# SalsaNext: Fast, Uncertainty-aware Semantic Segmentation of LiDAR Point Clouds for Autonomous Driving

Tiago Cortinhal[1], George Tzelepis[2] and Eren Erdal Aksoy[1,2]

*Abstract*— **In this paper, we introduce *SalsaNext* for the uncertainty-aware semantic segmentation of a full 3D LiDAR point cloud in real-time. *SalsaNext* is the *next* version of *SalsaNet* [1] which has an encoder-decoder architecture consisting of a set of ResNet blocks. In contrast to *SalsaNet,* we introduce a new context module, replace the ResNet encoder blocks with a new residual dilated convolution stack with gradually increasing receptive fields and add the *pixel-shuffle* layer in the decoder. Additionally, we switch from stride convolution to average pooling and also apply central dropout treatment. To directly optimize the Jaccard index, we further combine the weighted cross entropy loss with *Lovász-Softmax* loss [2]. We finally inject a Bayesian treatment to compute the *epistemic* and *aleatoric* uncertainties for each LiDAR point. We provide a thorough quantitative evaluation on the Semantic-KITTI dataset [3], which demonstrates that *SalsaNext* outperforms the previous networks and ranks first on the Semantic-KITTI leaderboard.**

## I. Introduction

Scene understanding is an essential prerequisite for autonomous vehicles. Semantic segmentation helps gaining a rich understanding of the scene by predicting a meaningful class label for each individual sensory data point. Safety-critical systems, such as self-driving vehicles, however, require not only highly accurate but also reliable scene segmentation with a consistent measure of uncertainty. This is because the quantitative uncertainty measures can be propagated to the subsequent units, such as decision making modules to lead to safe manoeuvre planning or emergency braking, which is of utmost importance in safety-critical systems. Therefore, semantic segmentation predictions integrated with reliable confidence estimates can significantly reinforce the concept of safe autonomy.

In this work, we introduce a novel neural network architecture to perform uncertainty-aware semantic segmentation of a full 3D LiDAR point cloud in real-time. Our proposed network is built upon the *SalsaNet* model [1], hence, named *SalsaNext.* The base *SalsaNet* model has an encoder-decoder skeleton where the encoder unit consists of a series of ResNet blocks and the decoder part upsamples and fuses features extracted in the residual blocks. In *SalsaNext,* our contributions lie in the following aspects:

- To capture the global context information in the full $360°$ LiDAR scan, we introduce a new context module before encoder, which consists of a residual dilated convolution stack fusing receptive fields at various scales.
- To increase the receptive field, we replaced the ResNet block in the encoder with a novel combination of a set of dilated convolutions (with a rate of 2) each of which has different kernel sizes $(3, 5, 7)$. We concatenated the convolution outputs and combined with residual connections yielding a branch-like structure.
- To avoid any checkerboard artifacts in the upsampling process, we replaced the transposed convolution layer in the *SalsaNet* decoder with a *pixel-shuffle* layer [4] which directly leverages on the feature maps to upsample the input with less computation.
- To boost the roles of very basic features (e.g. edges and curves) in the segmentation process, the dropout treatment was altered by omitting the first and last network layers in the dropout process.
- To have a lighter model, average pooling was employed instead of having stride convolutions in the encoder.
- To enhance the segmentation accuracy by optimizing the Jaccard index, the weighted cross entropy loss was combined with the *Lovász-Softmax* loss [2].
- To further estimate the *epistemic* (model) and *aleatoric* (observation) uncertainties for each 3D LiDAR point, the deterministic *SalsaNet* model was transformed into a stochastic format by applying the Bayesian treatment.

All these contributions form the here introduced *SalsaNext* model which is the probabilistic derivation of the *SalsaNet* with a significantly better segmentation performance. The input of *SalsaNext* is the rasterized image of the full LiDAR scan in the panoramic view. The final network output is the point-wise classification scores together with uncertainty measures. To the best of our knowledge, this is the first work showing the both *epistemic* and *aleatoric* uncertainty estimation on the LiDAR point cloud segmentation task.

Quantitative and qualitative experiments on the Semantic-KITTI dataset [3] show that the proposed *SalsaNext* significantly outperforms other state-of-the-art networks in terms of pixel-wise segmentation accuracy while having much fewer parameters, thus requiring less computation time. *SalsaNext* ranks first place on the Semantic-KITTI leaderboard. We release our source code and trained model to encourage research on the subject [1].

## II. Related Work

As comprehensively described in [5], there exists two mainstream deep learning approaches addressing the seman-

[1]Halmstad University, School of Information Technology, Center for Applied Intelligent Systems Research, Halmstad, Sweden

[2]Volvo Technology AB, Volvo Group Trucks Technology, Vehicle Automation, Gothenburg, Sweden

[1]https://github.com/TiagoCortinhal/SalsaNext

tic segmentation of 3D LiDAR data only: point-wise and projection-based neural networks.

Point-wise methods [6], [7] directly process the raw irregular 3D points without applying any additional transformation or pre-processing. Shared multi-layer perceptron-based PointNet [6], the subsequent work PointNet++ [7], and *superpoint* graph SPG networks [8] are considered in this group. Although such methods are powerful on small point clouds, their processing capacity and memory requirement, unfortunately, becomes inefficient when it comes to the full $360^\circ$ LiDAR scans.

Projection-based methods instead transform the 3D point cloud into various formats such as voxel cells [9], [10], [11], multi-view representation [12], lattice structure [13], [14], and rasterized images [1], [15], [16], [17]. For instance, voxel-based methods discretize the 3D space into 3D volumetric space and assign each point to the corresponding voxel. Sparsity and irregularity in point clouds, however, yield redundant computations since many voxel cells may stay empty. A common attempt to overcome this sparsity problem is to project 3D point clouds into 2D image space either in the Bird-Eye-View [1], [18], [19] or spherical Range-View (RV) [20], [15], [16], [17], [21]. Unlike point-wise and other projection-based approaches, such 2D rendered image representations are more compact, dense and computationally cheaper as they can be processed by standard 2D convolutionals. Therefore, our *SalsaNext* model projects the LiDAR point cloud into 2D RV image.

Bayesian Neural Networks (BNNs) learn approximate distribution on the weights to further generate uncertainty estimates. There are two types of uncertainties: *Aleatoric* which can quantify the intrinsic uncertainty coming from the observed data, and *epistemic* where the model uncertainty is estimated by inferring with the posterior weight distribution, usually through Monte Carlo sampling. Bayesian modelling helps estimating both uncertainty types.

Gal *et al.* [22] proved that dropout can be used as a Bayesian approximation to estimate the uncertainty in classification, regression and reinforcement learning tasks while this idea was also extended to semantic segmentation of RGB images by Kendall *et al.* [23]. Loquercio *et al.* [24] proposed a framework which extends the dropout approach by propagating the uncertainty that is produced from the sensors through the activation functions without the need of retraining. Recently, both uncertainty types were applied to 3D point cloud object detection [25] and optical flow estimation [26] tasks. To the best of our knowledge, BNNs have not been employed in modeling the uncertainty of semantic segmentation of 3D point clouds, which is one of the main contributions in this work.

## III. METHOD

*SalsaNext* is built upon the base *SalsaNet* model [1] which follows the standard encoder-decoder architecture with a bottleneck compression rate of 16. The original *SalsaNet* encoder contains a series of ResNet blocks each of which is followed by dropout and downsampling layers. The decoder blocks apply transpose convolutions and fuse upsampled features with that of the early residual blocks via skip connections. To further exploit descriptive spatial cues, a stack of convolution is inserted after the skip connection. We, in this study, improve the base structure of *SalsaNet* with the following contributions:

**Point Cloud Representation**: We project the unstructed 3D LiDAR point cloud onto a spherical surface to generate the LIDAR's native Range View (RV) image. This leads to dense and compact representation which allows standard convolution operations. Following the work of [20], we considered the full $360^\circ$ field-of-view in the projection process. During the projection, 3D point coordinates $(x, y, z)$, the intensity value $(i)$ and the range index $(r)$ are stored as separate RV image channels. This yields a $[w \times h \times 5]$ image.

**Contextual Module**: The global context information gathered by larger receptive fields plays a crucial role in learning complex correlations between classes [29]. To aggregate the context information in different regions, we place a residual dilated convolution stack that fuses a larger receptive field with a smaller one by adding $1 \times 1$ and $3 \times 3$ kernels right at the beginning of the network. This helps us capture the global context alongside with more detailed spatial information.

**Dilated Convolution**: Receptive fields play a crucial role in extracting spatial features. A straightforward approach to capture more descriptive spatial features would be to enlarge the kernel size. This has, however, a drawback of increasing the number of parameters drastically. Instead, we replace the ResNet blocks in the original *SalsaNet* encoder with a novel combination of a set of dilated convolutions having effective receptive fields of $3, 5$ and $7$. We further concatenate each dilated convolution output and apply a $1 \times 1$ convolution followed by a residual connection in order to let the network exploit more information from the fused features coming from various depths in the receptive field. Each of these new residual dilated convolution blocks is followed by dropout and pooling layers.

**Pixel-Shuffle Layer**: The original *SalsaNet* decoder involves transpose convolutions which are computationally expensive layers in terms of number of parameters. We replace these standard transpose convolutions with the *pixel-shuffle* layer [4] which leverages on the learnt feature maps to produce the upsampled feature maps by shuffling the pixels from the channel dimension to the spatial dimension. More precisely, the *pixel-shuffle* operator reshapes the elements of $(H \times W \times Cr^2)$ feature map to a form of $(Hr \times Wr \times C)$, where $H, W, C$, and $r$ represent the height, width, channel number and upscaling ratio, respectively. We additionally double the filters in the decoder side and concatenate the *pixel-shuffle* outputs with the skip connection before feeding them to the additional dilated convolutional blocks.

**Central Encoder-Decoder Dropout**: Lower network layers extract basic features such as edges and corners which are consistent over the data distribution and dropping out these layers will prevent the network to properly form the higher level features in the deeper layers. We, therefore, insert dropout only to the central encoder and decoder layers

| | Approach | Size | car | bicycle | motorcycle | truck | other-vehicle | person | bicyclist | motorcyclist | road | parking | sidewalk | other-ground | building | fence | vegetation | trunk | terrain | pole | traffic-sign | **mean-IoU** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Point-wise* | Pointnet [6] | 50K pts | 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 |
| | Pointnet++ [7] | | 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 |
| | TangentConv [27] | | 86.8 | 1.3 | 12.7 | 11.6 | 10.2 | 17.1 | 20.2 | 0.5 | 82.9 | 15.2 | 61.7 | 9.0 | 82.8 | 44.2 | 75.5 | 42.5 | 55.5 | 30.2 | 22.2 | 35.9 |
| | RandLa-Net [28] | | **94.2** | 26.0 | 25.8 | **40.1** | **38.9** | 49.2 | 48.2 | 7.2 | 90.7 | 60.3 | 73.7 | **38.9** | 86.9 | 56.3 | 81.4 | 61.3 | **66.8** | 49.2 | 47.7 | 53.9 |
| | LatticeNet [14] | | 92.9 | 16.6 | 22.2 | 26.6 | 21.4 | 35.6 | 43.0 | **46.0** | 90.0 | 59.4 | 74.1 | 22.0 | 88.2 | 58.8 | 81.7 | **63.6** | 63.1 | 51.9 | 48.4 | 52.9 |
| *Projection-based* | SqueezeSeg [15] | 64×2048 pixels | 68.8 | 16.0 | 4.1 | 3.3 | 3.6 | 12.9 | 13.1 | 0.9 | 85.4 | 26.9 | 54.3 | 4.5 | 57.4 | 29.0 | 60.0 | 24.3 | 53.7 | 17.5 | 24.5 | 29.5 |
| | SqueezeSegV2 [16] | | 81.8 | 18.5 | 17.9 | 13.4 | 14.0 | 20.1 | 25.1 | 3.9 | 88.6 | 45.8 | 67.6 | 17.7 | 73.7 | 41.1 | 71.8 | 35.8 | 60.2 | 20.2 | 36.3 | 39.7 |
| | RangeNet53++ [20] | | 91.4 | 25.7 | 34.4 | 25.7 | 23.0 | 38.3 | 38.8 | 4.8 | **91.8** | **65.0** | 75.2 | 27.8 | 87.4 | 58.6 | 80.5 | 55.1 | 64.6 | 47.9 | 55.9 | 52.2 |
| | 3D-MiniNet [21] | | 90.5 | 42.3 | **42.1** | 28.5 | 29.4 | 47.8 | 44.1 | 14.5 | 91.6 | 64.2 | 74.5 | 25.4 | 89.4 | 60.8 | **82.8** | 60.8 | 66.7 | 48.0 | 56.6 | 55.8 |
| | SqueezeSegV3 [17] | | 92.5 | 38.7 | 36.5 | 29.6 | 33.0 | 45.6 | 46.2 | 20.1 | 91.7 | 63.4 | 74.8 | 26.4 | 89.0 | 59.4 | 82.0 | 58.7 | 65.4 | 49.6 | 58.9 | 55.9 |
| | SalsaNet [1] | 64×2048 pixels | 87.5 | 26.2 | 24.6 | 24.0 | 17.5 | 33.2 | 31.1 | 8.4 | 89.7 | 51.7 | 70.7 | 19.7 | 82.8 | 48.0 | 73.0 | 40.0 | 61.7 | 31.3 | 41.9 | 45.4 |
| | SalsaNext [Ours] | | 91.9 | **48.3** | 38.6 | 38.9 | 31.9 | **60.2** | **59.0** | 19.4 | 91.7 | 63.7 | **75.8** | 29.1 | **90.2** | **64.2** | 81.8 | **63.6** | 66.5 | **54.3** | **62.1** | **59.5** |

TABLE I

QUANTITATIVE COMPARISON ON SEMANTIC-KITTI TEST SET (SEQUENCES 11 TO 21). IOU SCORES ARE GIVEN IN PERCENTAGE (%).

which leads to higher network performance.

**Average Pooling**: In the base *SalsaNet* model the downsampling was performed via a strided convolution which introduces additional learning parameters. Given that the down-sampling process is relatively straightforward, we hypothesize that learning at this level would not be needed. Thus, to allocate less memory *SalsaNext* switches to average pooling for the downsampling.

**Uncertainty Estimation**: In *SalsaNext,* the *epistemic* uncertainty is computed using the weight's posterior which is approximated by using dropout as shown in [22]. By following the work in [24], we compute the optimal dropout rate for an *already trained network* by applying a grid search on a log-range of a certain number of possible rates. To measure the *epistemic* uncertainty, we employ a Monte Carlo sampling during inference: we run $n$ trials with this optimal dropout rate and compute the average of the variance of the $n$ predicted outputs. To be able to track the *aleatoric* uncertainty, we propagate the known LiDAR noise characteristic through the network via Assumed Density Filtering (ADF) [30]. A forward pass in this ADF-based modified network finally generates output predictions with their respective aleatoric uncertainties [24].

**Loss**: To cope with the imbalanced class problem, we follow the same strategy in *SalsaNet* and add more value to the under-represented classes by weighting the softmax cross-entropy loss with the inverse square root of class frequency. This reinforces the network response to the classes appearing less in the dataset. In contrast to *SalsaNet,* we here also incorporate the *Lovász-Softmax* loss [2] in the learning procedure to maximize the intersection-over-union (IoU) score, i.e. the Jaccard index. The IoU metric is the most commonly used metric to evaluate the segmentation performance. Nevertheless, IoU is a discrete and not derivable metric that does not have a direct way to be employed as a loss. In [2], the authors adopt this metric with the help of the Lovász extension for submodular functions. Finally, the total loss function of *SalsaNext* is a linear combination of weighted cross-entropy and *Lovász-Softmax* losses.

**Optimizer and Regularization**: As an optimizer, we employed stochastic gradient descent with an initial learning rate of $0.01$ which is decayed by $0.01$ after each epoch. We also applied an L2 penalty with $\lambda = 0.0001$ and a momentum of $0.9$. The batch size and spatial dropout probability were fixed at $24$ and $0.2$, respectively. To prevent overfitting, we augmented the data by applying a random rotation/translation, flipping randomly around the y-axis and randomly dropping points before creating the projection. Every augmentation is applied independently of each other with a probability of $0.5$.

**Post-processing**: We further applied the kNN-based post-processing technique [20] to prevent the projection-based information loss when the RV image is re-projected back to the original 3D space.

## IV. EXPERIMENTS

We evaluate the performance of *SalsaNext* and compare with the state-of-the-art semantic segmentation methods on the large-scale challenging Semantic-KITTI dataset [3] which provides over 43K LiDAR data. Obtained quantitative results compared to state-of-the-art point-wise and projection-based approaches are reported in Table I. Our *SalsaNext* model considerably outperforms the others by leading to the highest mean IoU score ($59.5\%$) which is $+3.6\%$ over the previous state-of-the-art method [17]. In contrast to the original *SalsaNet,* we obtain $14\%$ improvement.

Following the work of [24], we further computed the *epistemic* and *aleatoric* uncertainty without retraining the *SalsaNext* model. Fig. 1 depicts the quantitative relationship between the *epistemic* (model) uncertainty and the number of points that each class has in the Semantic-KITTI test set. This plot has diagonally distributed samples, which clearly shows

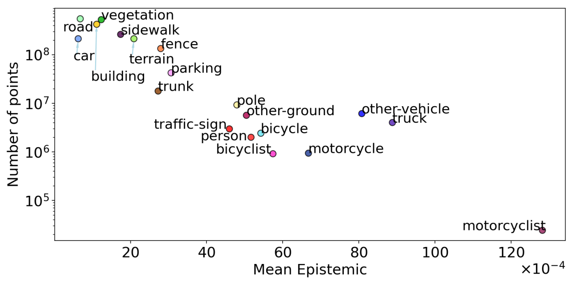

Fig. 1. The relationship between the *epistemic* uncertainty and the number of points (in log scale) in each class.

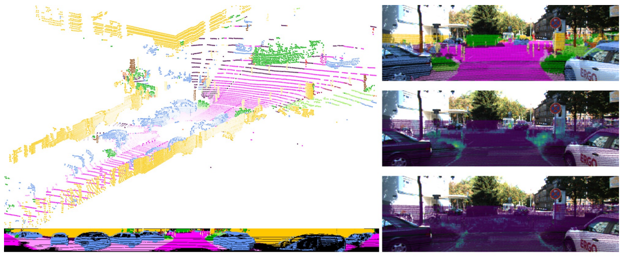

Fig. 2. A sample qualitative result. At the bottom, the range-view image of the network response is shown. The top camera image on the right shows the projected segments whereas the middle and bottom images depict the projected *epistemic* and *aleatoric* uncertainties, respectively. Note that the lighter the color is, the more uncertain the network becomes.

that the network becomes less certain about rare classes represented by low number of points (e.g. motorcyclist).

Fig. 2 shows sample qualitative segmentation and uncertainty results. In this figure, only for visualization purposes, segmented object points are also projected back to the respective camera image. Note that these camera images have not been used for training of *SalsaNext.* As depicted in Fig. 2, *SalsaNext* can, to a great extent, distinguish road, car, and other object points. In Fig. 2, we additionally show the estimated *epistemic* and *aleatoric* uncertainty values projected on the camera image for the sake of clarity. In line with Fig. 1, we obtain high *epistemic* uncertainty for rare classes such as other-ground (see Fig. 2). We also observe that high level of *aleatoric* uncertainty mainly appears around segment boundaries and on distant objects as shown in Fig. 2. In the supplementary video[2], we provide more qualitative results.

## V. Conclusion

We presented a new uncertainty-aware semantic segmentation network that can process the full 360° LiDAR scan in real-time. *SalsaNext* builds up on *SalsaNet* and can achieve over $14\%$ more accuracy. In contrast to state-of-the-art methods, *SalsaNext* returns $+3.6\%$ better mIoU score. *SalsaNext* can also estimate both data and model-based uncertainty.

## References

[1] E. E. Aksoy, S. Baci, and S. Cavdar, “Salsanet: Fast road and vehicle segmentation in lidar point clouds for autonomous driving,” in *IEEE Intelligent Vehicles Symposium (IV2020)*, 2020.

[2] M. Berman, A. Rannen Triki, and M. B. Blaschko, “The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks,” in *CVPR*, 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] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, “Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network,” *CoRR*, vol. abs/1609.05158, 2016.

[5] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, “Deep learning for 3d point clouds: A survey,” *CoRR*, 2019.

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

[7] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “Pointnet++: Deep hierarchical feature learning on point sets in a metric space,” in *NIPS*, 2017.

[8] L. Landrieu and M. Simonovsky, “Large-scale point cloud semantic segmentation with superpoint graphs,” in *CVPR*, 2018.

[9] C. Zhang, W. Luo, and R. Urtasun, “Efficient convolutions for real-time semantic segmentation of 3d point clouds,” in *3DV*, 2018.

[10] Y. Zhou and O. Tuzel, “Voxelnet: End-to-end learning for point cloud based 3d object detection,” in *CVPR*, 2018.

[11] L. P. Tchapmi, C. B. Choy, I. Armeni, J. Gwak, and S. Savarese, “Segcloud: Semantic segmentation of 3d point clouds,” in *3DV*, 2017.

[12] F. J. Lawin, M. Danelljan, P. Tosteberg, G. Bhat, F. S. Khan, and M. Felsberg, “Deep projective 3d semantic segmentation,” *CoRR*, 2017. [Online]. Available: http://arxiv.org/abs/1705.03428

[13] H. Su, V. Jampani, D. Sun, S. Maji, E. Kalogerakis, M. Yang, and J. Kautz, “Splatnet: Sparse lattice networks for point cloud processing,” in *CVPR*, 2018.

[14] R. Alexandru Rosu, P. Schütt, J. Quenzel, and S. Behnke, “LatticeNet: Fast Point Cloud Segmentation Using Permutohedral Lattices,” *arXiv e-prints*, p. arXiv:1912.05905, Dec. 2019.

[15] 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,” *ICRA*, 2018.

[16] B. Wu, X. Zhou, S. Zhao, X. Yue, and K. Keutzer, “Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud,” in *ICRA*, 2019.

[17] C. Xu, B. Wu, Z. Wang, W. Zhan, P. Vajda, K. Keutzer, and M. Tomizuka, “Squeezesegv3: Spatially-adaptive convolution for efficient point-cloud segmentation,” 2020.

[18] Y. Zeng, Y. Hu, S. Liu, J. Ye, Y. Han, X. Li, and N. Sun, “Rt3d: Real-time 3-d vehicle detection in lidar point cloud for autonomous driving,” *IEEE RAL*, vol. 3, no. 4, pp. 3434–3440, Oct 2018.

[19] M. Simon, S. Milz, K. Amende, and H. Gross, “Complex-yolo: Real-time 3d object detection on point clouds,” *CoRR*, 2018.

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

[21] I. Alonso, L. Riazuelo, L. Montesano, and A. C. Murillo, “3d-mininet: Learning a 2d representation from point clouds for fast and efficient 3d lidar semantic segmentation,” 2020.

[22] Y. Gal and Z. Ghahramani, “Dropout as a bayesian approximation: Representing model uncertainty in deep learning,” in *ICML*, 2016.

[23] A. Kendall, V. Badrinarayanan, and R. Cipolla, “Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding,” *arXiv preprint arXiv:1511.02680*, 2015.

[24] A. Loquercio, M. Segú, and D. Scaramuzza, “A general framework for uncertainty estimation in deep learning,” *RA-L*, 2020.

[25] D. Feng, L. Rosenbaum, and K. Dietmayer, “Towards safe autonomous driving: Capture uncertainty in the deep neural network for lidar 3d vehicle detection,” in *ITSC*. IEEE, 2018, pp. 3266–3273.

[26] E. Ilg, O. Cicek, S. Galesso, A. Klein, O. Makansi, F. Hutter, and T. Brox, “Uncertainty estimates and multi-hypotheses networks for optical flow,” in *ECCV*, 2018, pp. 652–667.

[27] M. Tatarchenko, J. Park, V. Koltun, and Q. Zhou, “Tangent convolutions for dense prediction in 3d,” in *CVPR*, 2018.

[28] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, “Randla-net: Efficient semantic segmentation of large-scale point clouds,” 2019.

[29] R. P. K. Poudel, S. Liwicki, and R. Cipolla, “Fast-scnn: Fast semantic segmentation network,” *CoRR*, vol. abs/1902.04502, 2019.

[30] J. Gast and S. Roth, “Lightweight probabilistic deep networks,” in *CVPR*, 2018, pp. 3369–3378.

[31] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in *CVPR*, 2012.

[2]https://youtu.be/MlSaIcD9ItU

## VI. Supplementary Material

### A. Ablation Study

In this ablative analysis, we investigate the individual contribution of each improvements over the original *SalsaNet* model. Table II shows the total number of model parameters and FLOPs (Floating Point Operations) with the obtained mIoU scores on the Semantic-KITTI test set before and after applying the kNN-based post processing.

As depicted in Table II, each of our contributions on *SalsaNet* has a unique improvement in the accuracy. The post processing step leads to a certain jump (around 2%) in the accuracy. The peak in the model parameters is observed when dilated convolution stack is introduced in the encoder, which is vastly reduced after adding the *pixel-shuffle* layers in the decoder. Combining the weighted cross-entropy loss with *Lovász-Softmax* leads to the highest increment in the accuracy as the Jaccard index is directly optimized. We can achieve the highest accuracy score of 59.5% by having only 2.2% (i.e. 0.15M) extra parameters compared to the original *SalsaNet* model. Table II also shows that the number of FLOPs is correlated with the number of parameters. We note that adding the *epistemic* and *aleatoric* uncertainty computations do not introduce any additional training parameter since they are computed after the network is trained.

### B. Runtime Evaluation

Runtime performance is of utmost importance in autonomous driving. Table III reports the total runtime performance for the CNN backbone network and post-processing module of *SalsaNext* in contrast to other networks. To obtain fair statistics, all measurements are performed using the entire Semantic-KITTI dataset on the same single NVIDIA Quadro RTX 6000 - 24GB card. As depicted in Table III, our method clearly exhibits better performance compared to, for instance, RangeNet++ [20] while having 7× less parameters. *SalsaNext* can run at 24 Hz when the uncertainty computation is excluded for a fair comparison with deterministic models. Note that this high speed we reach is significantly faster than the sampling rate of mainstream LiDAR sensors which typically work at 10 Hz [31]. Fig. 3 also compares the overall performance of *SalsaNext* with the other state-of-the-art semantic segmentation networks in terms of runtime, accuracy, and memory consumption.

As illustrated in Fig. 3, there is a clear split between projection-based and point-wise networks in terms of accuracy, runtime and memory consumption. For instance, projection-based approaches (shown in green circles in Fig. 3) achieve the state-of-the-art accuracy while running significantly faster. Although point-wise networks (red squares) have slightly lower number of parameters, they cannot efficiently scale up to large point sets due to the limited processing capacity, thus, they take a longer runtime. *SalsaNext* falls into the projection-based networks and achieves the highest score while achieving real-time performance with relatively low number of parameters. It is also highly important to note that unlike *SalsaNext,* both point-wise and projection-based approaches in Fig. 3 lack uncertainty measures, i.e. confidence scores, for their predictions.

| | mean IoU (w/o kNN) | mean IoU (+kNN) | Number of Parameters | FLOPs |
|---|---|---|---|---|
| SalsaNet [1] | 43.5 | 44.8 | 6.58 M | 51.60 G |
| + context module | 44.7 | 46.0 | 6.64 M | 69.20 G |
| + central dropout | 44.6 | 46.3 | 6.64 M | 69.20 G |
| + average pooling | 47.7 | 49.9 | 5.85 M | 66.78 G |
| + dilated convolution | 48.2 | 50.4 | 9.25 M | 161.60 G |
| + Pixel-Shuffle | 50.4 | 53.0 | 6.73 M | 125.68 G |
| + *Lovász-Softmax* loss | **56.6** | **59.5** | 6.73 M | 125.68 G |

TABLE II

Ablative analysis.

| | Processing Time (msec) | | | | | |
|---|---|---|---|---|---|---|
| | CNN | kNN | Total | Speed (fps) | Parameters | FLOPs |
| RangeNet++ [20] | 63.51 | 2.89 | 66.41 | 15 Hz | 50 M | 720.96 G |
| SalsaNet [1] | 35.78 | 2.62 | 38.40 | 26 Hz | 6.58 M | 51.60 G |
| SalsaNext [Ours] | 38.61 | 2.65 | 41.26 | 24 Hz | 6.73 M | 125.68 G |

TABLE III

Runtime performance on the Semantic-KITTI test set

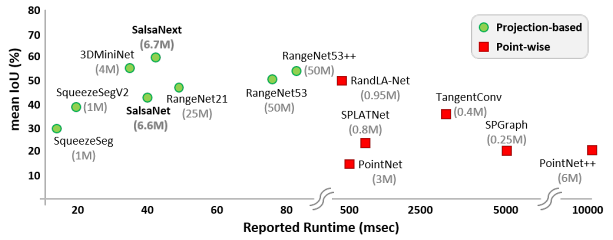

Fig. 3. Mean IoU versus runtime plot for the state-of-the-art 3D point cloud semantic segmentation networks on the Semantic-KITTI dataset [3]. Inside parentheses are given the total number of network parameters in Millions. All deep networks visualized here use only 3D LiDAR point cloud data as input. Note that only the published methods are considered.