surface is significantly lower in comparison to points sampled from the surface of cars. Due to these reasons, these two classes are under represented and as mentioned before, we use weight balancing in order to have a large penalty for misclassifying points in these classes. The last reason is the over segmentation of points on a bicyclist into classes *bicyclist* and *pedestrian* as shown in Fig.3. This misclassification is not a common occurrence but still hampers the overall performance.

TABLE II: A comparison with other DCNNs proposed for semantic segmentation of a LiDAR scan. For each method we report class wise and mean IoU

Method	Cars	Pedestrians	Bicyclists	meanIoU	t [ms]
SqueezeSeg [29]	60.9	22.8	26.4	36.7	**8.7**
SqueezeSeg w/ CRF [29]	64.6	21.8	25.1	37.1	13.5
PointSeg [28]	67.4	19.2	32.7	39.7	12
PointSeg w/ RANSAC [28]	67.3	23.9	38.7	43.3	14
SqueezeSegV2 [30]	73.2	27.8	33.6	44.8	-
DBLiDARNet (Ours)	**75.1**	**47.4**	**45.4**	**56.0**	40

TABLE III: Results for ablation study. For each method we report class wise and mean IoU.

Method	Cars	Pedestrians	Bicyclists	meanIoU	t [ms]
100 Layer Tiramisu [13]	74.2	48.7	43.7	55.5	41
TD block [13]	72.2	48.3	41.2	53.9	43
Down-sample 8×	74.1	43.8	39.7	52.5	43
Down-sample width 4×	74.7	45.0	38.6	52.8	66
db_3 depth separable	74.2	49.2	36.8	53.4	41
db_3 + db_2 depth separable	73.6	41.2	33.2	49.3	**40**
W/o weight balance	72.4	40.9	35.1	49.5	41
DBLiDARNet	**75.1**	**47.4**	**45.4**	**56.0**	41

1) Ablation Study: In this ablation study, we justify the network design choices we mentioned in Sec.III. We first discuss the differences between our dense blocks based fully convolutional network and the architecture proposed by Jégou et al. [13].

1) Their architecture consists of transition-down block for down-sampling the feature maps. This block implements a composite function comprising of batch normalization, ReLU activation, convolution layer (1×1), dropout and max-pooling. We replace this transition-down block by a max-pooling layer. This decision is based on our empirical findings, which showed replacing this block which contains learnable parameters by a max-pooling layer helps in reducing the parameters while maintaining similar performance.
2) As as already mentioned and shown in Fig.2, we use depth separable convolution layers instead of convolution layers for dense blocks in the decoder. This again helps in reducing the learnable parameters significantly without reducing the performance. The number of parameters for our proposed network is 2.8M and their architecture is 3.6M. This large difference between the parameters is mainly attributed to depth separable convolution.

The architecture proposed by Jégou et al. [13] consists of five transition down blocks for down-sampling the feature maps 32×. They, therefore use five up-convolution layers in the decoder along with same number of dense blocks. Such a high down-sampling rate will result in significant loss of information for reasons discussed before (Sec.III). Therefore in our implementation of their architecture we only use two transition down blocks instead of five. In Tab.III we report

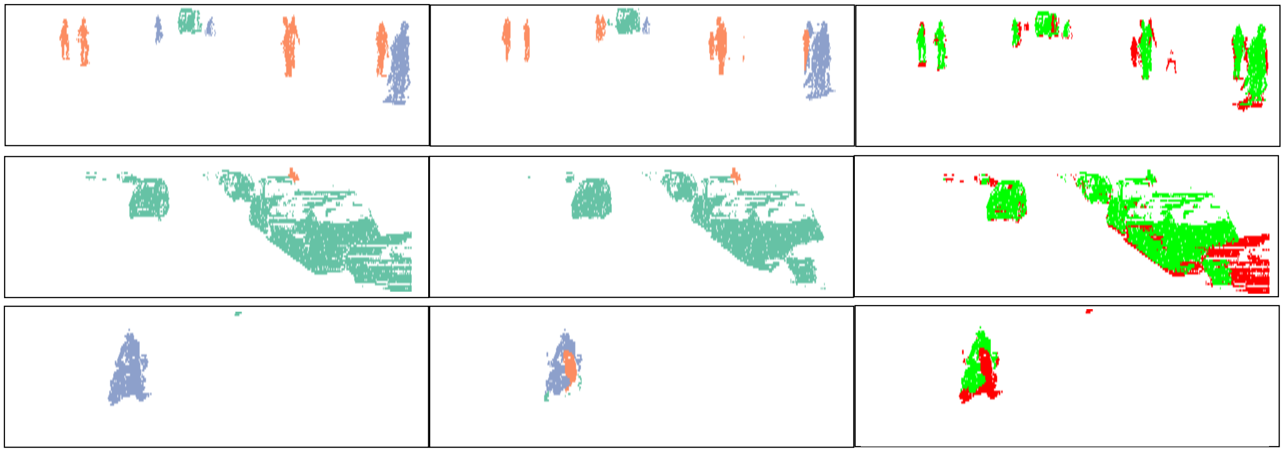


Fig. 3: An illustration of the semantic segmentation results. In the left column we show the ground-truth segmentation masks where points belonging to the class *car*, *pedestrian* and *bicyclist* are show in color green, orange and blue respectively. In the middle column we show the predicted segmentation masks with the same color scheme as the ground-truth masks. To clearly visualize the differences between the ground-truth and predicted masks, in the last we show the correctly segmented points in green color and the misclassified points in color red. The top row illustrates the case where our proposed DCNN is able to successfully segment objects of different classes. The middle row shows a hard case, where a pedestrian is walking behind the cars and is heavily occluded and our method is still able to correctly segment the pedestrian. The bottom row illustrates a case where our method under performs. In some cases bicyclists are over segmented into the classes *bicyclist* and *pedestrian* due to presence of a person in both classes.

both class wise and mean IoU for their architecture and also for a model where we use our architecture but replace max-pool layers with transition down (TD) blocks. Our proposed architecture outperforms their architecture marginally while using fewer parameters. Using transition down blocks instead of a max-pooling layer leads to a slight decrease in performance as well. These results clearly indicate that our proposed changes help in reducing the parameters while improving the performance.

Comparing different down-sampling strategies, we trained two different models. For the first model we down-sample 8× instead of 4 and for the second model we down-sample 4× but only along the width dimension while keeping the height unchanged, similar to [29]. As reported in Tab.III, for the first model (down-sample 8×), the IoU for the class *car* remains comparable but a decrease in performance is observed for the other classes. In comparison to cars, pedestrians and bicyclists are smaller and therefore a large down-sampling rate adversely affects these classes in comparison to other classes. For the second model, similar to the first, a noticeable decrease in performance is observed for both *pedestrian* and *bicyclist* classes. Without decreasing the height, feature maps have larger spatial resolution, thereby requiring more operations. Therefore for this case, the inference time increases by 20ms. Even though large down-sampling rate can hamper the performance, especially for the task of semantic segmentation, it is still required for increasing the receptive field as well as making the model efficient considering both the memory and computational requirements. Our proposed strategy of down-sampling the feature maps 4× allows us to exploit the advantages of such operations without losing the crucial information necessary for predicting accurate segmentation masks.

Depth separable convolution is an ingenious way of reducing parameters, but excessively using it can potentially decrease performance. To justify this, we train two models, using depth separable convolution in the last dense block of the encoder (db_3) and then in last two dense blocks together (db_3 + db_2). This decreases number of parameters from 2.8M (DBLiDARNet) to 1.9M and 1.4M respectively. In both cases performance decreases, especially for the second case the decrease is substantial.

To limit the detrimental impact of class imbalancing on the overall performance, we use weight balancing. In the loss function, the contributions made by the under represented classes are multiplied by a large weight, thereby incurring a large penalty if points from these classes are incorrectly classified. The lowest weight is assigned to the *background* class, while weights in increasing order is assigned to classes *car*, *pedestrian* and *bicyclist* respectively. To analyze the impact of weight balancing, we trained a model where we did not use balancing and report results for this in Tab.III. The decrease in performance is evident for all the classes, where the most under represented class suffers the most with performance decreasing for the class *bicyclist* by 22% and for class *pedestrian* by 5%. These results highlights the necessity of using the weight balancing technique.

B. Bayes Filter

To evaluate our proposed Bayes filter approach, we use the KITTI tracking benchmark. The benchmark contain 20 sequences and to evaluate our approach on all of the sequences, we split the sequences into two different sets. We train our network on both sets separately and use the other set for testing i.e. we train a model on the first set

TABLE IV: Splitting of sequences in KITTI tracking benchmark

Seq. ID	# of scans	Cars	Pedestrains	Bicyclists
Set 1				
0	153	528	21	153
4	313	908	65	60
5	296	1307	0	139
6	269	661	0	0
8	388	1334	0	0
9	802	3135	29	0
10	293	673	30	14
11	372	3579	197	0.0
19	1058	1411	6595	306
Set 2				
2	229	1127	177	75
3	143	38	0	0
7	799	2488	67	0
12	77	142	64	42
13	339	123	1096	237
14	105	523	120	0
15	375	899	751	537
16	208	832	2019	271
17	144	0	776	100
18	338	1413	0	0
20	836	6244	0	0

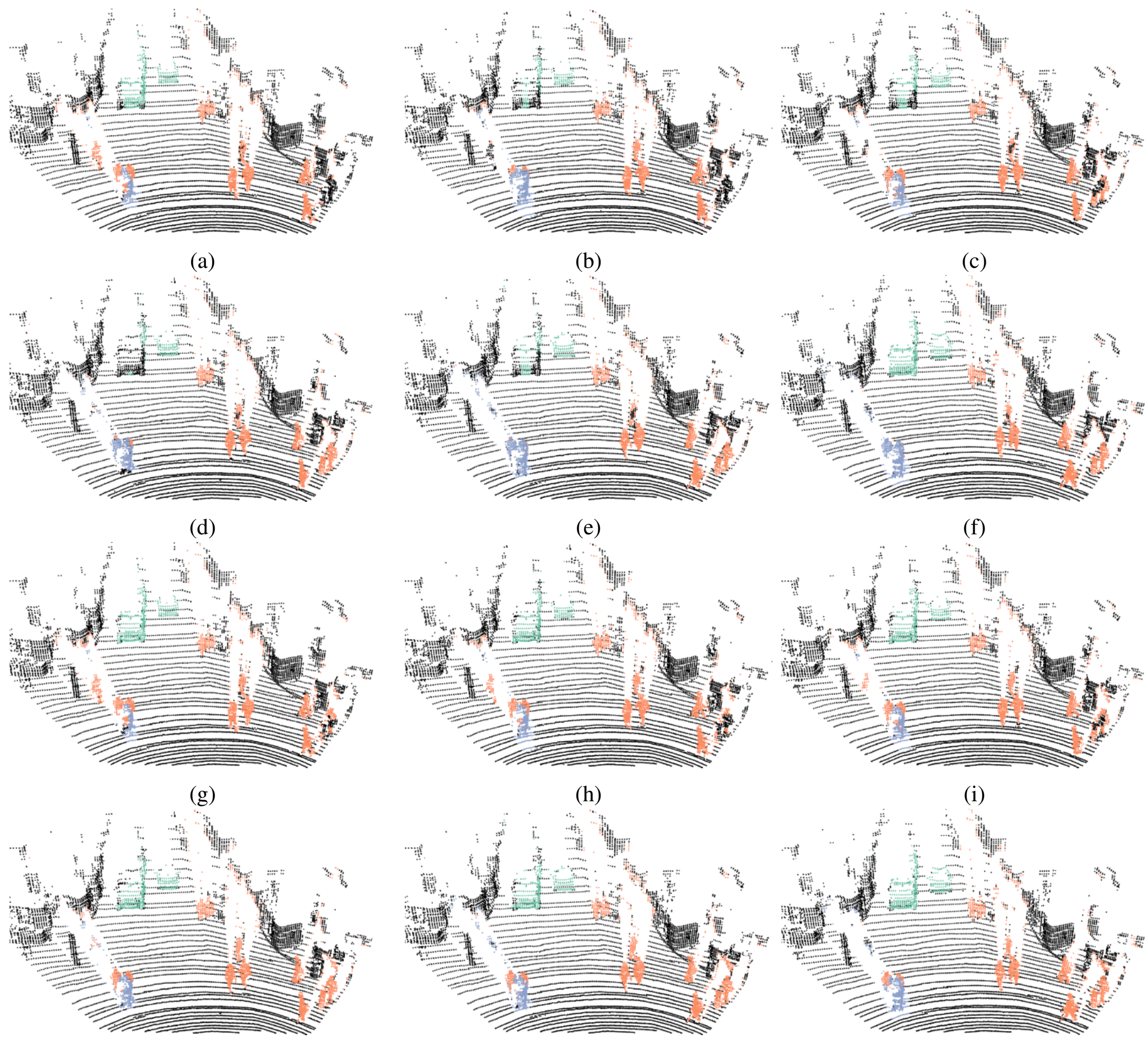
TABLE V: Class wise IoU for DCNN and the binary object Bayes filter

Seq. ID	DBLiDARNet			Object Bayes Filter		
	Cars	Pedestrians	Bicyclist	Cars	Pedestrians	Bicyclist
0	76.2	2.0	**29.6**	**79.2**	2.0	23.6
2	54.9	37.0	0.0	**55.3**	**46.9**	0.0
3	75.2	-	-	**75.5**	-	-
4	66.6	40.8	35.2	**69.1**	**47.4**	**53.2**
5	**70.1**	-	-	70.0	-	
6	**87.2**	-	-	87.1	-	-
7	83.2	28.2	-	**83.5**	**32.7**	-
8	66.9	-	-	**69.9**	-	-
9	71.9	18.6	-	**72.9**	**21.6**	-
10	72.4	0.0	0.0	**75.1**	0.0	0.0
11	88.4	**15.6**	-	**89.6**	15.3	-
12	51.5	0.0	**4.0**	**58.5**	0.0	1.6
13	24.2	**50.7**	39.5	**31.3**	50.6	**41.1**
14	**89.6**	40.2	-	86.3	42.6	-
15	83.9	70.1	5.0	**85.7**	**72.5**	5.0
16	63.8	75.3	54.5	**64.1**	**77.0**	**60.7**
17	-	81.8	0.0	-	**83.7**	0.0
18	84.7	-	-	84.7	-	-
19	68.4	66.2	36.9	**74.0**	**66.1**	**37.8**
20	69.1	-	-	**69.4**	-	-

and test the learned model on the second set and then train on the second set and test on the first set. While splitting the sequences we assure the number of scans and the instances of the different classes have similar distribution. In Tab.IV we report the number of scans in each sequence and the number of instances of each class in a given sequence. Among the three classes, number of instances of class *bicyclist* is minimum and instances of class *car* in large numbers is consistently prevalent across sequences. As mentioned before, the number of point measurements from the surface of pedestrians and bicyclists is significantly less in comparison to the measurements from cars. Therefore due to limited instances and smaller size, segmenting these classes is challenging.

For training the network we use our proposed network with the exact same parameters as discussed in Sec.III-A, with the one difference. In this case the input resolution of the images are $64 \times 324 \times 5$, in comparison to $64 \times 512 \times 5$. For evaluating the proposed Bayes filter we use our network as the baseline method and report comparison with the segmentation results from the network. In Fig.4, we illustrate the differences in the segmentation results for a sequence of six consecutive scans. In the top two rows, we shows results for our proposed neural network and in the bottom two rows we show results for our proposed Bayes filter approach. In the case of neural network, points on a car are correctly classified in the first scan but in the next few scans, points on the same car are misclassified as background. For the same scans, our proposed Bayes filter is able to consistently classify points on the car correctly. These results clearly illustrates that our Bayes filter approach successfully leverages over the sequential nature of the input data, to correct the segmentation, thereby making our predictions temporally consistent. In Tab.V, we report class wise IoU for different sequences, for both our DCNN and the Bayes filter approach (handcrafted and learned descriptor). In the cases where no instances of a class is observed, we do not report results as well (indicated by a dash sign). Analyzing the neural network predictions, our DCNN is consistently able to segment *cars* in comparison to the other classes. Since the LiDAR scanner is mounted on a vehicle, it shares the same space where other vehicles operate, in comparison to pedestrians or bicyclist which are either walking or biking on a sidewalk or a bike lane. This also explains why the instances of cars outnumbers pedestrians or bicyclists by a significant margin. In the cases of pedestrians, a high IoU is achieved for the cases where pedestrians are in close proximity of the vehicle collecting the sensor data for instance on a crowded small street or at an intersection. For some sequences, the IoU for the class bicyclists is zero. In these cases, majority of times these objects are either far from the sensor or occluded and in the rare cases they are misclassified as pedestrians. In the case of LiDAR data, with the increase in distance the data gets sparser and especially in the case of bicyclists or pedestrians, the surface is smaller in comparison to cars and therefore they are not enough point measurements to a make an accurate prediction.

Comparing the DCNN results with the Bayes filter approach, across different sequences and classes, an improvement in IoU is consistently observed after using the Bayes filter approach. For most cases the improvement in IoU is around 4% to 9% but an improvement of 27% is achieved for class *pedestrian* in sequence 2 and staggering improvement of 51% is achieved for class *bicyclist* in sequence 4. For couple of isolated cases, a decrease in IoU is observed after using the filter approach. The implicit assumption of our Bayes filter approach is that the predictions from DCNN is seldom wrong and for cases, the filter uses the previous knowledge to correct those predictions. In the rare cases where this assumption is violated, the information accumulated by the filter spurs from incorrect measurements and therefore the filter approach needs multiple correct predictions from DCNN to improve its knowledge in comparison to a single prediction



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j) (k) (l)

Fig. 4: Illustration of semantic segmentation with the object Bayes filter. In the top two rows ((a)-(f)), we show the output of our proposed DCNN, for six consecutive scans. In the top left image, points on a car (top left) are correctly classified, but in subsequent scans, points on the same car are first partially ((b)-(c)) and then completely ((d)) misclassified as background. In the bottom two rows, we show the output of our proposed binary Bayes filter for the same six consecutive scans. For all the six scans, points on the same car are correctly classified. These results clearly illustrate that our proposed Bayed filter method is able to successfully mitigate the sporadic erroneous predictions from the neural network.

needed by DCNN. For instance, in the sequence 0, points on a bicyclist were labeled as pedestrian more than often, causing Bayes filter to accumulate the incorrect predictions.

Through these qualitative and quantitative results we show the importance of our proposed static binary Bayes filter approach. Having such an approach is especially necessary in the cases, where the input data is sequential which is seldom not true in the case of perception in robotics. Through the Bayes filter approach we completely exploit this sequential nature of the input data, making our predictions temporally consistent and report a persistent improvement in IoU across different sequences and classes.

VI. Conclusions

In this paper, we proposed a DCNN to segment points in a 3D LiDAR scan into multiple semantic categories. Our proposed architecture is based on dense blocks and uses depth separable convolution to reduce the parameters while still maintaining competitive performance. It significantly outperforms state-of-the-art neural network architectures, with an average improvement of around 16% across different classes. In the presented ablation study, we justify our architecture choices. The neural network predicts the segmentation mask for each scan independently and to make these predictions temporally consistent, we proposed a Bayes filter method. Through extensive evaluation on the KITTI tracking benchmark, we report a consistent improvement across classes and sequences. These results clearly show the need of such an approach, especially when the input data is sequential, which is rarely not true in the case of robotic perception.

REFERENCES

[1] Waymo: Our Mission. `https://waymo.com/mission/`, 2019. [Online; accessed 28-May-2019].

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[3] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv: 1511.00561*, 2015. URL `http://arxiv.org/abs/1511.00561`.

[4] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2018.

[6] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[7] Ayush Dewan, Tim Caselitz, Gian Diego Tipaldi, and Wolfram Burgard. Rigid scene flow for 3d lidar scans. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[8] Ayush Dewan, Gabriel L Oliveira, and Wolfram Burgard. Deep semantic classification for 3d lidar data. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3544–3549. IEEE, 2017.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[10] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[11] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*, 2016.

[12] 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.

[13] Simon Jégou, Michal Drozdzal, David Vazquez, Adriana Romero, and Yoshua Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*, pages 1175–1183. IEEE, 2017.

[14] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[15] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[17] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[18] Tayyab Naseer, Gabriel L Oliveira, Thomas Brox, and Wolfram Burgard. Semantics-aware visual localization under challenging perceptual conditions. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2614–2620. IEEE, 2017.

[19] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017.

[20] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 918–927, 2018.

[21] 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*, pages 5099–5108, 2017.

[22] Noha Radwan, Abhinav Valada, and Wolfram Burgard. Vlocnet++: Deep multitask learning for semantic visual localization and odometry. *IEEE Robotics and Automation Letters*, 3(4):4407–4414, 2018.

[23] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[24] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[25] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[26] Philipp Ruchti and Wolfram Burgard. Mapping with dynamic-object probabilities calculated from single 3d range scans. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Brisbane, Australia, May 2018. URL `http://ais.informatik.uni-freiburg.de/publications/papers/ruchti18icra.pdf`.

[27] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICLR)*, 2015.

[28] Yuan Wang, Tianyue Shi, Peng Yun, Lei Tai, and Ming Liu. Pointseg: Real-time semantic segmentation based on 3d lidar point cloud. *arXiv preprint arXiv:1807.06288*, 2018.

[29] 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 *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1887–1893. IEEE, 2018.

[30] Bichen Wu, Xuanyu Zhou, Sicheng Zhao, Xiangyue 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.

[31] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018.