EasyChair Preprint

№ 4398

SalsaNext: Fast, Uncertainty-Aware Semantic Segmentation of LiDAR Point Clouds

Tiago Cortinhal, George Tzelepis and Eren Aksoy

October 14, 2020

SalsaNext: Fast, Uncertainty-aware Semantic Segmentation of LiDAR Point Clouds

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

[1] Halmstad University, School of Information Technology, Halmstad, Sweden
[2] Volvo Technology AB, Volvo Group Trucks Technology, Gothenburg, Sweden

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 where the encoder unit has a set of ResNet blocks and the decoder part combines upsampled features from the residual 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 [4]. We finally inject a Bayesian treatment to compute the *epistemic* and *aleatoric* uncertainties for each point in the cloud. We provide a thorough quantitative evaluation on the Semantic-KITTI dataset [3], which demonstrates that the proposed *SalsaNext* outperforms other published semantic segmentation networks and achieves 3.6% more accuracy over the previous state-of-the-art method. We also release our source code [1].

Keywords: Semantic Segmentation · LiDAR Point Clouds · Deep Learning.

1 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. Achieving such a fine-grained semantic prediction in real-time accelerates reaching the full autonomy to a great extent.

Safety-critical systems, such as self-driving vehicles, however, require not only highly accurate but also reliable predictions 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.

Advanced deep neural networks recently had a quantum jump in generating accurate and reliable semantic segmentation with real-time performance. Most of these

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

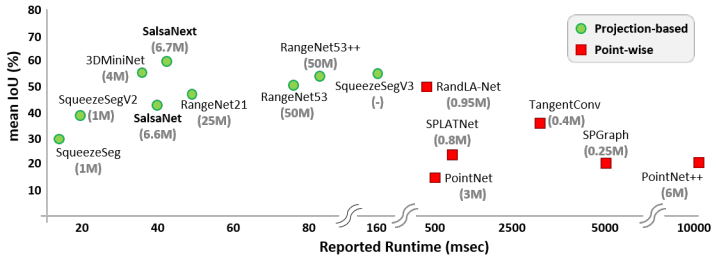

Fig. 1. Mean IoU versus runtime plot for the state-of-the-art 3D point cloud semantic segmentation networks on the Semantic-KITTI dataset [3]. Inside parentheses are given the total number of network parameters in Millions. All deep networks visualized here use only 3D LiDAR point cloud data as input. Note that only the published methods are considered.

approaches, however, rely on the camera images [13], whereas relatively fewer contributions have discussed the semantic segmentation of 3D LiDAR data [27,19]. The main reason is that unlike camera images, LiDAR point clouds are relatively sparse, unstructured, and have non-uniform sampling, although LiDAR scanners have a wider field of view and return more accurate distance measurements.

As comprehensively described in [9], there exists two mainstream deep learning approaches addressing the semantic segmentation of 3D LiDAR data only: point-wise and projection-based neural networks (see Fig. 1). The former approach operates directly on the raw 3D points without requiring any pre-processing step, whereas the latter projects the point cloud into various formats such as 2D image view or high-dimensional volumetric representation. As illustrated in Fig. 1, there is a clear split between these two approaches in terms of accuracy, runtime and memory consumption. Projection-based approaches (shown in green circles in Fig. 1) 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. Note also that both point-wise and projection-based approaches in the literature lack uncertainty measures, i.e. confidence scores, for their predictions.

We here 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 *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 the proposed *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 has 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 further concatenated the convolution outputs and combined with residual connections yielding a branch-like structure.
- To avoid any checkerboard artifacts in upsampling, we replaced the transposed convolution layer in the *SalsaNet* decoder with a *pixel-shuffle* layer [24] 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 stride convolutions in the encoder.
- To enhance the segmentation accuracy by optimizing the mean intersection-over-union score, i.e. the Jaccard index, the weighted cross entropy loss in *SalsaNet* was combined with the *Lovász-Softmax* loss [4].
- 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.

The input of *SalsaNext* is the rasterized image of the full LiDAR scan, where each image channel stores position, depth, and intensity cues in the panoramic view format. 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. Computing both uncertainties is of utmost importance in safe autonomous driving since the *epistemic* uncertainty can indicate the limitation of the segmentation model while the *aleatoric* one highlights the sensor observation noises for segmentation.

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

2 Related Work

Regarding the processing of unstructured 3D LiDAR points, there are two common methods as depicted in Fig. 1: point-wise representation and projection-based rendering. We refer the interested readers to [9] for more details.

Point-wise methods [20,21] directly process the raw irregular 3D points without applying any additional transformation or pre-processing. Shared multi-layer perceptron-based PointNet [20], the subsequent work PointNet++ [21], and *superpoint* graph SPG networks [14] 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° LiDAR scans.

Projection-based methods instead transform the 3D point cloud into various formats such as voxel cells [32], multi-view representation [15], lattice structure [25,23], and rasterized images [1,27,28,29]. In the multi-view representation, a 3D point cloud is projected onto multiple 2D surfaces from various virtual camera viewpoints. Each view

is then processed by a multi-stream network as in [15]. In the lattice structure, the raw unorganized point cloud is interpolated to a permutohedral sparse lattice where bilateral convolutions are applied to occupied lattice sectors only [25]. Methods relying on the voxel representation discretize the 3D space into 3D volumetric space (i.e. voxels) and assign each point to the corresponding voxel [32]. Sparsity and irregularity in point clouds, however, yield redundant computations in voxelized data since many voxel cells may stay empty. A common attempt to overcome the sparsity in LiDAR data is to project 3D point clouds into 2D image space either in the top-down [1,30] or spherical Range-View (RV) (i.e. panoramic view) [2,19,27,28,29] formats. 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 convolutional layers. Therefore, our *SalsaNext* model projects the LiDAR point cloud into 2D RV image generated by mapping each 3D point onto a spherical surface.

When it comes to the uncertainty estimation, Bayesian Neural Networks (BNNs) are the dominant approach. BNNs learn approximate distribution on the weights to further generate uncertainty estimates, i.e. prediction confidences. 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. Unlike *aleatoric* uncertainty, which captures the irreducible noise in the data, *epistemic* uncertainty can be reduced by gathering more training data. For instance, segmenting out an object that has relatively fewer training samples in the dataset may lead to high *epistemic* uncertainty, whereas high *aleatoric* uncertainty may rather occur on segment boundaries or distant and occluded objects due to noisy sensor readings inherent in sensors. Bayesian modelling helps estimating both uncertainties.

Gal *et al.* [7] 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.* [13]. Loquercio *et al.* [18] 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 [6] and optical flow estimation [12] tasks. To the best of our knowledge, BNNs have not been employed in modeling the uncertainty of semantic segmentation of 3D LiDAR point clouds, which is one of the main contributions in this work.

In this context, the closest work to ours is [31] which introduces a probabilistic embedding space for point cloud instance segmentation. This approach, however, captures neither the aleatoric nor the epistemic uncertainty but rather predicts the uncertainty between the point cloud embeddings. Unlike our method, it has also not been shown how the aforementioned work can scale up to large and complex LiDAR point clouds.

3 Method

In this section, we give a detailed description of our method including the point cloud representation, network architecture, uncertainty estimation, and training details.

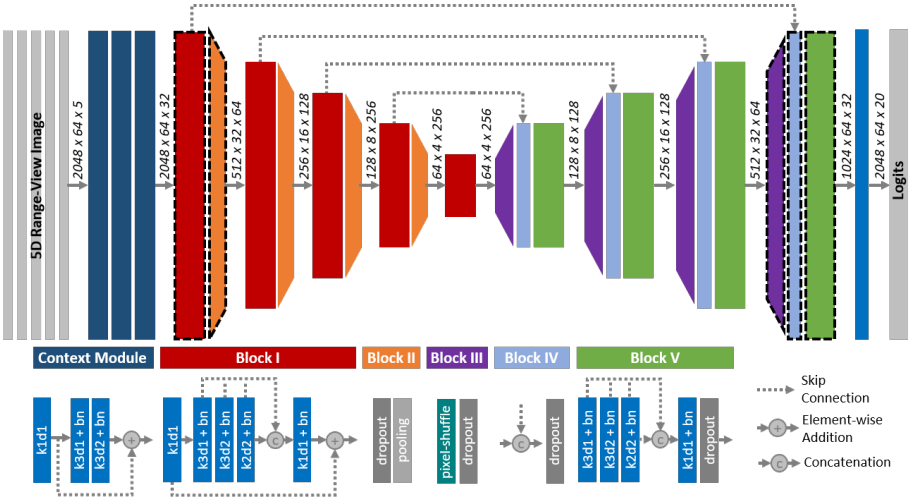

Fig. 2. Architecture of the proposed *SalsaNext* model. Blocks with dashed edges indicate those that do not employ the dropout. The layer elements k, d, and bn represent the kernel size, dilation rate and batch normalization, respectively.

3.1 LiDAR Point Cloud Representation

As in [19], we project the unstructed 3D LiDAR point cloud onto a spherical surface to generate the LIDAR's native Range View (RV) image. This process leads to dense and compact point cloud representation which allows standard convolution operations.

In the 2D RV image, each raw LiDAR point (x, y, z) is mapped to an image coordinate (u, v) as

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{2}[1 - \arctan(y, x)\pi^{-1}]w \\ [1 - (\arcsin(z, r^{-1}) + f_{down})f^{-1}]h \end{pmatrix} \quad ,$$

where h and w denote the height and width of the projected image, r represents the range of each point as $r = \sqrt{x^2 + y^2 + z^2}$ and f defines the sensor vertical field of view as $f = |f_{down}| + |f_{up}|$.

Following the work of [19], we considered the full 360° 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 to be fed to the network.

3.2 Network Architecture

The architecture of the proposed *SalsaNext* is illustrated in Fig. 2. The input to the network is an RV image projection of the point cloud as described in section 3.1.

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 [10] 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. As illustrated in Fig. 2, we in this study improve the base structure of *SalsaNet* with the following contributions:

Contextual Module: To aggregate the context information in different regions, we place a residual dilated convolution stack that fuses a small receptive field with a larger one right at the beginning of the network. More specifically, we have one 1×1 and two 3×3 kernels with $dilation\ rates = (1, 2)$, which are residually connected and fused by applying element-wise addition (see Fig. 2). Starting with relatively small 1×1 kernel helps aggregate channel-wise local spatial features while having 3×3 kernels with different dilation rates captures various complex correlations between different segment classes. This helps focusing on more contextual information alongside with more detailed global spatial information via pyramid pooling similar to [5].

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 (see Block I in Fig. 2). We further concatenate each dilated convolution output and apply a 1×1 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 (i.e. Block I) is followed by dropout and pooling layers (Block II in Fig. 2).

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 [24] (see Block III in Fig. 2) 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 (Block IV in Fig. 2) before feeding them to the dilated convolutional blocks (Block V in Fig. 2) in the decoder.

Central Encoder-Decoder Dropout: As quantitative experiments in [13] show, inserting dropout only to the central encoder and decoder layers results in better segmentation performance. It is because the 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. Central dropout approach eventually leads to higher network performance. We, therefore, insert dropout in every encoder-decoder layer except the first and last one highlighted by dashed edges in Fig. 2.

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.

All these contributions from the proposed *SalsaNext* network. Furthermore, we applied a 1×1 convolution after the decoder unit to make the channel numbers the same with the total number of semantic classes. The final feature map is finally passed to a soft-max classifier to compute pixel-wise classification scores. Note that each convolution layer in the *SalsaNext* model employs a leaky-ReLU activation function and is followed by batch normalization to solve the internal covariant shift. Dropout is then placed after the batch normalization. It can, otherwise, result in a shift in the weight distribution which can minimize the batch normalization effect during training [16].

3.3 Uncertainty Estimation

Heteroscedastic Aleatoric Uncertainty We can define *aleatoric* uncertainty as being of two kinds: *homoscedastic* and *heteroscedastic*. The former defines the type of *aleatoric* uncertainty that remains constant given different input types, whereas the later may rather differ for different types of input. In the LiDAR semantic segmentation task, distant points might introduce a *heteroscedastic* uncertainty as it is increasingly difficult to assign them to a single class. The same kind of uncertainty is also observable in the object edges when performing semantic segmentation, especially when the gradient between the object and the background is not sharp enough.

LiDAR observations are usually corrupted by noise and thus the input that a neural network is processing is a noisy version of the real world. Assuming that the sensor's noise characteristic is known (e.g. available in the sensor data sheet), the input data distribution can be expressed by the normal $\mathcal{N}(\mathbf{x}, \mathbf{v})$, where $\mathbf{x}$ represents the observations and $\mathbf{v}$ the sensor's noise. In this case, the aleatoric uncertainty can be computed by propagating the noise through the network via Assumed Density Filtering (ADF). This approach was initially applied by Gast *et al.* [8], where the network's activation functions including input and output were replaced by probability distributions. A forward pass in this ADF-based modified neural network finally generates output predictions μ with their respective aleatoric uncertainties σ_A.

Epistemic Uncertainty In *SalsaNext,* the *epistemic* uncertainty is computed using the weight's posterior $p(\mathbf{W}|\mathbf{X}, \mathbf{Y})$ which is intractable and thus impossible to present analytically. However, the work in [7] showed that dropout can be used as an approximation to the intractable posterior. More specifically, dropout is an approximating distribution $q_\theta(\omega)$ to the posterior in a BNN with L layers, $\omega = [\mathbf{W}_l]_{l=1}^{L}$ where θ is a set of variational parameters. The optimization objective function can be written as:

$$\hat{\mathcal{L}}_{MC}(\theta) = -\frac{1}{M} \sum_{i \in S} \log p(y_i | f^{\omega}(x_i)) + \frac{1}{N} \mathbf{KL}(q_\theta || p(\omega))$$

where the $\mathbf{KL}$ denotes the regularization from the Kullback-Leibler divergence, N is the number of data samples, S holds a random set of M data samples, y_i denotes the ground-truth, $f^{\omega}(x_i)$ is the output of the network for x_i input with weight parameters ω and $p(y_i|f^{\omega}(x_i))$ likelihood. The $\mathbf{KL}$ term can be approximated as:

$$KL(q_M(\mathbf{W})||p(\mathbf{W})) \propto \frac{i^2(1-p)}{2}||\mathbf{M}||^2 - K\mathcal{H}(p)$$

where

$$\mathcal{H}(p) := -p\log(p) - (1-p)\log(1-p)$$

represents the entropy of a Bernoulli random variable with probability p and K is a constant to balance the regularization term with the predictive term.

For example, the negative log likelihood in this case will be estimated as

$$-\log p(y_i|f^{\omega}(x_i)) \propto \frac{1}{2}\log\sigma + \frac{1}{2\sigma}||y_i - f^{\omega}(x_i)||^2$$

for a Gaussian likelihood with σ model's uncertainty.

To be able to measure the *epistemic* uncertainty, we employ a Monte Carlo sampling during inference: we run n trials and compute the average of the variance of the n predicted outputs:

$$\text{Var}^{epistemic}_{p(y|f^{\omega}(x))} = \sigma_{epistemic} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y})^2 \quad .$$

As introduced in [18], the optimal dropout rate p which minimizes the $\mathbf{KL}$ divergence, is estimated for an *already trained network* by applying a grid search on a log-range of a certain number of possible rates in the range $[0, 1]$. In practice, it means that the optimal dropout rates p will minimize:

$$p = \arg\min_{\hat{p}} \sum_{d\in D} \frac{1}{2}\log(\sigma^d_{tot}) + \frac{1}{2\sigma^d_{tot}}(y^d - y^d_{pred}(\hat{p}))^2 \quad ,$$

where σ_{tot} denotes the total uncertainty by summing the aleatoric and the epistemic uncertainty, D is the input data, $y^d_{pred}(\hat{p})$ and y^d are the predictions and labels.

3.4 Loss Function

Datasets with imbalanced classes introduce a challenge for neural networks. Take an example of a bicycle or traffic sign which appears much less compared to the vehicles in the autonomous driving scenarios. This makes the network more biased towards to the classes that emerge more in the training data and thus yields significantly poor network performance.

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 $\mathcal{L}_{wce}$ with the inverse square root of class frequency as

$$\mathcal{L}_{wce}(y, \hat{y}) = -\textstyle\sum_i \alpha_i p(y_i) log(p(\hat{y}_i)) \quad with \quad \alpha_i = 1/\sqrt{f_i} \quad ,$$

where y_i and $\hat{y}_i$ define the true and predicted class labels and f_i stands for the frequency, i.e. the number of points, of the i^{th} class. 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 [4] in the learning procedure to maximize the intersection-over-union (IoU) score, i.e. the Jaccard index. The IoU metric (see section 4) 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 [4], the authors adopt this metric with the help of the Lovász extension for submodular functions. Considering the IoU as a hypercube where each vertex is a possible combination of the class labels, we relax the IoU score to be defined everywhere inside of the hypercube. In this respect, the *Lovász-Softmax* loss ($\mathcal{L}_{ls}$) can be formulated as follows:

$$\mathcal{L}_{ls} = \frac{1}{|C|} \sum_{c \in C} \overline{\Delta_{J_c}}(m(c)) \ , \ \ and \ \ m_i(c) = \begin{cases} 1 - x_i(c) & \text{if} \ \ c = y_i(c) \\ x_i(c) & \text{otherwise} \end{cases} \ ,$$

where $|C|$ represents the class number, $\overline{\Delta_{J_c}}$ defines the Lovász extension of the Jaccard index, $x_i(c) \in [0, 1]$ and $y_i(c) \in \{-1, 1\}$ hold the predicted probability and ground truth label of pixel i for class c, respectively.

Finally, the total loss function of *SalsaNext* is a linear combination of both weighted cross-entropy and *Lovász-Softmax* losses as follows: $\mathcal{L} = \mathcal{L}_{wce} + \mathcal{L}_{ls}$.

3.5 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. To prevent overfitting, we augmented the data by applying a random rotation and 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.

3.6 Post-processing

The main drawback of the projection-based point cloud representation is the information loss due to discretization errors and blurry convolutional layer responses. This problem emerges when, for instance, the RV image is re-projected back to the original 3D space. The reason is that during the image rendering process, multiple LiDAR points may get assigned to the very same image pixel which leads to misclassification of, in particular, the object edges. This effect becomes more obvious, for instance, when the objects cast a shadow in the background scene.

To cope with these back-projection related issues, we employ the kNN-based post-processing technique introduced in [19]. The post-processing is applied to every LIDAR point by using a window around each corresponding image pixel, that will be translated into a subset of point clouds. Next, a set of closest neighbors is selected with the help of kNN. The assumption behind using the range instead of the Euclidian distances lies in the fact that a small window is applied, making the range of close (u, v) points serve as a good proxy for the Euclidian distance in the 3D space.

Table 1. Quantitative comparison on Semantic-KITTI test set (sequences 11 to 21). IoU scores are given in percentage (%). Note that only the published methods are considered.

	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 [20]	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++ [21]		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
	SPGraph [14]		68:3	0.9	4.5	0.9	0.8	1.0	6.0	0.0	49.5	1.7	24.2	0.3	68.2	22.5	59.2	27.2	17.0	18.3	10.5	20.0
	SPLATNet [25]		66.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	70.4	0.8	41.5	0.0	68.7	27.8	72.3	35.9	35.8	13.8	0.0	22.8
	TangentConv [26]		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 [11]		**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 [23]		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 [27]	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
	SqueezeSeg-CRF [27]		68.3	18.1	5.1	4.1	4.8	16.5	17.3	1.2	84.9	28.4	54.7	4.6	61.5	29.2	59.6	25.5	54.7	11.2	36.3	30.8
	SqueezeSegV2 [28]		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
	SqueezeSegV2-CRF [28]		82.7	21.0	22.6	14.5	15.9	20.2	24.3	2.9	88.5	42.4	65.5	18.7	73.8	41.0	68.5	36.9	58.9	12.9	41.0	39.6
	RangeNet21 [19]		85.4	26.2	26.5	18.6	15.6	31.8	33.6	4.0	91.4	57.0	74.0	26.4	81.9	52.3	77.6	48.4	63.6	36.0	50.0	47.4
	RangeNet53 [19]		86.4	24.5	32.7	25.5	22.6	36.2	33.6	4.7	**91.8**	64.8	74.6	27.9	84.1	55.0	78.3	50.1	64.0	38.9	52.2	49.9
	RangeNet53++ [19]		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 [2]		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 [29]		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**

4 Experiments

We evaluate the performance of *SalsaNext* and compare with the other state-of-the-art semantic segmentation methods on the large-scale challenging Semantic-KITTI dataset [3] which provides over 43K point-wise annotated full 3D LiDAR scans. We follow exactly the same protocol in [19] and divide the dataset into training, validation, and test splits. Over 21K scans (sequences between 00 and 10) are used for training, where scans from sequence 08 are particularly dedicated to validation. The remaining scans (between sequences 11 and 21) are used as test split. The dataset has in total 22 classes 19 of which are evaluated on the test set by the official online benchmark platform. We implement our model in PyTorch and release the code for public use [2].

To evaluate the results, we use the Jaccard Index, i.e. mean intersection-over-union (IoU) over all classes given by $mIoU = \frac{1}{C}\sum_{i=1}^{C}\frac{|\mathcal{P}_i \cap \mathcal{G}_i|}{|\mathcal{P}_i \cup \mathcal{G}_i|}$, where $\mathcal{P}_i$ is the set of point with a class prediction i, $\mathcal{G}_i$ the labelled set for class i and $|\cdot|$ the cardinality of the set.

4.1 Quantitative and Qualitative Results

Table 1 reports obtained quantitative results compared to other published point-wise and projection-based approaches. Our proposed model *SalsaNext* 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 [29]. In contrast to the original *SalsaNet,* we also obtain more than 14% improvement in the accuracy. When it comes to the performance of each individual category, *SalsaNext* performs the best in 9 out of 19 categories. Note that in most of these remaining 10 categories (e.g. road, vegetation, and terrain) *SalsaNext* has a comparable performance with the other approaches.

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

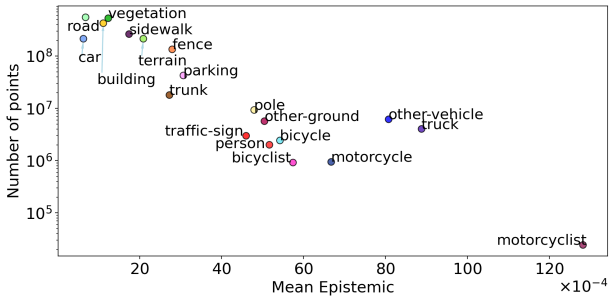

Fig. 3. The relationship between the *epistemic* (model) uncertainty and the number of points (in log scale) that each class has in the entire test dataset.

Following the work of [18], we further computed the *epistemic* and *aleatoric* uncertainty without retraining *SalsaNext* (see sec. 3.3). Fig. 3 depicts the quantitative relationship between the *epistemic* (model) uncertainty and the number of points that each class has in the entire Semantic-KITTI test dataset. This plot has diagonally distributed samples, which clearly shows that the network becomes less certain about rare classes represented by low number of points (e.g. motorcyclist and motorcycle). There is, to some degree, an inverse correlation between the obtained uncertainty and the segmentation accuracy: when the network predicts an incorrect label, the uncertainty becomes high as in the case of motorcyclist which has the lowest IoU score (19.4%) in Table 1.

For the qualitative evaluation, Fig. 4 shows some sample semantic segmentation and uncertainty results generated by *SalsaNext* on the Semantic-KITTI test set. In this

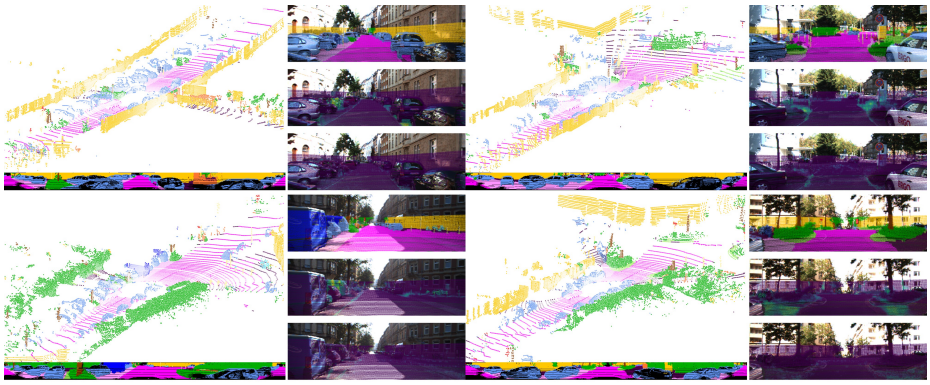

Fig. 4. Sample qualitative results of *SalsaNext* [best view in color]. At the bottom of each scene, the range-view image of the network response is shown. Note that the corresponding camera images on the right are only for visualization purposes and have not been used in the training. The top camera 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.

figure, only for visualization purposes, segmented object points are also projected back to the respective camera image. We, here, emphasize that these camera images have not been used for training of *SalsaNext.* As depicted in Fig. 4, *SalsaNext* can, to a great extent, distinguish road, car, and other object points. In Fig. 4, we additionally show the estimated *epistemic* and *aleatoric* uncertainty values projected on the camera image for the sake of clarity. Here, the light blue points indicate the highest uncertainty whereas darker points represent more certain predictions. In line with Fig. 3, we obtain high *epistemic* uncertainty for rare classes such as other-ground as shown in the last frame in Fig. 4. We also observe that high level of *aleatoric* uncertainty mainly appears around segment boundaries (see the second frame in Fig. 4) and on distant objects (e.g. last frame in Fig. 4). In the supplementary video [3], we provide more qualitative results.

4.2 Ablation Study

Table 2 shows the total number of model parameters and FLOPs (Floating Point Operations) with the obtained mIoU scores on the Semantic-KITTI validation set before and after applying the kNN-based post processing (see section 3.6). As depicted in this table, 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. Switching to the *pixel-shuffle* layers yields 1.0% more accuracy while having $2.52M$ less parameters and 22% less FLOPs. Recall that pixel shuffle is a differentiable process which rearranges elements from depth dimension to spatial domain in a deterministic way. Therefore, shuffling pixels in the decoder leads to more accurate image reconstruction as it introduces fewer checkerboard artifacts with a vastly reduced number of parameters. Combining the weighted cross-entropy loss with *Lovász-Softmax* leads to the highest increment in the accuracy. This is mainly because the Jaccard index which is the main metric to measure the segmentation accuracy is directly optimized as a part of the loss function. We can achieve the highest accuracy score of 59.9% by having only 2.2% (i.e. 0.15M) extra parameters compared to the original *SalsaNet* model. Table 2 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.

[3] https://www.youtube.com/watch?v=MlSaIcD9ItU

Table 2. Ablative analysis on the validation set

	mean IoU (w/o kNN)	mean IoU (+kNN)	Number of Parameters	FLOPs
SalsaNet [1]	43.2	44.4	6.58 M	51.60 G
+ context module	45.0	46.4	6.64 M	69.20 G
+ central dropout	48.5	50.8	6.64 M	69.20 G
+ average pooling	48.9	51.2	5.85 M	66.78 G
+ dilated convolution	50.6	52.3	9.25 M	161.60 G
+ Pixel-Shuffle	51.2	53.3	6.73 M	125.68 G
+ *Lovász-Softmax* loss	**56.4**	**59.9**	6.73 M	125.68 G

Table 3. Runtime performance on the Semantic-KITTI test set

	Processing Time (msec)					
	CNN	kNN	Total	Speed (fps)	Parameters	FLOPs
RangeNet++ [19]	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

4.3 Runtime Evaluation

Table 3 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. *SalsaNext* clearly exhibits better performance compared to RangeNet++ [19] while having $7\times$ less parameters. *SalsaNext* can run at 24 Hz when the uncertainty computation is excluded for a fair comparison with deterministic models. This achieved high speed is significantly faster than the sampling rate of mainstream LiDAR sensors which is typically 10 Hz. Fig. 1 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.

5 Conclusion

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

References

1. Aksoy, E.E., Baci, S, Cavdar, S.: Salsanet: Fast road and vehicle segmentation in lidar point clouds for autonomous driving. In *IEEE IV*, (2020).
2. Alonso, I., Riazuelo, L., Montesano, L., Murillo, A.C.: 3d-mininet: Learning a 2d representation from point clouds for fast and efficient 3d lidar semantic segmentation. *RA-L*, (2020).
3. Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., Gall, J.: SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *ICCV*, (2019).
4. Berman, M., Triki, A.R., Blaschko, M.: The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In *CVPR*, (2018).
5. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. In *arXiv*, (2017).
6. Feng, D., Rosenbaum, L., Dietmayer, K.: Towards safe autonomous driving: Capture uncertainty in the deep neural network for lidar 3d vehicle detection. In *ITSC*, (2018).
7. Gal, Y., Ghahramani, Z.: Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, (2016).

8. Gast, J., Roth, S.: Lightweight probabilistic deep networks. In *CVPR*, (2018).
9. Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., Bennamoun, M.: Deep learning for 3d point clouds: A survey. *IEEE TPAMI*, (2019).
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In *CVPR*, pages 770–778, (2016).
11. Hu, Q., Yang, B., Xie, L., Rosa, S., Guo, Y., Wang, Z., Trigoni, N., Markham, A.: Randla-net: Efficient semantic segmentation of large-scale point clouds. In *CVPR*, (2020).
12. Ilg, E., Cicek, O., Galesso, S., Klein, A., Makansi, O., Hutter, F., Brox, T., Uncertainty estimates and multi-hypotheses networks for optical flow. In *ECCV*, pages 652–667, (2018).
13. Kendall, A., Badrinarayanan, V., Cipolla, R.: Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. In *BMVC*, (2017).
14. Landrieu L., Simonovsky, M.: Large-scale point cloud semantic segmentation with superpoint graphs. In *CVPR*, (2018).
15. Lawin, F.J., Danelljan, M., Tosteberg, P., Bhat, G., Khan, F.S., Felsberg, M.: Deep projective 3d semantic segmentation. In *CAIP*, (2017).
16. Li, X., Chen, S., Hu, X., Yang, J.: Understanding the disharmony between dropout and batch normalization by variance shift. In *CVPR*, (2019).
17. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. *IEEE TPAMI*, 39(4):640–651, (2017).
18. Loquercio, A., Segu, M., Scaramuzza, D.: A general framework for uncertainty estimation in deep learning. *IEEE RA-L*, 5(2):3153–3160, (2020).
19. Milioto, A., Vizzo, I., Behley, J., Stachniss, C.: RangeNet++: Fast and Accurate LiDAR Semantic Segmentation. In *IROS*, (2019).
20. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, (2017).
21. Qi, C.R., Yi, L., Su, H., Guibas, L.j.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, (2017).
22. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241, (2015).
23. Rosu, A.R., Schütt, P., Quenzel, J., Behnke, S.: LatticeNet: Fast Point Cloud Segmentation Using Permutohedral Lattices. In *RSS*, (2020).
24. Shi, W., Caballero, J., Huszar, F., Totz, J., Aitken, A.P., Bishop, R., Rueckert, D., Wang, Z.: Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*, pages 1874–1883, (2016).
25. Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M.H., Kautz, J.: Splatnet: Sparse lattice networks for point cloud processing. In *CVPR*, (2018).
26. Tatarchenko, M., Park, J., Koltun, V., Zhou, Q.Y.: Tangent convolutions for dense prediction in 3d. In *CVPR*, (2018).
27. Wu, B., Wan, A., Yue, X., Keutzer, K.: Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *ICRA*, (2018).
28. Wu, B., Zhou, X., Zhao, S., Yue, X., Keutzer, K.: Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud. In *ICRA*, (2019).
29. Xu, C., Wu, B., Wang, Z., Zhan, W., Vajda, P., Keutzer, K. and Tomizuka, M.: Squeezesegv3: Spatially-adaptive convolution for efficient point-cloud segmentation. In *arXiv*, (2020).
30. Zeng, Y., Hu, Y., Liu, S., Ye, J., Han, Y., Li, X., Sun, N.: Rt3d: Real-time 3-d vehicle detection in lidar point cloud for autonomous driving. *IEEE RAL*, 3(4):3434–3440, (2018).
31. Zhang, B., Wonka, P.: Point cloud instance segmentation using probabilistic embeddings. In *CoRR*, (2019).
32. Zhou, Y., Tuzel, O.: Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, (2018).