Semantic Segmentation of LiDAR Point Cloud for Autonomous Vehicles

Aniket Manish Patil [1] **Bhushan Ashok Rane** [1] **Chinmay Madhukar Todankar** [1]

Abstract

Perception of the environment in Autonomous driving systems can be performed using a variety of sensors such as LiDAR, Camera and RADAR. This paper is focused on the implementation of RangeNet, an encoder-decoder based architecture, for semantic segmentation of a LiDAR Point Cloud. In addition to implementing the baseline RangeNet architecture from scratch, we tried a couple of modifications to the network.

First modification of adding a Pixel Shuffling layer successfully decreased the complexity of the network, but a drop in the performance was observed as well. The second modification was to combine output of past encoder before passing it to the decoder, to explore relation between consecutive inputs. The observation was that the baseline implementation performs better than these modifications, under the caveat that training was done on 15 epochs for all experiments due to processing and time constraints. **GitHub**: https://github.com/aniketmpatil/semantic_segmentation

1. Introduction

In the Autonomous Vehicles domain, the task of Perception is dependent on the high accuracy of the data provided by the sensors and thus processing that data to extract useful information. Two the highly used sources of information are images and point clouds. There are various ways to process images and point clouds, one of them is semantic segmentation. Semantic Segmentation is a process that labels each point or pixel with a class label corresponding to what is being represented.

In this paper, we focus on the implementation of semantic segmentation of the LiDAR point cloud on the Semantic KITTI dataset using the RangeNet architecture (Milioto et al., 2019) which uses the range image representation (spherical projection) of the point clouds. The architecture produces state-of-the-art results, but the reported run-time is significantly high. We first implemented the same architecture from scratch using the Tensorflow library. Inside this architecture, Pixel Shuffling layer was added and attempted to explore the effect of introducing temporal dependence in the architecture. This was done by using the encoder output from the previous step in the current decoder input.

1.1. Research contributions

As our contribution to the research community, we try to introduce some changes in the RangeNet architecture, and study their effects on the test accuracy. Some of the key contributions are:

1. Addition of a Pixel Shuffling Layer (first introduced in (Shi et al., 2016)) in the decoder, which will significantly reduce the number of trainable parameters.

2. Exploring a modification to see if previous encoder output can have a impact on the segmentation task. The motivation behind this is that, we know that an object will be in same general location in consecutive frames.

2. Related Work

There are multiple ways to tackle this problem of preprocessing a LiDAR point cloud and segmenting it that have been studied over the past years.
SqueezeSegv1(Wu et al., 2018) uses the spherical projection of LiDAR point cloud from the Semantic KITTI dataset (Behley et al., 2019) enabling the usage of 2D convolutions. Furthermore, it outputs a point-wise segmentation label map, refined by a Conditional Random Field (CRF) implemented as a recurrent layer and finally the last step is un-discretization of the points from the range image back into the 3D world.
SqueezeSegv2 (Wu et al., 2019) with improved model structure, is more robust and certainly performs better because of Context Aggregation Module (CAM) used to combat dropout noise in LiDAR point cloud. Both lightweight convolutional neural networks are capable of running faster than normal sensor rate.
SqueezeSegv3 (Xu et al., 2020) outperforms previous published methods on Semantic KITTI benchmark with a com-

[1]Worcester Polytechnic Institute. Correspondence to: Jacob Whitehill <jrwhitehill@wpi.edu>.

CS/DS 541 Deep Learning Final Project, Worcester Polytechnic Institute.

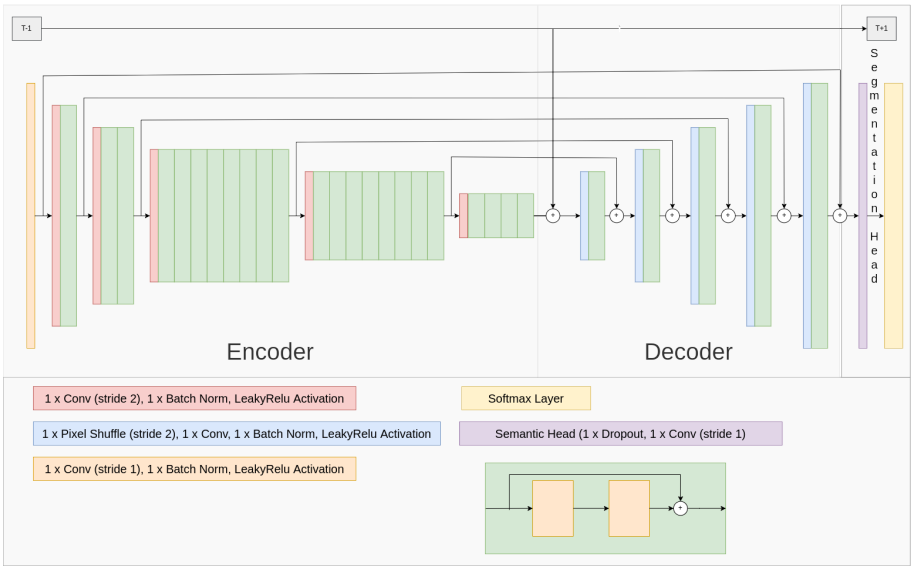

Figure 1. Architecture

parable inference speed. To tackle the problem of standard convolution filters picking up local features that are only active in specific region in the images, SqueezeSegv3 uses Spatially-Adaptive Convolutions (SAC) to adopt different filters for different locations according to input images.
SalsaNext (Cortinhal et al., 2020) introduces a new context module, replacing the ResNet encoder blocks, with a new residual dilated convolution stack with increasing receptive fields and adds a pixel shuffle layer in the decoder. Additionally, the Jaccard index is optimized further by combining the weighted cross-entropy loss with the Lovazs Softmax Loss.

3. Proposed Method

The proposed implementation is based on the RangeNet architecture, which is based on range image projection. The steps are described in the sections below:

3.1. Semantic KITTI Dataset

The most commonly used dataset for Perception tasks is Semantic KITTI which uses real-world inputs taken using Velodyne HDL 64-E sensor for collecting LiDAR Point Cloud data. This data contains multiple sequences taken in different cities, and contains the Velodyne sensor data in a bin file format, along with ground truth labels for the semantic classes of each of the point cloud scans.

3.2. Point Cloud as Input

Processing of the input point cloud from the LiDAR sensor can be done in 3D or 2D. In 3D, it is possible to process the 3D point cloud and perform semantic segmentation to obtain class labels for each point in the cloud. The other method involved projecting the point cloud from the 3D space to the 2D space, which can be done in many ways. This is done in order to reduce dimensionality.

In this paper, we used the spherical projections of the point clouds as our input. The spherical projections (or range images) generally consists of 5 channels (X, Y, Z, R and I), each channel representing some information of the 3D points. The X, Y, Z channels corresponds to the X, Y, Z coordinates of the 3d points, R is the euclidean distance or range of the point from the origin and I is the intensity of the point.(Topiwala, 2020)

3.3. Architecture

The architecture used is based on the RangeNet(Milioto et al., 2019). It is a Encoder-Decoder Network which uses the YOLOv3's Darknet53(Redmon & Farhadi, 2018) feature extractor as the Encoder. The encoder output is fed to a Decoder network, which in turn connects to a Segmentation Head block. There exist skip connections between the corresponding encoder and decoder blocks. This entire configuration results in a network with 50,377,364 trainable parameters. With the inclusion of the pixel shuffling layer, the number of trainable parameters decreases to 42,165,588.

3.3.1. Encoder

The Encoder Consists of a total of 53 convolution layers. Each of the convolution layer is followed by batch normalization and LeakyReLU activation. The complete encoder

network can be divided into multiple residual blocks. Each residual block consists of 2 convolution layers with kernel size of 1x1 followed by 3x3 kernel, and the input of this block is added into the output as the shortcut connections. This residual blocks are then grouped together in the groups of 1, 2, 8, 8, 4 and each of the group has a down sampling convolution before it and is followed by a dropout layer.

3.3.2. Decoder

The decoder used in the RangeNet consists of deconvolution layers corresponding to the downsampling convolutions in the encoder, followed by a residual block and a dropout layer. These deconvolution layers are replaced by pixel shuffling layers. These pixel shuffling layers perform upsampling of the input without adding parameters to the network, thus increasing the computational efficiency.

The network also consists of skip connections, that connect the input of each encoder block to the corresponding output of the decoder block, which can be seen in the architecture in figure 1. This helps in recovering some of the high frequency edge information which is lost during the down sampling process.

In order to implement the second modification to the network, the input to the decoder consists of sum of two components: the output of the encoder from current input frame and that from the previous input frame. This is done to consider the temporal dependence of two consecutive frames. For example, the cars from the previous frames will still be in the same general location as the current frame.

3.3.3. Segmentation Head

After the process of encoding and decoding the last layer performs a 1 x 1 convolution to get the output in the shape of C x H x W where C is the number of classes, which is followed by Softmax activation to get the pixel-wise probability distribution of the classes.

3.4. Training Process

Semantic KITTI is a vast dataset, containing many sequences of LiDAR point clouds. There are 21 sequences, out of which we use sequence 01 to 07 for training, sequence 08 and 09 for validation and sequence 00 is used for testing.

The training dataset is then divided into batches based on the batch size. Given the constraints of the hardware available, we could train with a batch size of 2 on Nvidia RTX 3070 (8GB GPU), while Google Colab Pro platform (P100 GPU) could train with a higher batch size of 4.

The loss was calculated using the weighted categorical cross entropy. The unlabelled points were excluded from the loss calculation and thus from the training by setting the weight of unlabelled points to zero. The weights are calculated as the inverse of the square root of the frequency of the classes.

3.5. Evaluation Metrics and Optimization

For the dense prediction task of semantic segmentation where we predict the class of each pixel of range image of LiDAR point cloud, the performance of the model is analyzed by calculating the pixel accuracy, mean IoU for all classes, individual class IoUs, Precision and Recall.

Intersection over Union (IoU) metric quantifies the extent of overlap between the ground truth and our prediction output. It is simply a ratio of number of pixels in common to the total number of pixels across the ground truth and prediction of class labels.

Implementation was tried with both SGD and ADAM as the optimizers, which are the preferred choice when it comes to semantic segmentation tasks. The final implementation uses ADAM optimizer because:

1. Adam computes individual learning rates for different parameters. In each iteration, the actual step size taken by Adam is approximately bounding the step-size hyperparameter.

2. Step Size of Adam Updates Rule is invariant to the magnitude of gradient which helps when going through areas with tiny gradients (such as saddle points or ravines). In such areas, Stochastic Gradient Descent (SGD) optimizer struggles to quickly navigate through them.

4. Experiments

4.1. Pixel Shuffling Layer

Upsampling in the decoder is a computationally expensive operation using a transpose convolution layer. This upsampling can also be performed by pixel shuffling, which does not increase the number of trainable parameters in the network. It leverages on the learnt feature maps to produce the upsampled feature maps by shuffling the pixels from the channel dimension to spatial dimension. This pixel shuffle layer reshapes the (Cr^2 x W x H) feature map to (C x Wr x Hr), where H, W, C and r represents the height, width, number of channels and up-scaling factor, respectively.

After inserting a pixel-shuffle layer followed by a dropout, in the baseline RangeNet implementation, the output shape of pixel shuffle layer did not match the inputs required by the up-sampling decoder block. To tackle this problem, a 1 x 1 convolution layer with required number of filters was introduced after the pixel shuffle layer to scale the output of pixel shuffle layer to correct the dimensions.

Table 1. Quantitative Comparison of Evaluation Metrics on Semantic-KITTI Test Dataset (in %)

Architecture	Epochs	Batch Size	Training Time (mins / epoch)	Accuracy	Cars	Pedestrian	Road	Parking	Sidewalk	Fence	Traffic Sign	Mean IOU
RangeNet (base)	15	2	67	**68.96**	**84.72**	0.00	**94.84**	**58.21**	**82.2**	**28.39**	2.69	**50.15**
+ PS	15	2	40	62.67	83.78	**0.26**	89.67	41.02	73.89	23.01	**20.62**	47.46
+ TC	15	2	67	30.64	33.6	**0.13**	47.15	0.00	24.31	3.61	0.00	15.54
+ PS + TC	15	2	40	60.72	75.85	0.00	85.8	40.62	68.32	22.12	14.21	43.84

+ PS denotes using Pixel Shuffle Layer. **+ TC** denotes using Temporal Connections.

4.2. Temporal Connections from previous encoder

In traditional neural network, all the inputs and outputs are independent of each other. But, whenever there is a sequence of data, the temporal dynamics that connects the data is more important than the spatial content of each individual frame. We tried to implement this concept and fed the temporal information from the previous hidden layer t to the next hidden layer $t + 1$ in our network, keeping the input data unshuffled and studied the outcomes of this modification.

5. Results

Table 1 shows the quantitative comparison of the evaluation metrics Accuracy and IoU on Semantic KITTI Dataset.

The baseline implementation of RangeNet resulted in better accuracy and IoU scores as compared to the modified implementations using Pixel Shuffle Layer and Encoder-Decoder Temporal Connections. Addition of Decoder Pixel Shuffle Layer resulted in poor performance as compared to Vanilla Implementation. This may be attributed to the lack of proper model training and may require more training (up to 150 epochs) to get solid conclusive results.

Also, our network implementation achieved high IoU scores very close to RangeNet, especially for the Cars class, which is desirable for autonomous driving, as false negatives are more likely to lead to accidents than false positives. The lower IoU scores of Fence, Pedestrian, and Traffic sign class is attributed to very few instances of fences, pedestrians and traffic signs in the dataset as well as these fence, pedestrians and traffic signs instances are much smaller in size and have very finer details, making it more difficult to segment. It is very obvious for Road and Sidewalk Class to have higher IoU Scores since these classes have very high occurrences in the dataset.

The figure 2 shows the output of the trained networks.

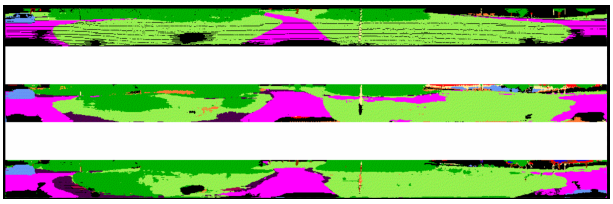

Figure 2. Ground truth (top), Segmented Output from Pixel Shuffle (middle) and Segmented Output from RangeNet baseline implementation (bottom)

6. Discussion

Our work suggests that, addition of Pixel Shuffle Layer in the baseline architecture does reduce the number of trainable parameters in the network. This reduces the computational memory requirements and eventually helps to reduce the network training time. Also, usage of temporal connections from the encoder to decoder proved inconclusive and will require training over more number of epochs to provide some substantial results.

More importantly, it can be noted that the above results are observed under training over a small number of epochs (20), and a maximum batch size of 4 due to limitations of computational resources. Thus, the achieved results might improve on further training of the model and modifying the current implementations to optimize the network architecture.

7. Conclusions and Future Work

In this paper, we studied the RangeNet architecture and implemented it from scratch using Tensorflow and observed the impact of adding a Pixel Shuffle layer and a temporal connection from encoder to decoder on the baseline architecture. It was observed that the Pixel Shuffling layer drastically reduces the number of trainable parameters, however, this implementation also reduces the mean IoU score of the segmentation task. The addition of the temporal connection between the past encoder output to the current network, does not improve the accuracy either. Though, the

network does learn during the training process, the baseline implementation gives higher accuracy over this implementation.

For the future, we would like to exploit our understanding of semantic segmentation architectures gathered from our analysis to design more efficient architectures for real-time applications. It is clear from our experiments that LiDAR Point Clouds captured from a moving car are easy to segment and deep network architectures perform robustly. We hope our experiments will encourage students and researchers to engage their attention towards the more challenging semantic segmentation task using deep learning techniques.

References

Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., and Gall, J. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019.

Cortinhal, Tiago, Tzelepis, George, and Aksoy, Eren Erdal. Salsanext: Fast, uncertainty-aware semantic segmentation of lidar point clouds for autonomous driving, 2020. URL https://arxiv.org/abs/2003.03653.

Milioto, Andres, Vizzo, Ignacio, Behley, Jens, and Stachniss, Cyrill. Rangenet ++: Fast and accurate lidar semantic segmentation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4213–4220, 2019. doi: 10.1109/IROS40897.2019.8967762.

Redmon, Joseph and Farhadi, Ali. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018. URL http://arxiv.org/abs/1804.02767.

Shi, Wenzhe, Caballero, Jose, Huszár, Ferenc, Totz, Johannes, Aitken, Andrew P., Bishop, Rob, Rueckert, Daniel, and Wang, Zehan. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network, 2016. URL https://arxiv.org/abs/1609.05158.

Topiwala, Anirudh. Spherical projection for point clouds, 2020. URL https://towardsdatascience.com/spherical-projection-for-point-clouds-56a2fc258e6c.

Wu, Bichen, Wan, Alvin, Yue, Xiangyu, and Keutzer, Kurt. 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)*, pp. 1887–1893, 2018. doi: 10.1109/ICRA.2018.8462926.

Wu, Bichen, Zhou, Xuanyu, Zhao, Sicheng, Yue, Xiangyu, and Keutzer, Kurt. Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 4376–4382, 2019. doi: 10.1109/ICRA.2019.8793495.

Xu, Chenfeng, Wu, Bichen, Wang, Zining, Zhan, Wei, Vajda, Peter, Keutzer, Kurt, and Tomizuka, Masayoshi. Squeezesegv3: Spatially-adaptive convolution for efficient point-cloud segmentation. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII*, pp. 1–19, Berlin, Heidelberg, 2020. Springer-Verlag. ISBN 978-3-030-58603-4. doi: 10.1007/978-3-030-58604-1_1. URL https://doi.org/10.1007/978-3-030-58604-1_1.