VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection

Date: 2022.07.13

Name: ChangMin An

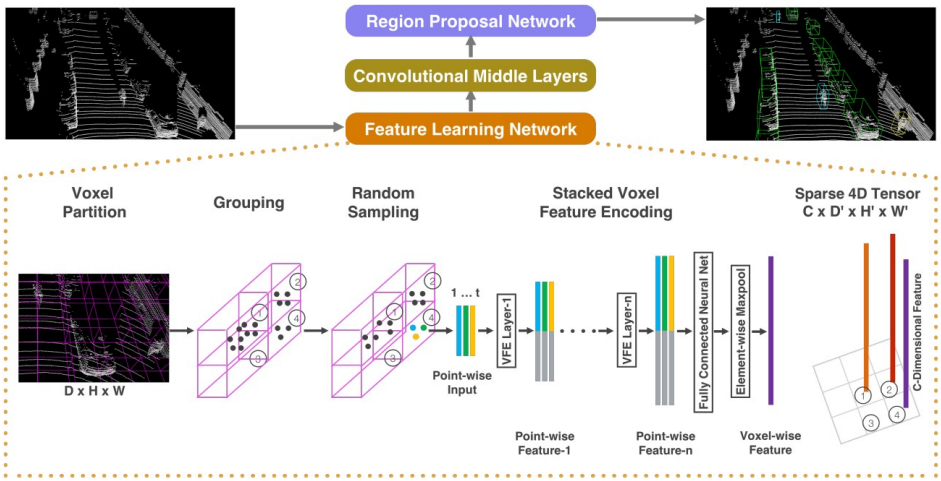
Github: Link

I. Introduction

Point cloud로 3D object detection을 할 때 부딪치는 문제점들은 2D와 비교했을 때, 데이터가 매우 sparse하고 sensor의 상대적 위치에 따라 point 수가 달라져서 매우 다양한 point density를 갖는 것에 있다. 기존에는 이 문제를 해결하기 위해 직접 feature를 찾아서 변환하는 hand crafted feature representation 방법을 사용하였다. 하지만 VoxelNet은 이러한 과정 없이 feature extraction과 bounding box prediction을 single stage로 합쳐서 학습하는 것을 제안한다.

- Hand crafted feature가 필요 없이 raw point cloud를 input으로 받아 end-to-end로 학습할 수 있는 VoxelNet
- VoxelNet을 sparse point structure와 voxel grid 병렬 처리에 대한 효율적인 학습 방법 제안

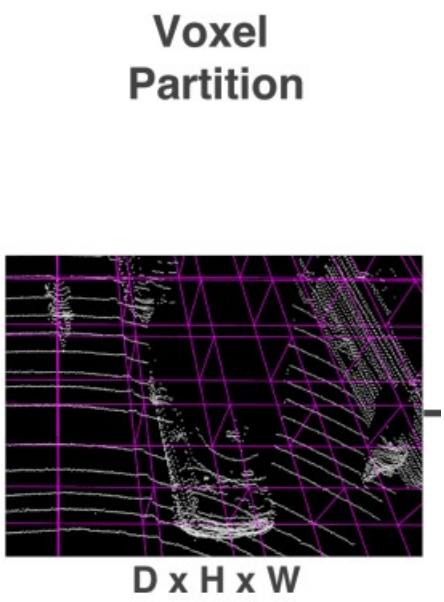
II. PointNet Architecture



VoxelNet의 구조는 1) Feature Learning Network, 2) Convolutional middle layers, 3) Region Proposal Network로 나뉜다.

1. Feature Learning Network

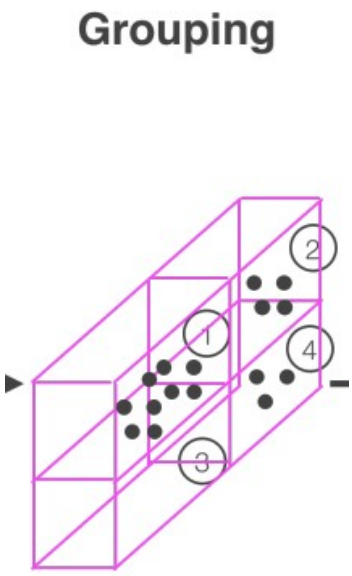
1) Voxel Partition



주어진 point cloud를 위의 그림처럼 D x H x W(z, y, x 축)의 개수를 각 voxel of size로 나눈다.

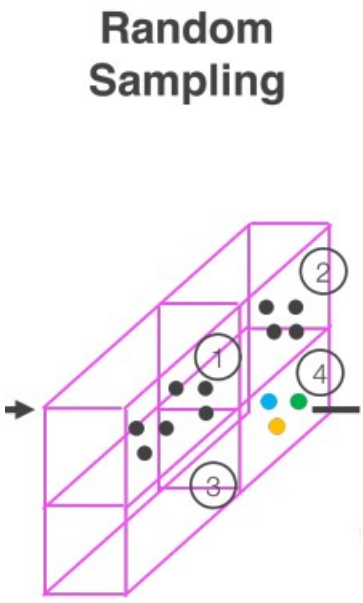
$$D' = D/v_D, H' = H/v_H, W' = W/v_W$$

2) Grouping



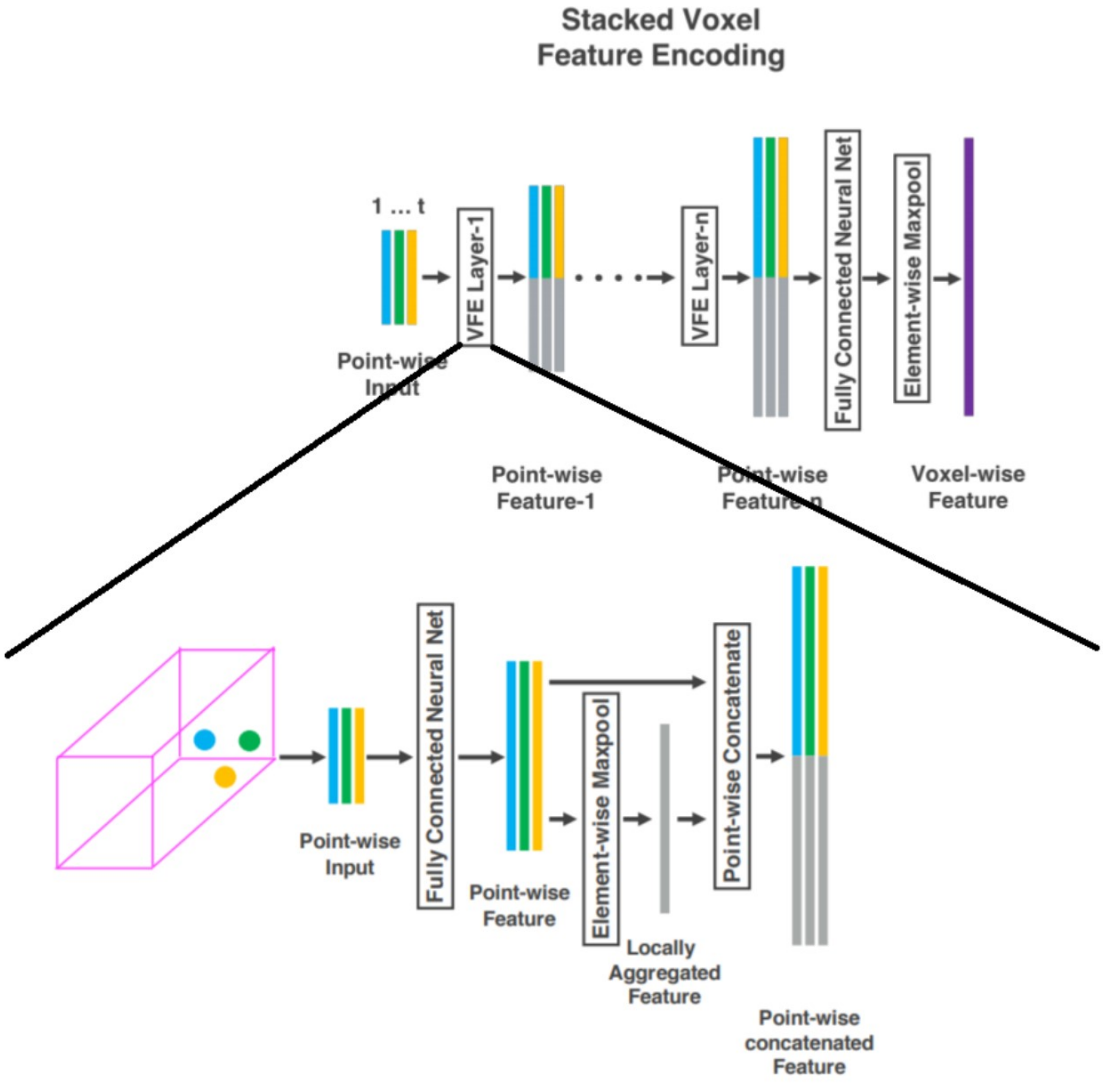
하나의 voxel grid 내에 있는 point를 같은 group으로 묶는다. Point cloud는 density의 variance가 심하고 sparse한 특성을 가지고 있기 때문에 voxel마다 point의 수가 다양하게 된다. 이러한 빈 voxel을 통한 연산을 줄이기 위해 grouping을 진행한다. Grouping 이후에는 voxel은 다양한 수의 point를 얻게 된다.

3) Random Sampling



일반적으로 LiDAR sensor로부터 얻은 point cloud는 1frame 당 10만개 정도의 point를 가지고 있다. 이는 computation, memory 부담이 심하고 point에 대한 imbalance가 심하게 된다. 이를 줄이기 위해 T개 이상의 point를 가진 voxel에서 T개의 point를 random하게 sample하여 computation을 줄이고 sampling bias를 줄인다.

4) Stacked Voxel Feature Encoding

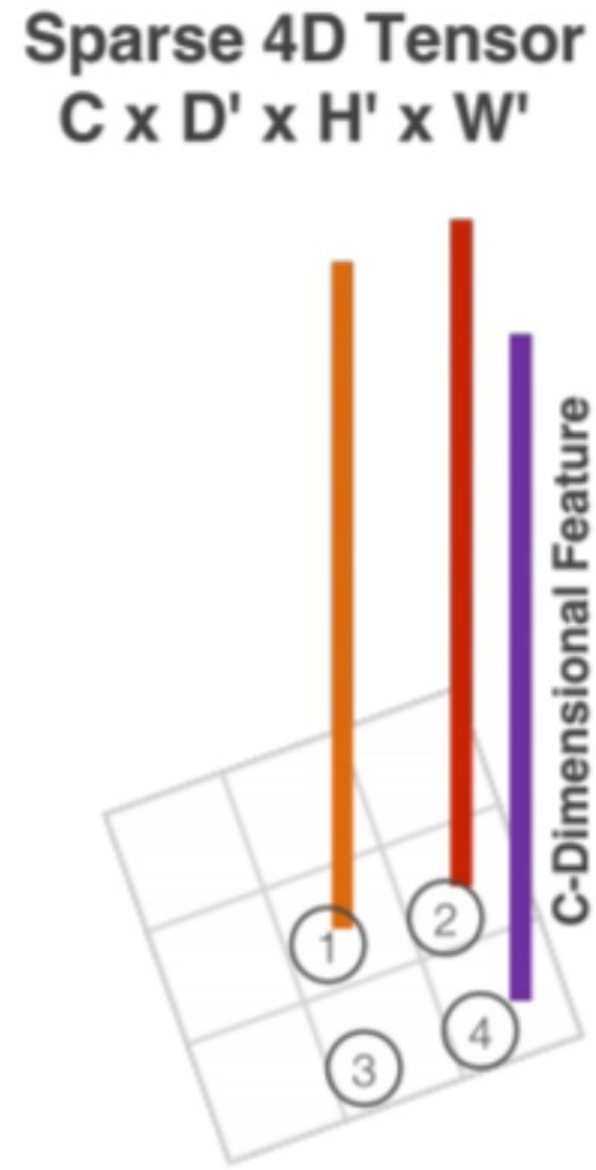


핵심은 연속적인 voxel feature encoding(VFE) layer에 있다. 먼저 random sampling된 점들의 local mean을 구하여 V의 centroid를 구한다. 그리고 각 포인트에 대해 centroid로부터의 상대적 위치를 feature로 augment하여 input feature set을 얻는다.

이를 FCN을 통해 point-wise feature를 얻고, MaxPooling을 통해 aggregate한다. 이 aggregate한 feature과 point-wise feature를 concatenated하여 output feature를 만든다.

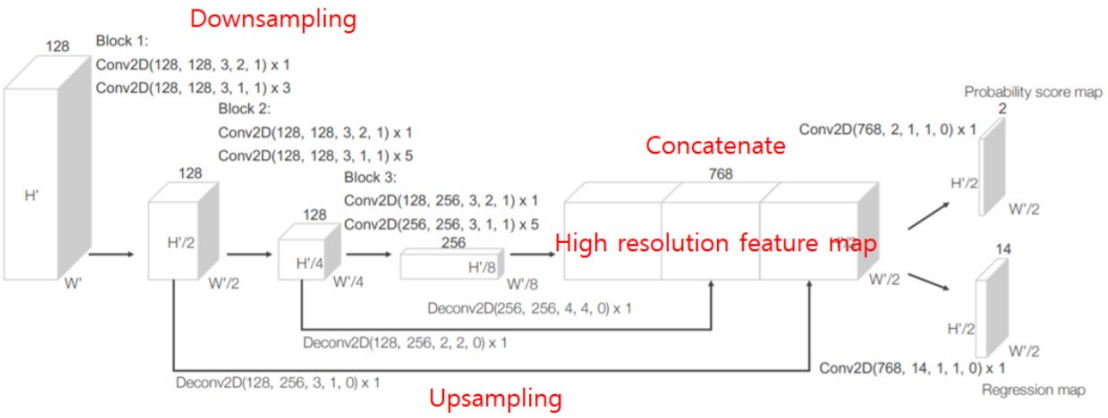
즉 output feature는 point-wise feature와 locally aggregated feature의 결합된 형태이기 때문에 voxel 내의 point들의 상관관계로부터 feature를 얻을 수 있다.

5) Sparse Tensor Representation



Non-empty voxel를 처리함으로써 voxel feature의 list를 얻을 수 있다. Point cloud가 10만개 정도의 point를 가지고 있지만 90% voxel은 비어있게 된다. 이를 통해 memory 사용량과 computation을 줄일 수 있다.

2. Convolutional Middle Layers

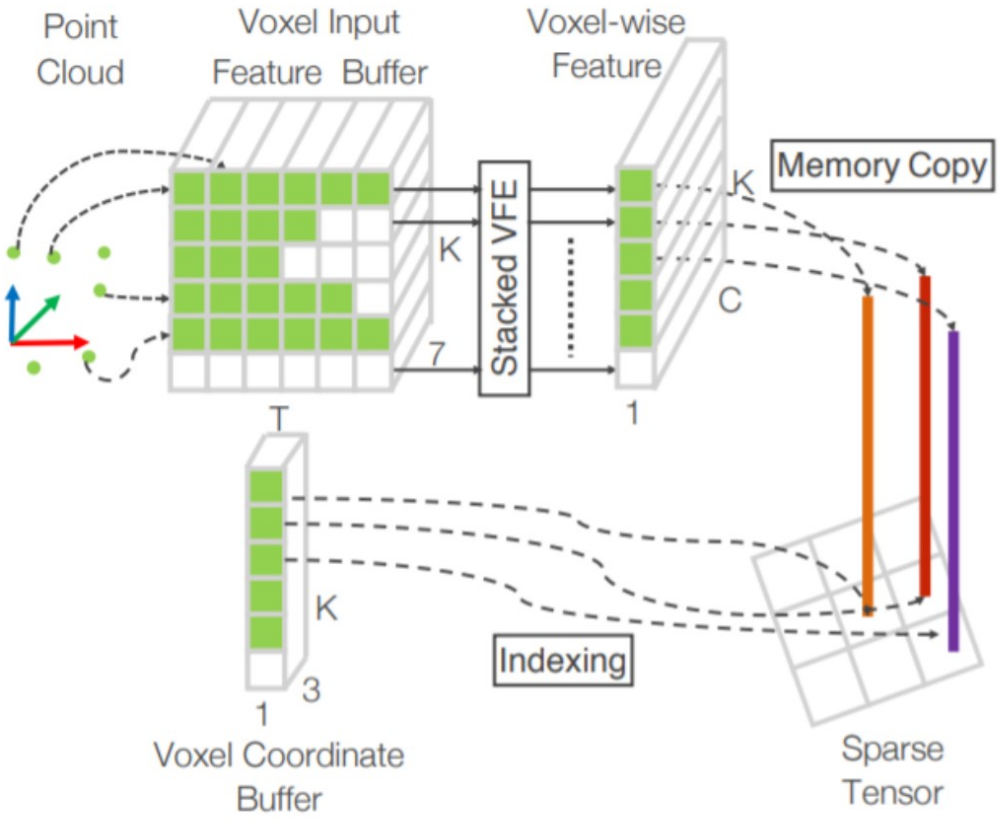


M-dimension conv operator를 표현하기 위해 ConvMD를 사용하였다. 각각 Conv2D, ReLU, BN으로 이루어져 있다.

3. Region Proposal Network

RPN은 높은 성능을 보이는 최근 object detection framework들에 있어서 중요하게 사용되는 network이다. RPN의 input은 ConvMD를 통해 얻을 수 있는 feature map이고, network는 3개의 FCN block으로 이루어져 있다. ouput은 Region Proposal이고 selective search의 역할을 대체한다. 즉 RPN은 bounding box를 추정하기 위한 알고리즘이다. 기본 Anchor box에서 크기와 위치를 조정하는 Delta를 통해 학습된다. 이를 통해 probability score map과 regression map 형태로 그려진다.

III. Efficient Implementation



GPU는 dense한 tensor를 처리하기에 최적화 되어있기 때문에 sparse한 point cloud를 처리하기에는 효율적이지 않다. 이를 해결하기 위해, point cloud를 VFE를 쌓은 dense한 tensor의 형태로 바꾸어 병렬 처리 방법을 고안하였다.

K x T x 7 크기의 tensor를 만들어 voxel input feature buffer를 저장하였고, 이 때 K는 비어있지 않은 voxel의 최대 개수, T는 voxel 당 들어있는 최대 point 수, 7은 각 point 당 가지는 input feature의 dimension이다.

IV. Conclusion

대부분 LiDAR 기반의 3D detection은 BEV projection과 같은 hand crafted feature에 의존했는데, 이 논문에서 manual feature engineering의 bottleneck을 제거하고 end-to-end로 학습 가능한 구조를 제안하였다. 또한 sparse한 point cloud와 voxel을 병렬적으로 처리하는 효율적인 방법을 제안하였다. 하지만 과도한 연산량으로 real-time에서 구현하기에는 한계가 보임을 알 수 있다.