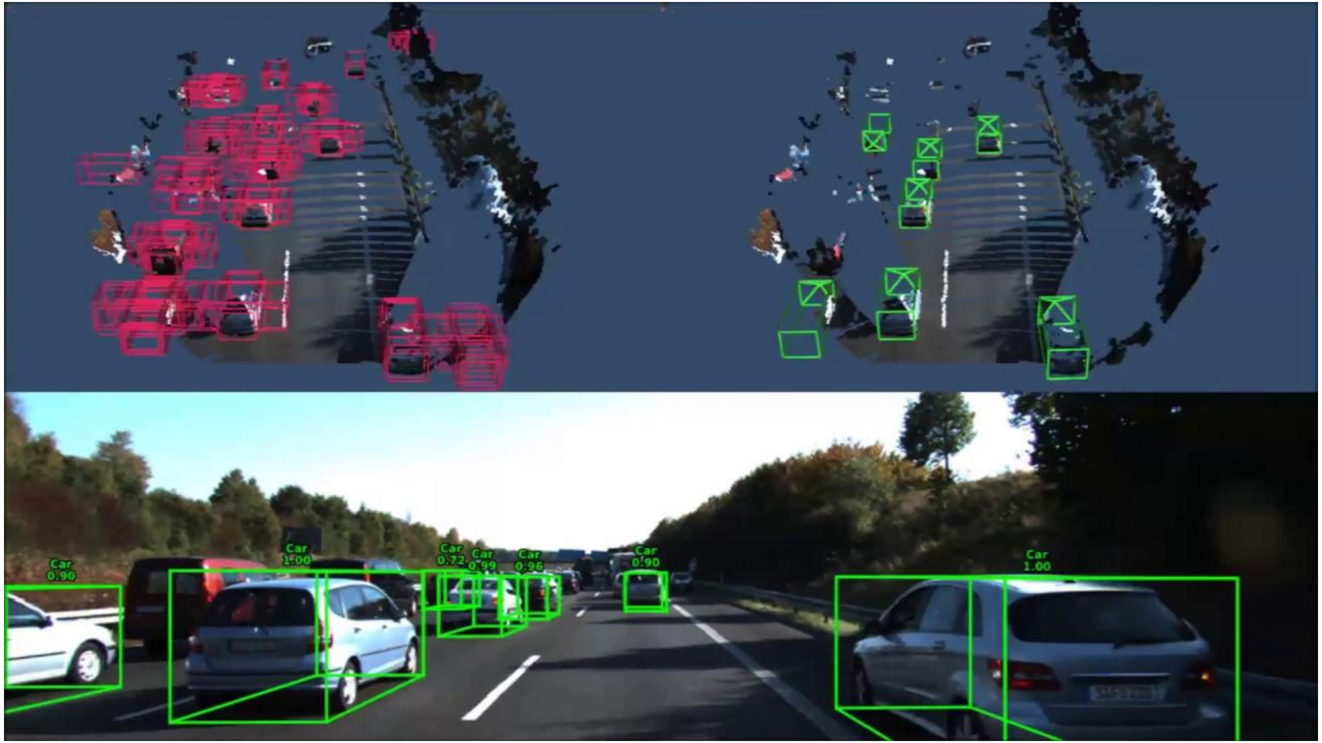
VOXELNET

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

Topics to be covered

- ➔ Object Detection in 3D space
- ➔ VoxelNet Architecture
- ➔ Loss Function
- ➔ Car Detection Example
- ➔ Results

Object Detection in 3D Space



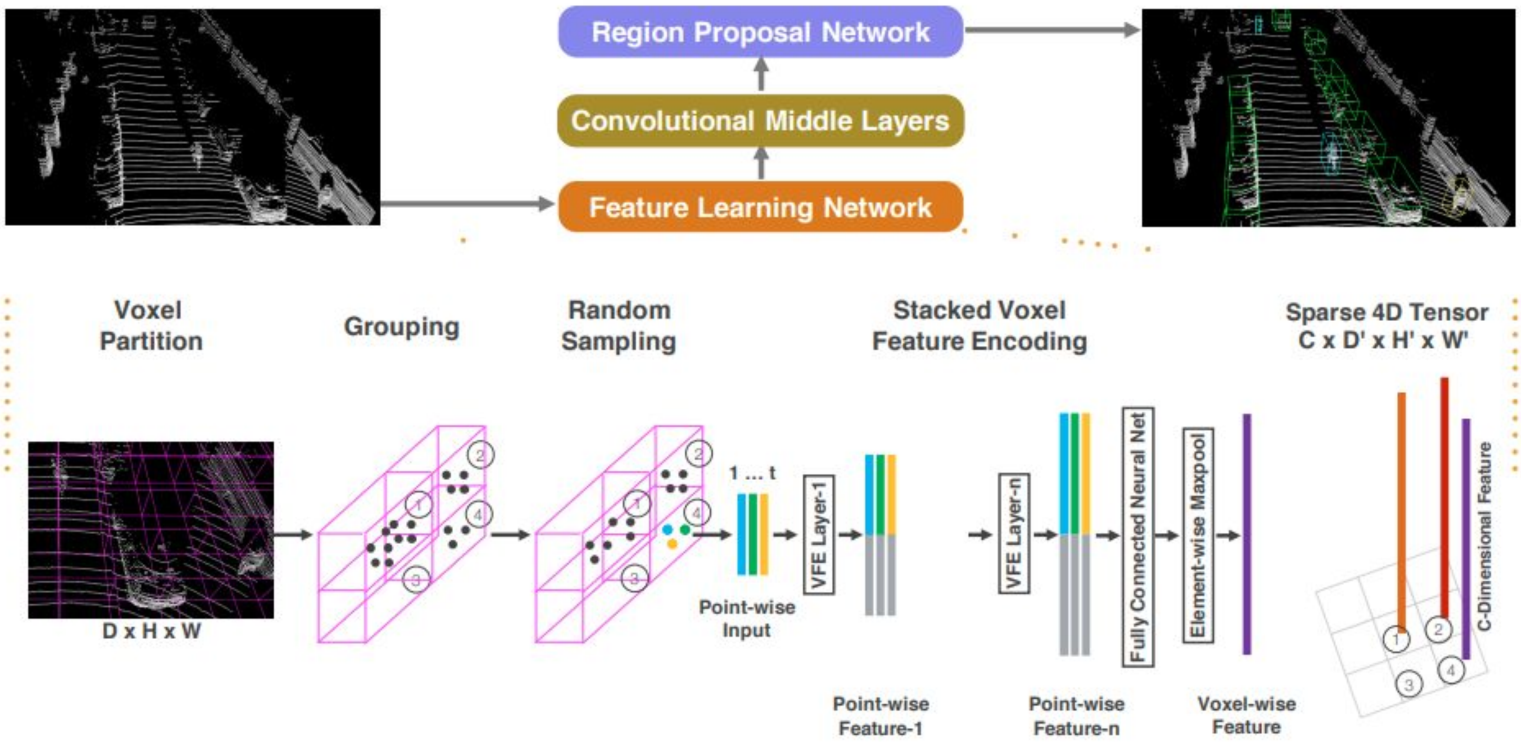
Object Detection in 3D Space

- ➔ The task is to identify the 3D objects in a scene and put a bounding box around them.
- ➔ Also, predict a probability score map for each of the bounding box.
- ➔ Idea is to initialize a set of **“Anchors”** and then refine them to get a bounding box with highest IOU among them.
- ➔ In this module, we will use point cloud representation of the 3D scene.

VoxelNet

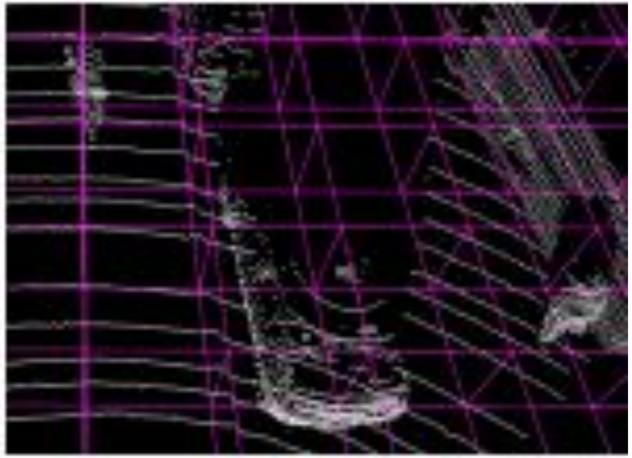
- VoxelNet was developed by **Yin Zhou** and **Oncel Tuzel** in 2017.
- Three main parts:
  - Feature Learning Network
  - Convolutional Middle Layers
  - Region Proposal Network
- Key feature of the architecture is VFE (Voxel Feature Encoding) layer.

VoxelNet Architecture



VoxelNet Architecture
Feature Learning Network

- **Voxel Partitioning :**
  - Suppose point cloud encompasses the 3D space with range D, H, W along the Z, Y, X axis respectively.
  - Define voxel size as **v_D , v_H and v_W.**
  - Sub-divide the 3D space into equally spaced voxels.
  - The resulting 3D voxel grid is of dimensions : **($D' = D/v_D$, $H' = H/v_H$, $W' = W/v_W$)**



D x H x W

VoxelNet Architecture
Feature Learning Network

➔ **Grouping :**
- 3D points are grouped based based on the voxel they reside in.
- Due to LiDAR scanning, point cloud is sparse and has highly variable point cloud density.
- Therefore, voxels will contain different number of points. Some voxels might be empty.

➔ **Random Sampling :**
- Randomly sample **T** points from each voxel (if voxel contains more than T points).
- Benefits? **1.** Computational Savings **2.** Reduction in imbalance of points among voxels.

VoxelNet Architecture
Feature Learning Network

➔ **Stacked Voxel Feature Encoding :**

- Key innovation is chain of VFE layers, which hierarchically perform feature encoding process for a voxel.

- Denote $\mathbf{V = \{ p_i = [x_i, y_i, z_i, r_i]^T \in R^4 \}_{i = 1,2,.....t}}$ as a voxel which contains **t<=T** points.

- Compute the mean $\mathbf{(v_x, v_y, v_z)}$ of all poits in V.

- Input feature set $\mathbf{V_{in} = \{ p_i' = [x_i, y_i, z_i, r_i, x_i\text{-}v_x, y_i\text{-}v_y, z_i\text{-}v_z]^T \in R^7 \}_{i = 1,2,.....t}}$

VoxelNet Architecture
Feature Learning Network

➔ **Stacked Voxel Feature Encoding :**

- Each $\mathbf{p_i}$**'** is transformed through the **Fully Connected Network** into a feature point $\mathbf{f_i \in R^m}$
- This FCN is composed of a linear layer, and a batch norm layer with ReLu activation function.
- Then, Maxpooling is applied element-wise to get a locally aggregated feature $\mathbf{f' \in R^m}$ for the voxel **V.**
- Define $\mathbf{f^{out} = [\, f_i^T, f'^T \,]^T \in R^{2m}}$
- $\mathbf{V_{out} = \{ f_i^{out} \}_{i = 1,2,.....t}}$

VoxelNet Architecture
Feature Learning Network

- **Stacked Voxel Feature Encoding :**
  - **VFE-i(c_{in}, c_{out})** is the i-th VFE layer which transforms input features of dimension c_{in} into output features of dimension c_{out}.
  - The linear layer learns a matrix of size $\mathbf{c_{in} \times (c_{out}/2)}$
  - The output of VFE-n is transformed into $\mathbf{R^C}$ using an FCN and applying Maxpooling. (C is the dimension of voxel-wise feature)

VoxelNet Architecture
Feature Learning Network

- **Stacked Voxel Feature Encoding :**
  - **VFE-i(c_{in}, c_{out})** is the i-th VFE layer which transforms input features of dimension c_{in} into output features of dimension c_{out}.
  - The linear layer learns a matrix of size $\mathbf{c_{in} \times (c_{out}/2)}$
  - The output of VFE-n is transformed into $\mathbf{R}^{C}$ using an FCN and applying Maxpooling. (C is the dimension of voxel-wise feature)
- **Sparse Tensor Representation :**
  - The list of voxel-wise features can be represented as a sparse 4D tensor of size **C x D' x H' x W'.**
  - This sparse tensor reduces the memory usage.

VoxelNet Architecture

Feature Learning Network

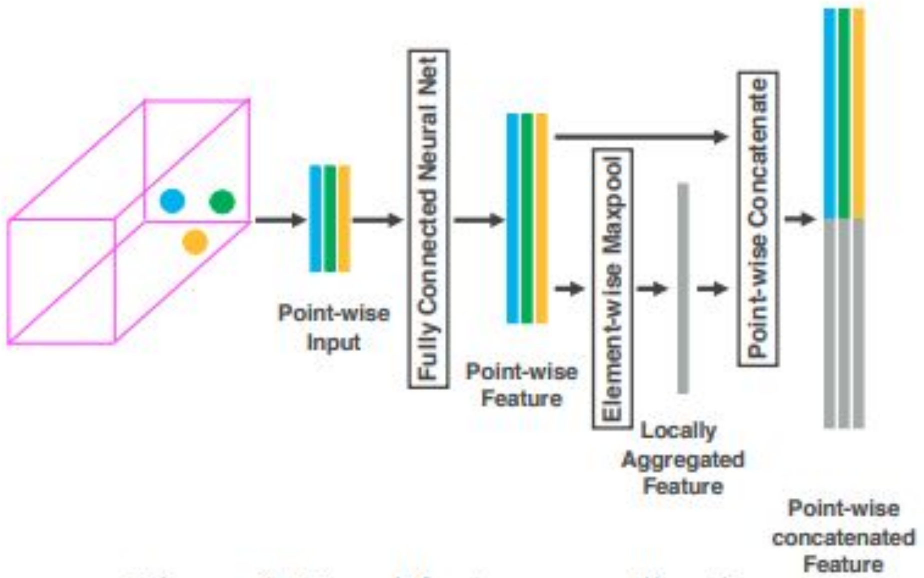
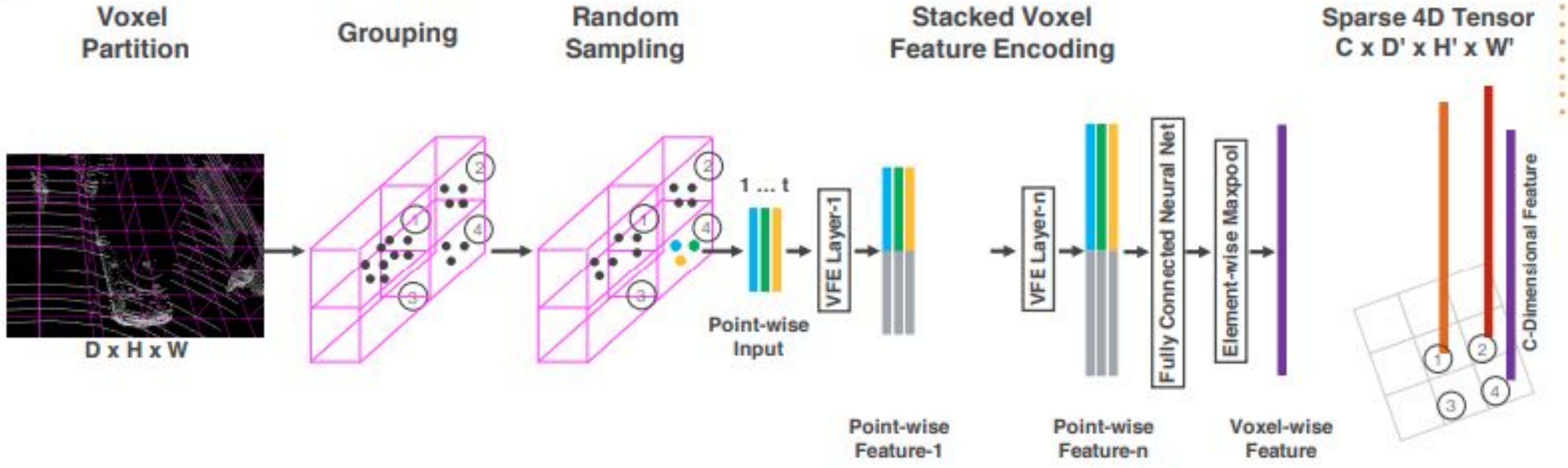


Figure 3. Voxel feature encoding layer.

VoxelNet Architecture
Feature Learning Network



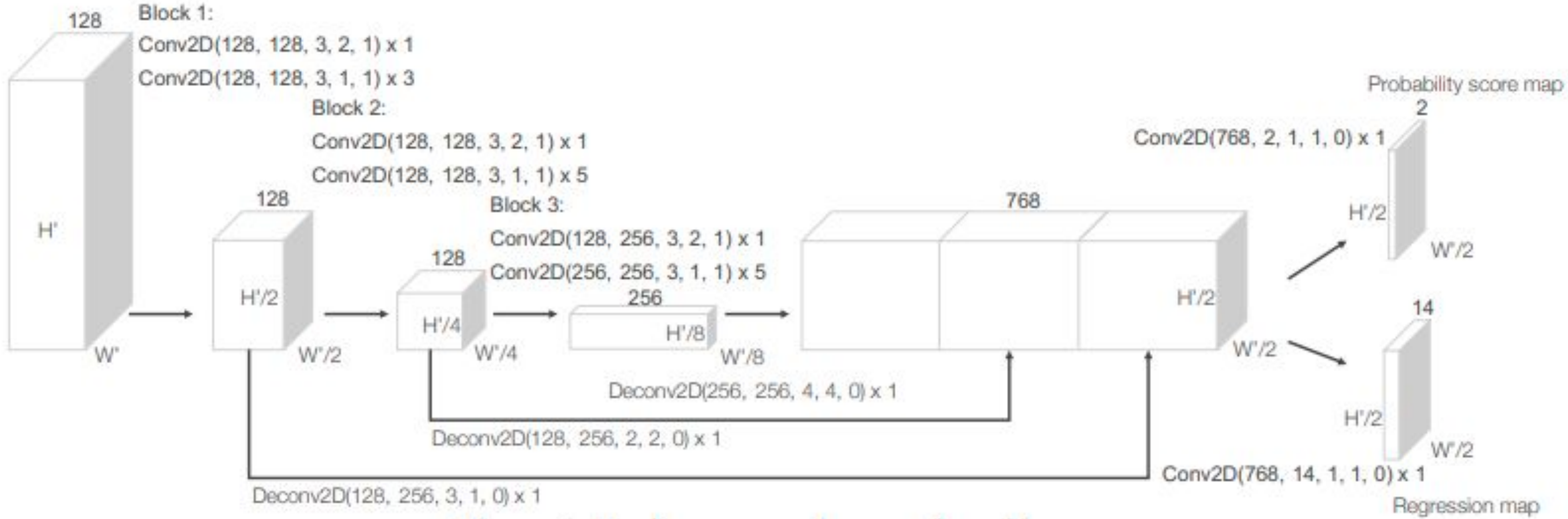
VoxelNet Architecture
Convolutional Middle Layers

- **ConvMD(c_{in}, c_{out}, k, s, p)** an M-dimensional Convolutional Operator, where c_{in} and c_{out} are the number of input and output channels.
- Each convolutional layer applies 3D convolution, Batch Norm, and ReLu sequentially.

VoxelNet Architecture
Region Proposal Network

- ➔ The input to RPN is the feature map provided by final convolutional layer.
- ➔ The RPN has 3 blocks of fully convolutional layers.
- ➔ The first layer of each map downsamples the featue map by half via a convolution with a stride size of 2.
- ➔ It is followed by the convolutions of stride 1.
- ➔ **×q** means applying the filter q times.
- ➔ Then, BN and ReLU is applied.
- ➔ Output of each block is then upsampled to a fixed size and concatenated to form a feature map.
- ➔ Finally, this feature map is mapped to the desired learnings: **1.** Probability score map **& 2.** Regression map.

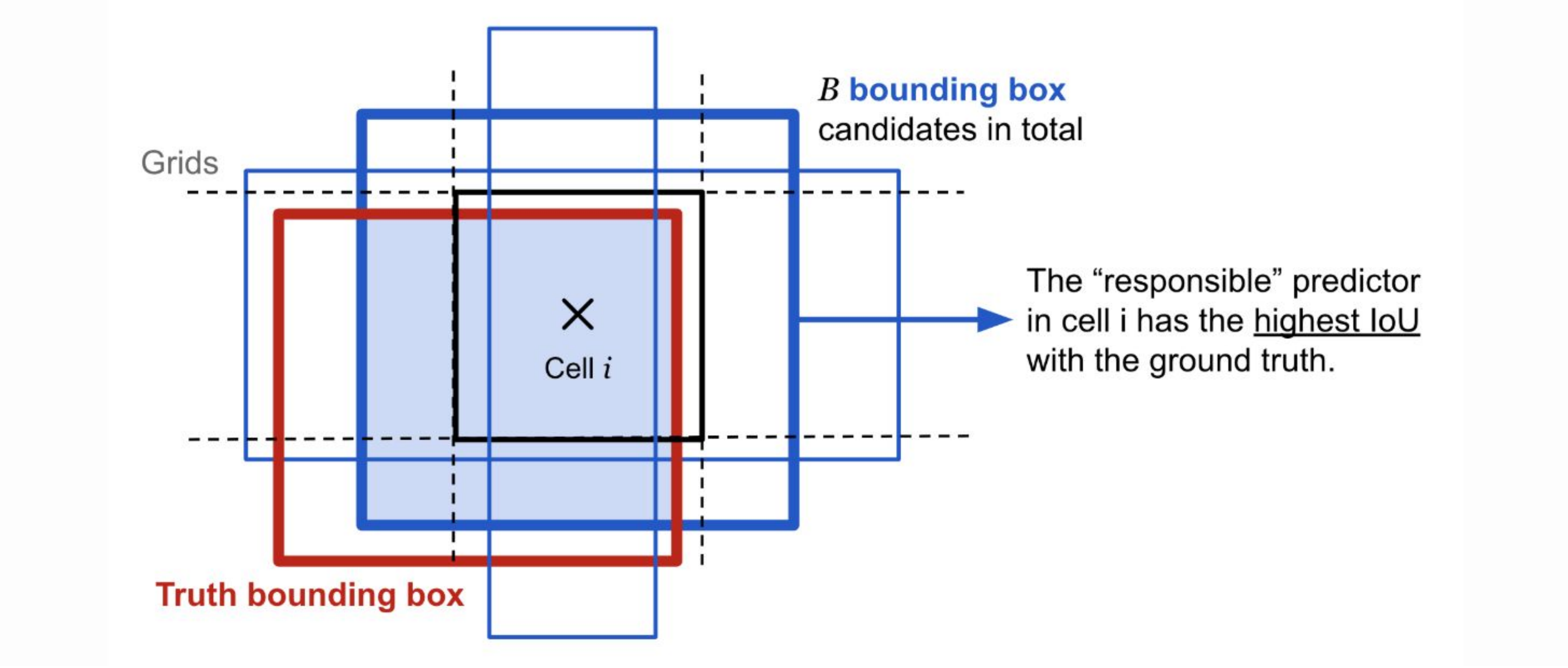
VoxelNet Architecture
Region Proposal Network



Loss Function

Anchors

Loss Function
Anchors



Loss Function
Formulation

- ➔ Let $\{ a_i^{pos} \}_{i = 1,2,.....Npos}$ be the set of N_{pos} positive anchor boxes.
- ➔ Similarly, let $\{ a_i^{neg} \}_{i = 1,2,.....Nneg}$ be the set of N_{neg} negative anchor boxes.
- ➔ 3D ground truth bounding box is parametrized as $(x_c^g, y_c^g, z_c^g, l^g, w^g, h^g, \theta^g)$.
- ➔ Anchors is parametrized as $(x_c^a, y_c^a, z_c^a, l^a, w^a, h^a, \theta^a)$.
- ➔ Define a residual vector $u^* \in R^7$, $u^* = [\Delta x, \Delta y, \Delta z, \Delta l, \Delta w, \Delta h, \Delta\theta]^T$.

Loss Function

Formulation

$$\Delta x = \frac{x_c^g - x_c^a}{d^a}, \Delta y = \frac{y_c^g - y_c^a}{d^a}, \Delta z = \frac{z_c^g - z_c^a}{h^a},$$

$$\Delta l = \log(\frac{l^g}{l^a}), \Delta w = \log(\frac{w^g}{w^a}), \Delta h = \log(\frac{h^g}{h^a}),$$

$$\Delta\theta = \theta^g - \theta^a$$

where $d^a = \sqrt{(l^a)^2 + (w^a)^2}$

(diagonal of the base of the anchor box)

Loss Function

Formulation

$$L = \alpha \frac{1}{N_{\text{pos}}} \sum_i L_{\text{cls}}(p_i^{\text{pos}}, 1) + \beta \frac{1}{N_{\text{neg}}} \sum_j L_{\text{cls}}(p_j^{\text{neg}}, 0) + \frac{1}{N_{\text{pos}}} \sum_i L_{\text{reg}}(\mathbf{u}_i, \mathbf{u}_i^*)$$

➔ L_{cls} is the binary cross entropy loss.

➔ L_{reg} is the L1 loss.

Car Detection

- ➔ Point clouds are within the range **[-3,1] x [-40,40] x [0,70.4] metres** along Z, Y and X axis respectively.
- ➔ $v_D = 0.4$, $v_H = 0.2$, $v_W = 0.2$
- ➔ **D' = 10, H' = 400, W' = 352**
- ➔ **T = 35**
- ➔ Two VFE-layers **VFE-1(7,32)** and **VFE-2(32,128)**
- ➔ FCN maps VFE-2's ouptu to $\mathbf{R}^{128}$
- ➔ Generated sparse tensor is of dimension **128 x 10 x 400 x 352**

Car Detection

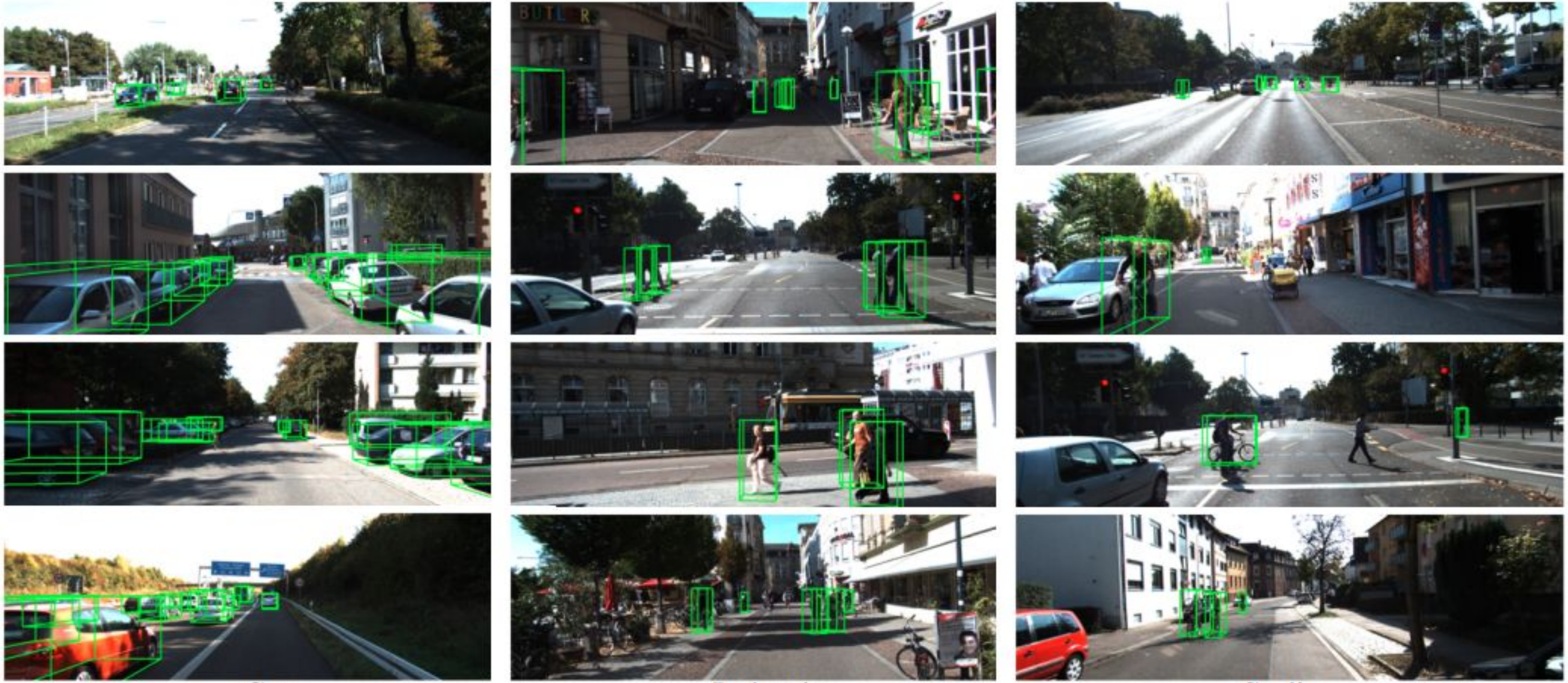
- 3 Conv3D layers are used :
  - Conv3D(128, 64, 3, (2,1,1), (1,1,1))
  - Conv3D(64, 64, 3, (1,1,1), (0,1,1))
  - Conv3D(64, 64, 3, (2,1,1), (1,1,1))

- Output is 64 x 2 x 400 x 352, which is reshaped into 128 x 400 x 352 and feeded to RPN.

- Only one anchor size is used : $l^a = 3.9$, $w^a = 1.6$, $h^a = 1.56$ metres

- **α=1.5, β=1**

Results



Car Pedestrian Cyclist

Results

Benchmark	Easy	Moderate	Hard
Car (3D Detection)	77.47	65.11	57.73
Car (Bird's Eye View)	89.35	79.26	77.39
Pedestrian (3D Detection)	39.48	33.69	31.51
Pedestrian (Bird's Eye View)	46.13	40.74	38.11
Cyclist (3D Detection)	61.22	48.36	44.37
Cyclist (Bird's Eye View)	66.70	54.76	50.55