# Multi Projection Fusion for Real-time Semantic Segmentation of 3D LiDAR Point Clouds

Yara Ali Alnaggar* Mohamed Afifi* Karim Amer Mohamed ElHelw
Center for Informatics Science, Nile University
Giza, Egypt
{y.ali, moh.afifi, k.amer, melhelw}@nu.edu.eg

## Abstract

*Semantic segmentation of 3D point cloud data is essential for enhanced high-level perception in autonomous platforms. Furthermore, given the increasing deployment of LiDAR sensors onboard of cars and drones, a special emphasis is also placed on non-computationally intensive algorithms that operate on mobile GPUs. Previous efficient state-of-the-art methods relied on 2D spherical projection of point clouds as input for 2D fully convolutional neural networks to balance the accuracy-speed trade-off. This paper introduces a novel approach for 3D point cloud semantic segmentation that exploits multiple projections of the point cloud to mitigate the loss of information inherent in single projection methods. Our Multi-Projection Fusion (MPF) framework analyzes spherical and bird's-eye view projections using two separate highly-efficient 2D fully convolutional models then combines the segmentation results of both views. The proposed framework is validated on the SemanticKITTI dataset where it achieved a mIoU of 55.5 which is higher than state-of-the-art projection-based methods RangeNet++ [23] and PolarNet [44] while being 1.6x faster than the former and 3.1x faster than the latter.*

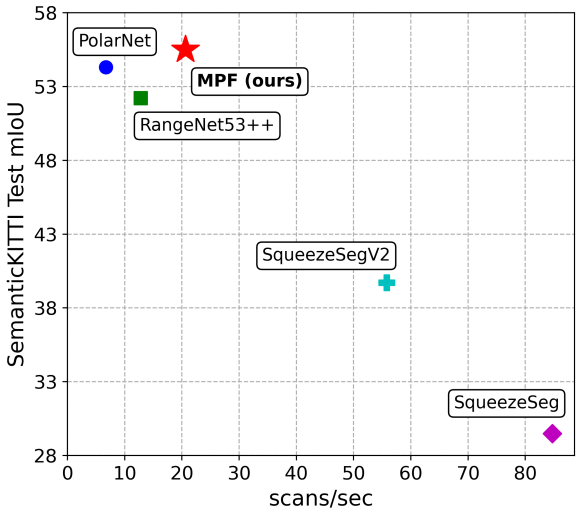



Figure 1: Computed scans per second vs. mIoU score on the test set of SemanticKITTI dataset using state-of-the-art projection-based methods. Our framework achieves the highest mIoU score; in addition, it is **3.1** and **1.6** times faster than PolarNet and RangeNet53++, respectively.

## 1. Introduction

Currently, Light Detection and Ranging (LiDAR) sensors are widely used in autonomous navigation systems where captured 3D point cloud data provides a rich source of information on the surrounding scene. Analyzing such data with deep learning models has gained a lot of attention in the research community especially for extracting semantic information to improve navigation accuracy and safety. In this case, semantic segmentation algorithms assign a label for each point in the 3D point cloud representing different classes of objects in the scene.

Convolutional Neural Networks (CNNs) have achieved state-of-the-art results in semantic segmentation tasks with fully convolutional architectures trained on huge amounts of labelled RGB data and making clever use of transfer learning. However, the same success has not yet been achieved in semantic segmentation of point cloud data due to lack of large annotated datasets. Furthermore, since point cloud semantic segmentation models are typically deployed on devices with limited computational capabilities onboard of mobile platforms (e.g. cars or drones), there is a need for high throughput while sustaining high accuracy to ensure the platform has enough time to make correct decisions.

There are two main approaches in the literature to tackle the task of semantic segmentation of 3D point clouds. The first applies 3D CNN models either on the raw cloud data

*equal contribution

points [27] or after transforming the points into 3D volumetric grid representations [38]. This incurs high computational costs [2] and hence not suitable for real-time systems. The second approach applies 2D CNN models to 2D projections of the 3D point cloud based on either bird's-eye view [29] or spherical view [40]. Currently, state-of-the-art methods such as RangeNet++ [23] applies a Fully Convolutional Neural Network (FCNN) on a spherical projection of the point cloud. However, there is an inevitable loss of information due to the projection operation which can limit model performance especially for distant points. In this paper, we introduce a novel framework for enhanced online semantic segmentation of 3D point clouds by incorporating multi-view projections of the same point cloud which results in an improved performance compared to single-projection models while attaining real-time performance. The contributions of this work can be summarized as follows. First, a novel MPF framework that utilizes multi-view projections and fusion of input point cloud to make up for the loss of information inherent in single projection methods. Second, the MPF framework processes spherical and bird's-eye projections using two independent models that can be selected to achieve optimum performance for a given platform (road vehicle, aerial drone, etc.) and/or deployed on separate GPUs. Third, the framework is scalable and, despite using only two projections in the current work, it can be directly extended to exploit multiple projections.

Incorporating information from multiple projections of 3D data has been used before in other domains to improve performance. Mortazi et al. [25] used a single 2D encoder-decoder CNN for CT-scan segmentation to parse all 2D slices in X, Y and Z directions. Chen et al. [4] used multiple 2D encoder CNN on spherical and bird's-eye views for the task of 3D object detection in point cloud data. However, to the best of our knowledge this setup has not been used before in semantic segmentation of 3D point clouds. This is primarily due to the added computational overhead of having the same complex network architecture for multiple views and back projection of results to the original point cloud space to compute point-level predictions (unlike [4] who only outputs 3D bounding boxes). The paper is organized as follows: Section 2 overviews related work, Section 3 describes the proposed framework and Section 4 presents obtained experimental results as well as an ablation study of our framework. Section 5 concludes the paper and points out future work directions.

## 2. Related Work

### 2.1. Semantic Image Segmentation

Semantic image segmentation has attracted a lot of attention in recent years following the success of deep CNN models, such as AlexNet [16], VGGNet [33], and ResNet [10] in image classification. The task aims at predicting pixel-level classification labels in order to have more precise information on objects in input images. One of the earliest work in this area is based on using Fully Convolutional Neural Networks (FCNNs) [21] where the model can assign a label for each pixel in the image in a single forward pass by extracting features using multi-layer encoder (in this case it was VGG model [33]) and apply up-sampling on these features combined with 1x1 convolution layer to classify each pixel. The idea was extended by Noh et al. [26] and Badrinarayanan et al. [1] by using a multi-layer decoder to transform extracted features into image space with the needed pixel labels. Ronneberger et al. [30] introduced the UNet architecture where skip connections between encoder and decoder layers further improve segmentation results. In addition to advances in model architecture design, having large annotated datasets for semantic segmentation tasks, such as Microsoft's COCO [20], and utilizing transfer learning from image classification models pre-trained on large datasets such as ImageNet [7] can significantly improve semantic segmentation accuracy.

### 2.2. Semantic Segmentation of 3D Point Cloud

Most 3D point cloud semantic segmentation algorithms employ a Fully Convolutional Neural Network (FCNN) in a way similar to 2D semantic image segmentation but with the difference of how the FCNN is applied to 3D structures. These algorithms can essentially be grouped into two categories. The first category includes models that use 3D convolutions as in SegCloud [38] where a 3D FCNN is applied to point cloud after voxelization and transformation into homogeneous 3D grid. However, 3D convolutions are computationally intensive and 3D volumes of point clouds are typically sparse. Other models have special convolution layers for 3d points in order to process a point cloud in its raw format with examples including PointNet++ [28], TangentConv [37] and KPConv [39]. Recent work in RandLA [12] improves run-time while maintaining a high segmentation accuracy compared to previously mentioned methods.

The second category of algorithms apply 2D FCNN models after projecting 3D point clouds onto 2D space. In SqueezeSeg [40] and SqueezeSegV2 [41], spherical projection is performed on point cloud then 2D encoder-decoder architecture is applied. RangeNet++ [23] improves segmentation results by using a deeper FCNN model and employing post-processing using K-Nearest Neighbour (KNN). Algorithms in this category not only improve inference time but also enhance segmentation accuracy by capitalizing on the success of CNNs in 2D image segmentation. Different projections can also be used as in VolMap [29] which uses Cartesian bird's-eye projection and PolarNet [44] which uses polar bird's-eye projection combined with ring convolutions. The proposed framework advances current state-of-

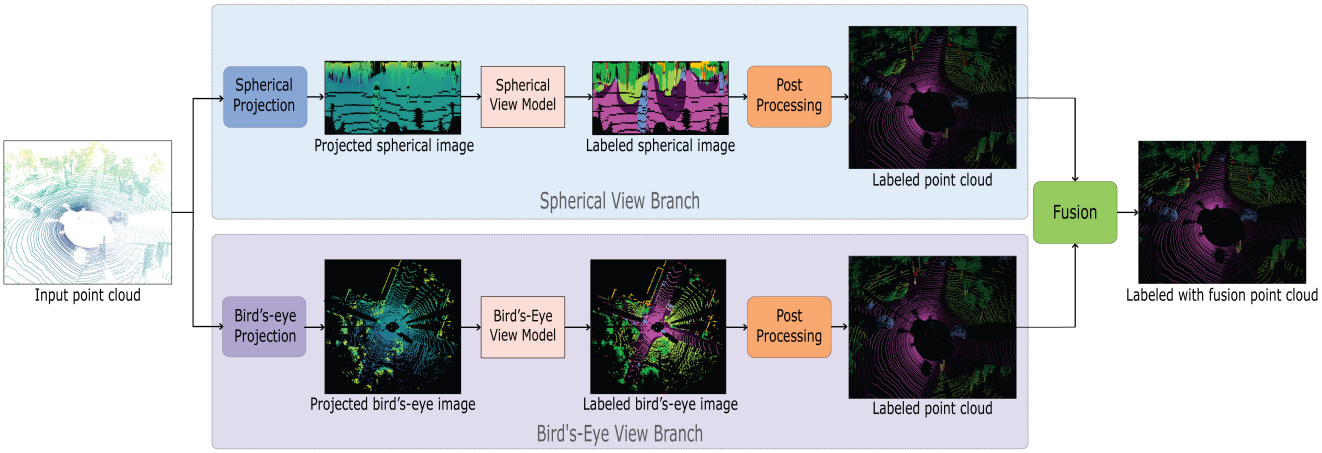



Figure 2: Proposed MPF framework overview. The framework takes as input a 3D point cloud that undergoes a series of operations in the Spherical View Branch and the Bird'e-Eye View branch. Each branch is composed of three main processing blocks where the first block transforms the 3D point point cloud into its respective 2D projection. The second block, Spherical or Bird's-Eye View Model, predicts segmentation of the projected 2D image with a FCNN model where each view has its own model. The third block, Post Processing, further processes the semantically-segmented projected view and assigns to each point in the input cloud its corresponding softmax probabilities. Finally, information from the two branches are fused by the Fusion block to produce the final semantic label of each point in the 3D point cloud.

the-art of projection based methods in point cloud segmentation by making use of both spherical and Cartesian bird's-eye projections to reduce the loss of information stemming from using a single projection. Finally, it is worth mentioning that one of the key challenges in 3D point cloud semantic segmentation is the lack of large labeled point cloud datasets. The current benchmark dataset is SemanticKITTI [2] which has around 43K frames collected using 360 Velodyne LiDAR sensor [18]. Other datasets are either generated synthetically using simulation environments such as Virtual Kitti [8] or have small number of samples such as Paris-Lille-3D [31].

### 2.3. Efficient Deep Learning Architectures

With the impressive results achieved by deep learning models in detection and classification tasks, there is a growing need for efficient models to be deployed on embedded devices and mobile platforms. A number of network architectures that achieve high classification accuracy while having real-time inference have been recently proposed. For instance, MobileNetV1 [11] uses depth-wise separable convolutions while ShuffleNet [43] utilizes group convolutions and channel shuffling to reduce computations. MobileNetV2 [32] achieves improved accuracy while maintaining fast inference by using inverted residual blocks. Although these models are designed for classification tasks, they can be used in the context of semantic segmentation as encoders in FCNN models in order to benefit from their efficient architecture.

### 2.4. Segmentation Loss

Segmentation models, regardless of their input, are initially trained using classification losses such as Cross Entropy loss or Focal loss [19] because the end goal is to assign a label for each pixel (or point). However, such losses lack the global information of predicted and target object masks. Therefore, research work has been conducted to develop a loss function that penalizes the difference between predicted and ground-truth masks as a whole. Milletari et al. [24] developed a soft version of Dice coefficient with continuous probabilities instead of discrete 0 or 1 values while Berman et al. [3] introduced Lovasz Softmax loss which is a function surrogate approximation of the Jaccard coefficient [14]. Currently, Lovasz Softmax loss is the state-of-the-art segmentation loss and is usually combined with classification loss to have a penalty on both local and global information.

## 3. Proposed Method

We developed a Multi-Projection Fusion (MPF) framework for semantic segmentation of 3D point clouds that relies on using two 2D projections of the point cloud where each projection is processed by an independent FCNN model. As illustrated in Figure 2, the proposed pipeline starts by feeding the input point cloud to two branches, one responsible for spherical view projection and the other for

| Input | Operator | t | c | n | s |
|---|---|---|---|---|---|
| $5 \times 64 \times w$ | conv2d | - | 32 | - | 1 |
| $32 \times 64 \times w$ | bottleneck | 1 | 16 | 1 | 2 |
| $16 \times 64 \times \frac{w}{2}$ | bottleneck | 6 | 24 | 2 | 2 |
| $24 \times 64 \times \frac{w}{4}$ | bottleneck | 6 | 32 | 3 | 2 |
| $32 \times 64 \times \frac{w}{8}$ | bottleneck | 6 | 64 | 4 | 2 |
| $64 \times 64 \times \frac{w}{16}$ | bottleneck | 6 | 96 | 3 | 1 |
| $96 \times 64 \times \frac{w}{16}$ | bottleneck | 6 | 160 | 3 | 2 |
| $160 \times 64 \times \frac{w}{32}$ | bottleneck | 6 | 320 | 1 | 1 |
| $320 \times 64 \times \frac{w}{4}$ | deconv2d | - | 96 | - | 8 |
| $96 \times 64 \times w$ | deconv2d | - | 32 | - | 4 |
| $32 \times 64 \times w$ | conv2d | - | 20 | - | 1 |

Table 1: Spherical-View Model Architecture. The model is based on MobileNetV2 [32] with additional 2 deconvolutional layers and one 1x1 convolutional layer as a decoder. $n$ is the repetition number for a sequence of layers in block, $t$ is block expansion factor, $c$ is the number of output channels, $s$ is block stride.

bird's-eye view projection. Each branch applies semantic segmentation on the projected point cloud. Subsequently, predictions from the two branches are fused to produce the final prediction. It is assumed that the input point cloud is collected by a LiDAR sensor that returns point coordinates $x, y, z$ values and remission of returned signals $rem$, e.g Velodyne HDL-64E [18]. In the following sub-sections, we present details of each block in the two branches of the pipeline.

### 3.1. Spherical View Projection

This section explains the process of transforming a $360^\circ$ point cloud into a 2D spherical image that is fed into subsequent Spherical View Model block, as proposed by [40]. At the start, the 3D point cloud is mapped from Euclidean space $(x, y, z)$ to Spherical space $(\theta, \phi, r)$ by applying Equation 1.

$$\begin{pmatrix} \theta \\ \phi \\ r \end{pmatrix} = \begin{pmatrix} arcsin(\frac{z}{\sqrt{x^2+y^2+z^2}}) \\ arctan(y, x) \\ \sqrt{x^2+y^2+z^2} \end{pmatrix} \quad (1)$$

Subsequently, the points are embedded into a 2D spherical image with dimensions $(H, W)$ by discretizing points' $\theta$ and $\phi$ angles using Equation 2:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{2}[1 - \phi\pi^{-1}]w \\ [1 - (\theta + f_{up})f^{-1}]h \end{pmatrix}, \quad (2)$$

where $u$ and $v$ represent point indices in the spherical image and $f = f_{up} + f_{down}$ is the sensor's vertical field-of-view.

| Input | Operator | c | s |
|---|---|---|---|
| $4 \times 256 \times 256$ | ConvBlock | 64 | 1 |
| $64 \times 256 \times 256$ | DownBlock | 128 | 2 |
| $128 \times 128 \times 128$ | DownBlock | 256 | 2 |
| $256 \times 64 \times 64$ | UpBlock | 128 | 2 |
| $128 \times 128 \times 128$ | UpBlock | 64 | 2 |
| $64 \times 256 \times 256$ | conv2d | 20 | 1 |

Table 2: Bird's-Eye View Model Architecture. $c$ is the number of output channels and $s$ is layer stride.

The mapping and discretization steps may result in some 3D points sharing the same $u$ and $v$ values. To mitigate this condition, 3D points that are closer to LiDAR are given priority to be represented in the 2D image by ordering the points descendingly based on their range value. The ordered list of points will be embedded into the 2D spherical image using its corresponding $u$ and $v$ coordinates. By the end of this process, the resulting 2D spherical image will have five channels corresponding to distinct point features: $x, y, z, r$ and remission $rem$ which is analogous to RGB images that have three channels, one for each color.

### 3.2. Spherical-View Model

The Spherical-View Model is a deep learning segmentation model based on FCNN architecture with encoder and decoder parts. The network encoder utilises MobileNetV2 [32] as lightweight backbone that provides real-time performance on mobile devices. The backbone is composed of a sequence of basic building blocks called inverted residual blocks that form bottlenecks with residuals. The first and last bottleneck layer expands and compresses input and output tensors, respectively. The intermediate layers are high-capacity layers responsible for extracting high-level information from the expanded tensors.

For network decoder, we apply two learnable upsampling layers known as transposed convolution layers. The first layer upsamples the input tensor 8 times and the second layer 4 times. At the end, we add convolution layer and softmax logits to output semantically segmented image. Furthermore, dropout layers are added as regularization. Table 1 provides details of the Spherical-View Model layers.

### 3.3. Bird's-Eye View Projection

The second projection in our framework is the 2D bird's-eye view projection. It uses the $x$ and $y$ coordinates of each point and collapses the 3D cloud along the $z$ dimension. The 3D point cloud is thus projected on the $x - y$ plane that is discretized using a rectangular grid with a defined width and height. For each cell in the grid, we keep at most

one projected point corresponding to the point that has the maximum $z$ value among all points projected onto that cell. Points that get projected outside the boundaries of the grid are discarded. Finally, the grid is converted into a 4-channel image where each pixel in the image represents a cell in the grid. Four cell attributes are extracted to form the 4 channels of the image, namely $x$, $y$, $z$, and remission $rem$.

### 3.4. Bird's-Eye View Model

Although MobileNetV2 [32] is highly efficient, using it twice for both views will decrease the overall throughput. Since the MPF framework allows using independent network in each processing branch, we decided to use a network with fewer parameters compared to MobileNetV2 [32]. Specifically, a light weight modified version of the UNet [30] encoder-decoder architecture is used for segmentation of bird's-eye view images. As shown in table 2, the encoder consists of 2 downsampling convolutional blocks and the decoder consists of 2 upsampling convolutional blocks with skip connections between corresponding encoder and decoder blocks. In our experiments it is shown that it is sufficient to use only two blocks in both encoder and decoder which significantly improves network inference efficiency. Each block consists of two 2D convolution layers with kernel size 3 followed by max pooling for encoder and preceded by bi-linear upsampling for decoder. We use 2D batch normalization [13] followed by ELU [5] non-linearity between successive convolutional layers.

### 3.5. Post-Processing

The goal of the Post-Processing step is to get semantic labels for all points in the input 3D point cloud based on the semantically-segmented images produced by the Spherical and Bird's-Eye View Models. The segmentation results for each pixel are softmax probability scores for each of the 20 possible classes. The segmented 3D point cloud is computed as follows: each 3D point is projected onto the 2D segmented image and a 2D square window centered around the projection location is calculated. Then, a weighted vote over all classes is performed by computing a weighted sum of the softmax probabilities of all pixels inside the window, where the weights are inversely proportional to the distance between the 3D point under consideration and the 3D points represented by pixels in the window. In particular, we use a Gaussian function with zero mean and fixed standard deviation to compute the weight corresponding to each distance. The output of this step is a vector of scores for each point in the 3D point cloud. Finally, the score vector is normalized by dividing by the number of points that contributed in the voting. This step is necessary because both views have sparse pixels and the number of pixels inside a window can vary considerably from one view to another. The details of the algorithm are shown in Algorithm 1. During implementation, we eliminated the use of all loops and used fully-vectorized code which run on GPU for fast processing. Our proposed post-processing is similar to KNN post-processing in [23] however it uses soft voting with softmax probabilities instead of hard voting and takes the vote of non-sparse pixels only.

**Algorithm 1:** Post-Processing Algorithm

**Post-Processing**
**Parameters**
Number of classes: $C$
Number of points: $N$
Size of the projection image: $H$x$W$
Size of the sliding window : $K$x$K$
Standard deviation for the gaussian function: $\sigma$

**Data**
Point cloud coordinates $P_{xyz}$. Size = $N$x3
Projection image $I_{xyz}$. Size = $H$x$W$x3
Output of the segmentation network $I_{softmax}$. Size = $H$x$W$x$C$

**Output**
$Scores$. Class scores for each point in the original cloud. Size = $N$x$C$

**Algorithm**
**foreach** $i \in [1:N]$ **do**
  // Get the pixel to which this point is projected
  $u, v = get\_projection\_indices(P_{xyz}[i])$
  // Initialize all class scores for the i'th points to zeros
  $Scores[i] = zeros(C)$
  // Initialize number of non-sparse pixels to zero
  $M = 0$
  // Loop over pixels currently inside the sliding window
  **foreach** $u' \in [u - \lfloor k/2 \rfloor : u + \lfloor k/2 \rfloor]$ **do**
    **foreach** $v' \in [v - \lfloor k/2 \rfloor : v + \lfloor k/2 \rfloor]$ **do**
      **if** $I_{xyz}[u', v']$ *is not sparse* **then**
        $d = get\_distance(P_{xyz}[i], I_{xyz}[u', v'])$
        $weight = exp(-d^2/2\sigma^2)$
        $Scores[i] += weight * I_{softmax}[u', v']$
        $M += 1$
      **end**
    **end**
  **end**
  $Scores[i] = Scores[i]/M$
**end**
return $Scores$

## 4. Experimental Evaluation

### 4.1. Datasets

We trained both Spherical View and Bird's-Eye View networks on the SemanticKITTI dataset [2] which provided

| Model | Input Size | scans/sec | #Params | #MACs | mIoU | car | bicycle | motorcycle | truck | other-vehicle | person | bicyclist | motorcyclist | road | parking | sidewalk | other-ground | building | fence | vegetation | trunk | terrain | pole | traffic-sign |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PointNet [27] | 50000pts | - | - | - | 14.6 | 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 |
| PointNet++ [28] | | | | | 20.1 | 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 |
| SPGraph [17] | | | | | 20.0 | 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 |
| SPLATNet [35] | | | | | 22.8 | 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 |
| TangentConv [37] | | | | | 35.9 | 86.8 | 1.3 | 12.7 | 11.6 | 10.2 | 17.1 | 20.2 | 0.5 | 82.9 | 15.2 | 61.7 | 9.0 | 82.2 | 44.2 | 75.5 | 42.5 | 55.5 | 30.2 | 22.2 |
| RandLA [12] | | | | | 53.9 | 94.2 | 26.0 | 25.8 | **40.1** | 38.9 | 49.2 | 48.2 | 7.2 | 90.7 | 60.3 | 73.7 | 20.4 | 86.9 | 56.3 | 81.4 | 66.8 | 49.2 | 47.7 | 38.1 |
| KPConv [39] | | | | | **58.8** | **96.0** | 30.2 | **42.5** | 33.4 | **44.3** | **61.5** | **61.6** | 11.8 | 88.8 | 61.3 | 72.7 | 31.6 | **90.5** | **64.2** | **84.8** | **69.2** | 69.1 | **56.4** | 47.4 |
| SqueezeSeg [40] | 64 × 2048 px | **84.7** | - | - | 29.5 | 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 |
| SqueezeSegV2 [41] | | 55.8 | **928.5** K | **13.6** G | 39.7 | 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 |
| RangeNet21 [23] | 64 × 2048 px | 21.7 | - | - | 47.4 | 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 |
| RangeNet53++ | 64 × 512 px | 38.5 | | | 41.9 | 87.4 | 9.9 | 12.4 | 19.6 | 7.9 | 18.1 | 29.5 | 2.5 | 90.0 | 50.7 | 70.0 | 2.0 | 80.2 | 48.9 | 77.1 | 45.7 | 64.1 | 37.1 | 42.0 |
| RangeNet53++ | 64 × 1024 px | 23.3 | - | - | 48.0 | 90.3 | 20.6 | 27.1 | 25.2 | 17.6 | 29.6 | 34.2 | 7.1 | 90.4 | 52.3 | 72.7 | 22.8 | 83.9 | 53.3 | 77.7 | 52.5 | 63.7 | 43.8 | 47.2 |
| RangeNet53 | 64 × 2048 px | 13.3 | 50.4 M | 377.1 G | 49.9 | 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 |
| RangeNet53++ | 64 × 2048 px | 12.8 | - | - | 52.2 | 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 |
| PolarNet [44] | 480 × 360 × 32 | 6.7 | 13.6 M | 135.0 G | 54.3 | 93.8 | **40.3** | 30.1 | 22.9 | 28.5 | 43.2 | 40.2 | 5.6 | 90.8 | 61.7 | 74.4 | 21.7 | 90.0 | 61.3 | 84.0 | 65.5 | 67.8 | 51.8 | 57.5 |
| MPF (ours) | 64 × 512 px | 33.7 | - | - | 48.9 | 91.1 | 22.0 | 19.7 | 18.8 | 16.5 | 30.0 | 36.2 | 4.2 | 91.1 | 61.9 | 74.1 | 29.4 | 86.7 | 56.2 | 82.3 | 51.6 | 68.9 | 38.6 | 49.8 |
| MPF (ours) | 64 × 1024 px | 28.5 | - | - | 53.6 | 92.7 | 28.2 | 30.5 | 26.9 | 25.2 | 42.5 | 44.5 | 9.5 | 90.5 | 64.7 | 74.3 | **32.0** | 88.3 | 59.0 | 83.4 | 56.6 | **69.8** | 46.0 | 54.9 |
| MPF (ours) | 64 × 2048 px | 20.6 | 3.18 M | 27.0 G | 55.5 | 93.4 | 30.2 | 38.3 | 26.1 | 28.5 | 48.1 | 46.1 | **18.1** | 90.6 | 62.3 | 74.5 | 30.6 | 88.5 | 59.7 | 83.5 | 59.7 | 69.2 | 49.7 | **58.1** |

Table 3: mIoU scores on SemanticKITTI test set [1]. Our proposed MPF utilizes smaller number of parameters compared to projection-based methods Rangenet53++ [23] and PolarNet [44] while maintaining higher segmentation results.

point-wise semantic label annotations for all scans in the KITTI odometry dataset [9]. The dataset consists of over 43,000 360° LiDAR scans, divided into 11 training sequences for which ground-truth annotations are provided and 11 test sequences. We used sequence 08 as our validation set and trained our networks on the other 10 sequences.

## 4.2. Training Configuration

### 4.2.1 Spherical View Model

The Spherical View Model was trained from scratch using a combined objective function of Focal [19] and Lovász-Softmax [3] losses:

$$L_{spherical\ view\ model} = L_{focal} + L_{lovasz}$$

$$L_{focal} = -\sum_{n}\sum_{i}(1-p_{n,i})^{\gamma} log(p_{n,i})$$

where $p_{n,i}$ is the probability of the ground-truth class at image $n$ and pixel $i$ and $\gamma$ is the focusing factor. For optimization, SGD with 0.9 momentum, 0.0001 weight decay and mini-batch of 8 was used. We also used Cosine Annealing scheduler with warm restart [22] for 5 cycles with learning rate that starts at 0.05 and decreases till 0, and cycle length of 30 epochs. Following similar works, the model was trained with image sizes of 64x512, 64x 1024 and 64x2048.

### 4.2.2 Bird's-Eye View Model

To train the Bird's-Eye View Model, we used SGD with one-cycle [34] learning and momentum annealing strategy. Learning rate was cycled between $0.001$ and $0.1$, while momentum was cycled inversely to learning rate between $0.85$ and $0.95$. The model was trained for 30 epochs using cross entropy loss and Lovász-Softmax loss:

$$L_{bird's-eye\ view\ model} = L_{cross\ entropy} + L_{lovasz}$$

$$L_{cross\ entropy} = -\sum_{n}\sum_{i} log(p_{n,i})$$

where $p_{n,i}$ is the probability of the ground-truth class at image $n$ and pixel $i$. The model was trained only using images of size 256x256.

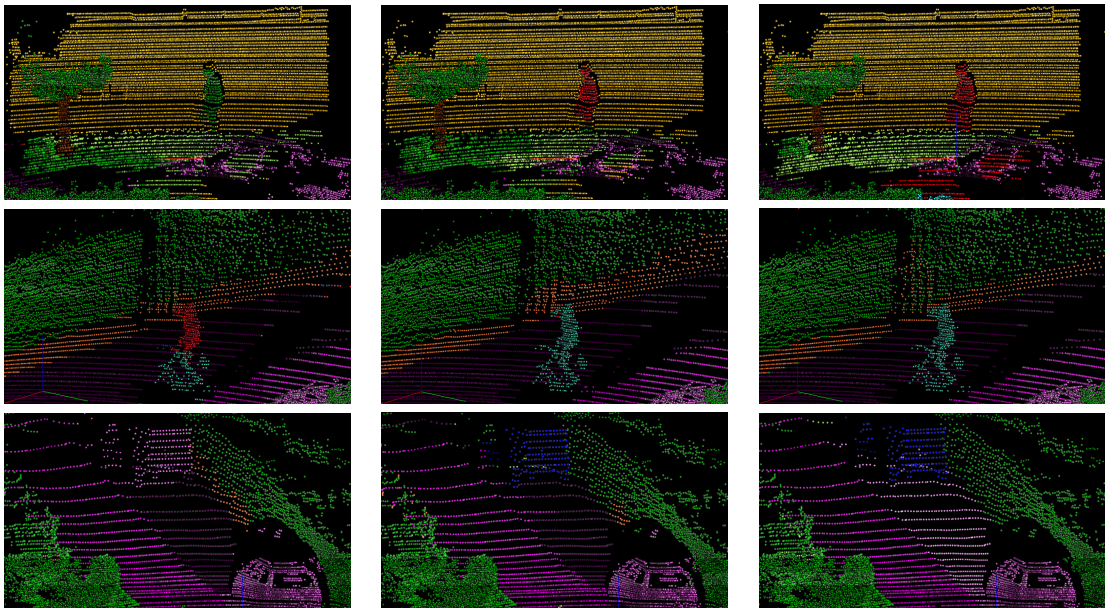
### 4.2.3 Post-Processing

For post-processing, we used square kernels of size 3 for both views to extract local windows centered at projected pixels. Network predictions for sparse pixels (pixels that have no corresponding 3D points) are ignored. Then to compute the weight for each pixel, a Gaussian function is used as described in Section 3.5 with $\sigma = 1$ to compute the weights corresponding to different distances.

## 4.3. Data Augmentation

Data augmentation is considered an effective tool to improve model generalization to unseen data. We therefore apply Spherical-View augmentation and Bird's-Eye View augmentation. In the former, the 3D point cloud is processed by pipeline of four randomly executed (with 0.5 probability) Affine transformations: translation parallel to y-axis, rotation about z-axis, scaling around the origin and flipping around y-axis. Augmentation is also applied after projecting the 3D point cloud onto 2D space. A CoarseDropout function by [15] is used to drop pixels randomly by 0.005 probability and the image is cropped to half on the horizontal axis starting from a randomly sampled coordinate.

[1] We included only peer-reviewed works in the study however there are other interesting approaches that can be easily incorporated in our framework such as SalsaNext [6] and SqueezeSegV3 [42].



(a) RangeNet53++ (b) MPF (ours) (c) Ground Turth

Figure 3: Qualitative segmentation results on SemanticKITTI validation sequence 8 that compare our MPF framework against RangeNet53++. Top: Our framework correctly segmented the 'person' labeled in red. Middle: Our framework segmented the 'bicyclist' object, a class rarely representated in the dataset, much better than RangeNet53++. Bottom: the MPF framework correctly segmented the 'other-vehicle' object (in upper middle location), which was completely missed by RangeNet53++.

In Bird's-Eye View augmentation and prior to projecting the point cloud on the $x-y$ plane, the cloud is transformed by applying random rotation around the $z$ axis, scaling by a uniformly sampled factor, translating in the $x$ and $y$ directions and finally a random noise sampled from a normal distribution with $0$ mean and $0.2$ standard deviation is applied to the $z$ channel. Each of these operations is applied with probability of $0.5$.

### 4.4. Results

This section describes the performance of the proposed Multi-Projection Fusion (MPF) framework. Table 3 presents the quantitative results obtained by the proposed approach versus state-of-the-art point cloud semantic segmentation methods on the SemanticKITTI test set over 19 classes. SemanticKITTI uses Intersection-over-Union (IoU) metric to report per-class score:

$$IoU = \frac{|P \cap G|}{|P \cup G|} \tag{3}$$

where $P$ and $G$ are class points predictions and ground truth, respectively. The mean IoU (mIoU) over all classes is also reported. The scans per second rate are reported by measuring combined projection and inference time (unlike PolarNet [44] which reports inference time only) on a single NVIDIA GeForce GTX 1080 Ti GPU card. The results demonstrate that the proposed MPF framework achieves the highest mIoU score across all baseline projection-based methods while also having higher scans per second rate and less parameters compared to RangeNet++ [23] and PolarNet [44]. Although 3D-methods achieves the highest mIOU scores, it lags significantly in running time as shown in [36] and [12].

Figure 1 show qualitative examples from the SemanticKITTI validation sequence where our proposed approach outperforms RangeNet53++ [23] in segmenting objects located far from LiDAR position. This is attributed to using two independent complementary projections and intelligently fusing segmentation results of each projection. It is worth mentioning that data augmentation described in Section 4.3 improved validation mIoU by $2.7\%$ for spherical view model, $7.7\%$ for bird's-eye view model and an overall improvement by $7.2\%$ as shown in Table 4.

### 4.5. Ablation Study

#### 4.5.1 Post-Processing

We studied the effect of different standard deviation values on the performance of the Gaussian function used presented in Algorithm 1. A grid search was used to jointly compute the best values for the standard deviation used in both spherical view and bird's-eye view post-processing. As shown in Figure 4, the results show that setting all standard devia-

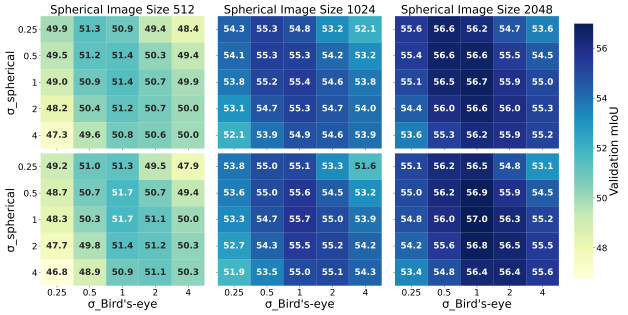



Figure 4: Post-processing $\sigma$ values of spherical and bird's-eye views against validation mIoU. Top: Euclidean distance. Bottom: Manhattan distance.

| Model | Without Augmentation | With Augmentation |
|---|---|---|
| Spherical view | 39.9 | **42.6** |
| Bird's-eye view | 33.6 | **41.3** |
| MPF | 44.5 | **51.7** |

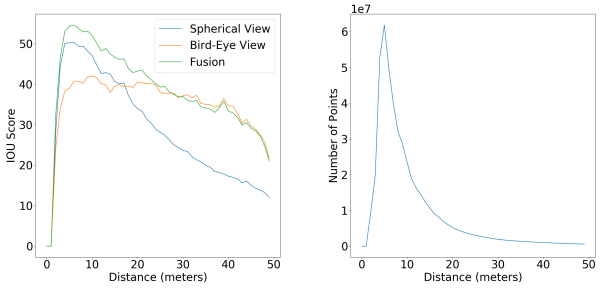
Table 4: Ablation study on the effect of using augmentation on validation mIoU score. The image size used for spherical view model is 64x512.

tion values to 1 when using Manhattan distance consistently yields the best results. We also tried larger sliding window sizes but it did not improve the score and reduced the overall FPS of our framework. The best configurations from this study were used for test submission.

#### 4.5.2 Multi-Projection Fusion

We conducted several ablation studies to demonstrate the efficacy of employing multiple projections as opposed to single projection. In the first experiment, we investigate the mIoU score of two established spherical projection models, SqueezeSeg [40, 41] and RangeNet [26], with and without the incorporation of the bird's-eye projection in table 5. The results demonstrate that fusion of information from more than one projection significantly enhances the obtained segmentation results despite using simple network model and projected low-resolution images for the bird's-eye view. It can thus be concluded that using multiple projections of the same point cloud does improve overall segmentation results by providing additional information for model adaptation.

The second experiment shows that fusion helps to improve mIoU score for both near and far points as seen in Figure 5a. Since the majority of LiDAR points are typically at distances $< 20$ meters, as shown in Figure 5b, a slight improvement of framework performance for near distance points can have a significant impact on the overall IoU results. It also observed that the performance of point cloud




(a) (b)

Figure 5: (a) The mIoU score vs. distance for near and far points. Spherical view image of size 512 used in this experiment. (b) Number of points in the validation set vs. distance.

| Network | Size | W/o fusion | W fusion |
|---|---|---|---|
| SqueezeSeg[40] | 2048 | 30.5 | **44.3** |
| SqueezeSegV2[41] | 2048 | 40.4 | **48.6** |
| RangeNet53++[23] | 512 | 37.5 | **44.4** |
| RangeNet53++ | 1024 | 36.5 | **42.2** |
| RangeNet53++ | 2048 | 50.3 | **55.4** |
| MPF (ours) | 512 | 42.6 | **51.7** |
| MPF (ours) | 1024 | 48.5 | **55.7** |
| MPF (ours) | 2048 | 50.7 | **57.0** |

Table 5: Results of adding the bird's-eye projection to single spherical projection models on SemanticKITTI validation set.

segmentation using spherical view projections degrades for points farther away from the LiDAR position. Unlike spherical view images, the bird's-eye view images use Cartesian coordinates which means the pixels of the image correspond to uniform 3D elements whose spatial resolution does not change as we go farther from the LiDAR position which makes segmentation results independent of point distances.

## 5. Conclusions and Future Work

This paper presented a novel multi-projection fusion framework for point cloud semantic segmentation by using spherical and bird's-eye view projections and fusion of results using soft voting mechanism. The proposed framework achieves improved segmentation results over single projection methods while having higher throughput. Future work directions include combining both projections into a single multi-view unified model and investigating using more than two projections within the framework.

## References


[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[2] 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 *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019.

[3] Maxim Berman, Amal Rannen Triki, and Matthew B Blaschko. The lovász-softmax loss: a tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4413–4421, 2018.

[4] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017.

[5] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[6] Tiago Cortinhal, George Tzelepis, and Eren Erdal Aksoy. Salsanext: Fast semantic segmentation of lidar point clouds for autonomous driving. *arXiv preprint arXiv:2003.03653*, 2020.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[8] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4340–4349, 2016.

[9] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[12] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. *arXiv preprint arXiv:1911.11236*, 2019.

[13] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[14] Paul Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bull Soc Vaudoise Sci Nat*, 37:547–579, 1901.

[15] Alexander B. Jung, Kentaro Wada, Jon Crall, Satoshi Tanaka, Jake Graving, Christoph Reinders, Sarthak Yadav, Joy Banerjee, Gábor Vecsei, Adam Kraft, Zheng Rui, Jirka Borovec, Christian Vallentin, Semen Zhydenko, Kilian Pfeiffer, Ben Cook, Ismael Fernández, François-Michel De Rainville, Chi-Hung Weng, Abner Ayala-Acevedo, Raphael Meudec, Matias Laporte, et al. imgaug. `https://github.com/aleju/imgaug`, 2020. Online; accessed 01-Feb-2020.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[17] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4558–4567, 2018.

[18] Velodyne Lidar. Velodyne lidar hdl-64e.

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

[20] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[21] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[22] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[23] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Rangenet++: Fast and accurate lidar semantic segmentation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[24] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 565–571. IEEE, 2016.

[25] Aliasghar Mortazi, Rashed Karim, Kawal Rhode, Jeremy Burt, and Ulas Bagci. Cardiacnet: segmentation of left atrium and proximal pulmonary veins from mri using multi-view cnn. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 377–385. Springer, 2017.

[26] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015.

[27] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification

and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[28] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017.

[29] Hager Radi and Waleed Ali. Volmap: A real-time model for semantic segmentation of a lidar surrounding view. *arXiv preprint arXiv:1906.11873*, 2019.

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

[31] Xavier Roynard, Jean-Emmanuel Deschaud, and François Goulette. Paris-lille-3d: A large and high-quality ground-truth urban point cloud dataset for automatic segmentation and classification. *The International Journal of Robotics Research*, 37(6):545–557, 2018.

[32] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[34] Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of residual networks using large learning rates. *CoRR*, abs/1708.07120, 2017.

[35] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2530–2539, 2018.

[36] Haotian* Tang, Zhijian* Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution. In *European Conference on Computer Vision*, 2020.

[37] Maxim Tatarchenko*, Jaesik Park*, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3D. *CVPR*, 2018.

[38] Lyne Tchapmi, Christopher Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. Segcloud: Semantic segmentation of 3d point clouds. In *2017 international conference on 3D vision (3DV)*, pages 537–547. IEEE, 2017.

[39] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6411–6420, 2019.

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

[41] 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 *2019 International Conference on Robotics and Automation (ICRA)*, pages 4376–4382, 2019.

[42] Chenfeng Xu, Bichen Wu, Zining Wang, Wei Zhan, Peter Vajda, Kurt Keutzer, and Masayoshi Tomizuka. Squeezesegv3: Spatially-adaptive convolution for efficient point-cloud segmentation. *arXiv preprint arXiv:2004.01803*, 2020.

[43] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856, 2018.

[44] Yang Zhang, Zixiang Zhou, Philip David, Xiangyu Yue, Zerong Xi, and Hassan Foroosh. Polarnet: An improved grid representation for online lidar point clouds semantic segmentation. *arXiv preprint arXiv:2003.14032*, 2020.