Weekly seminar

# SECOND: Sparsely Embedded Convolutional Detection

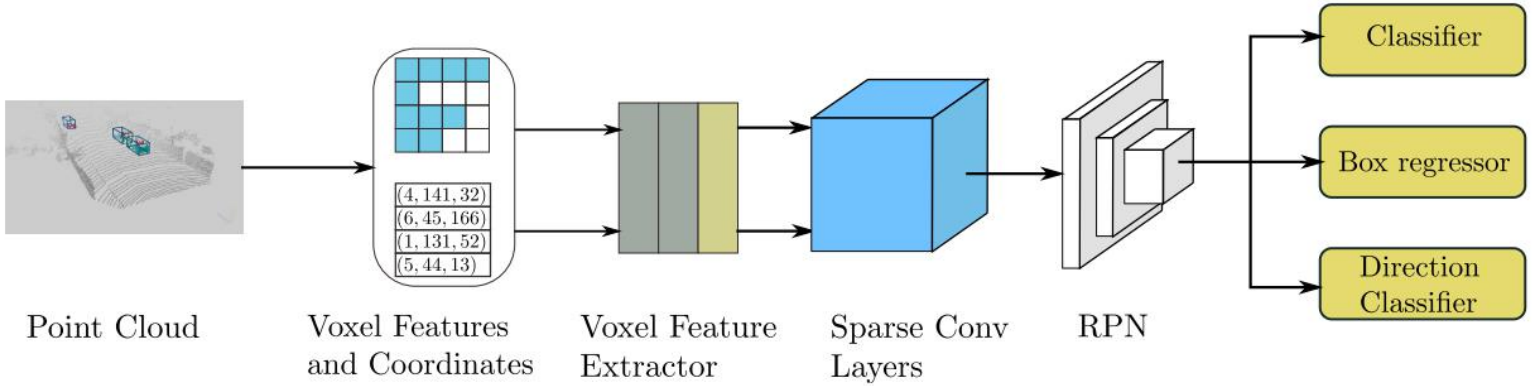
vietanhdev

# Table of Contents

# 1/ What?

## What is SECOND?

LiDAR 3D voxel-based object detection network




**The structure of our proposed SECOND detector.**
Source: SECOND paper

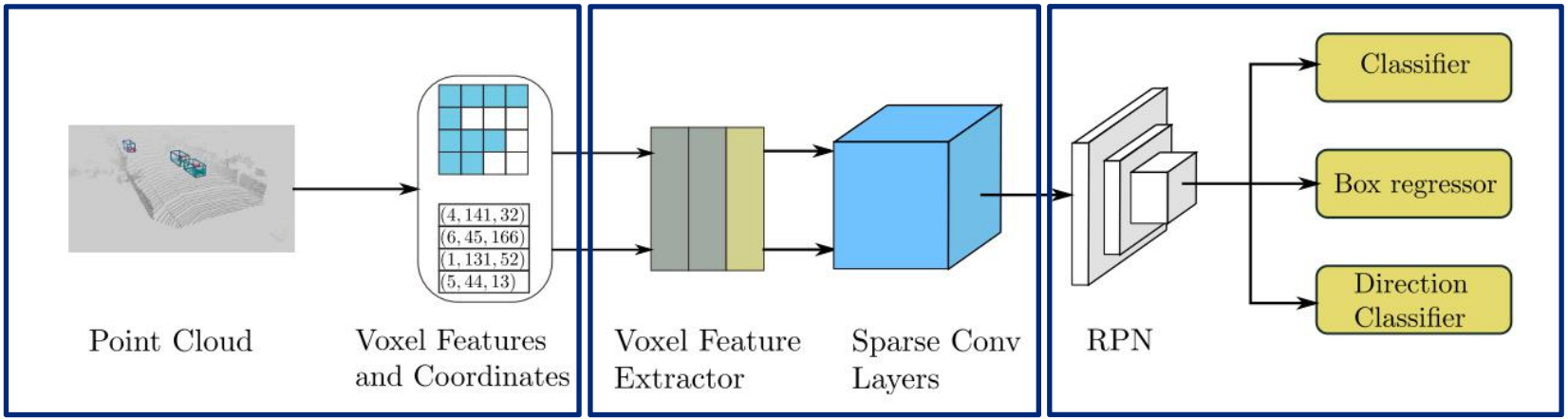
# 1/ What?

**Main contributions**

- A Voxel-based 3D object detection network which applies **an improved version of sparse convolution layers.**
- **A new angle loss** function to improve the orientation estimation.
- **A new data augmentation method** for point cloud to enhance the convergence speed and the performance.

# 2/ How?

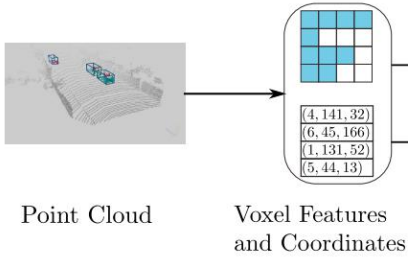
## Main steps




(1) Point cloud grouping (2) Feature extraction (3) Detection generation

# 2/ How?

## (1) Point cloud grouping




**Hyperparams:** Crop sizes, Max # of voxels, Voxel size

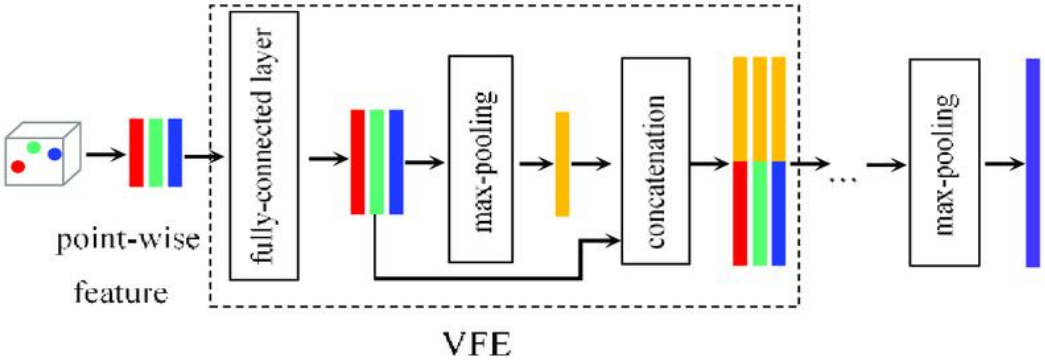
Crop the point cloud based on the ground-truth distribution **z** x **y** x **x**

→

Assign points into associated voxels (Limit # of voxels here)

# 2/ How?

## (2) Feature extraction - VFE




**Structure of voxel feature encoding layer**
Source: https://www.researchgate.net/figure/Structure-of-voxel-feature-extraction-network_fig2_338876233

# 2/ How?

## (2) Feature extraction - Sparse Conv.

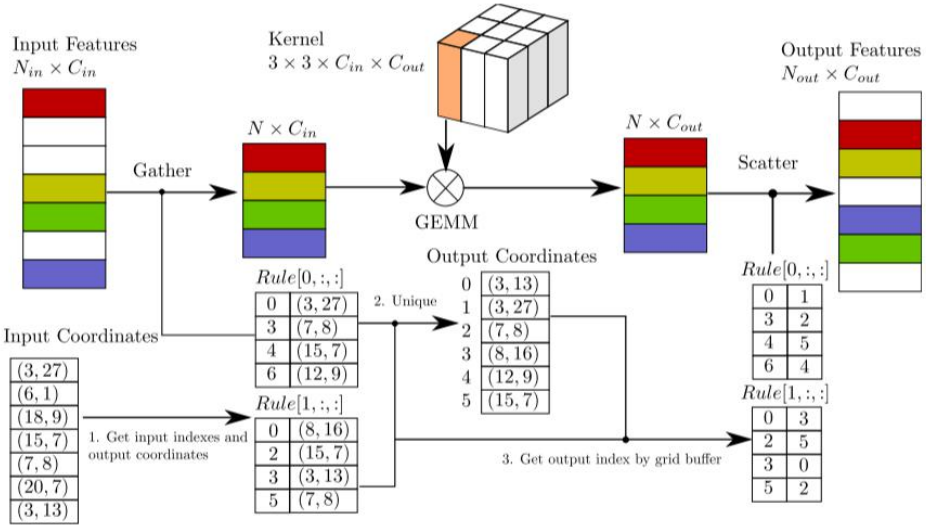



**Figure 2.** The sparse convolution algorithm is shown above, and the GPU rule generation algorithm is shown below. $N_{in}$ denotes the number of input features, and $N_{out}$ denotes the number of output features. $N$ is the number of gathered features. *Rule* is the rule matrix, where $Rule[i,:,:]$ is the $i$-th rule corresponding to the $i$-th kernel matrix in the convolution kernel. The boxes with colors except white indicate points with sparse data and the white boxes indicate empty points.

CPU-based Rule Generation Algorithm using a hash table is **slow** and requires **CPU-GPU data transfer.**

➠ **Need to invent a New Rule Generation Algorithm on GPU**

# 2/ How?

## (2) Feature extraction - Sparse Conv.

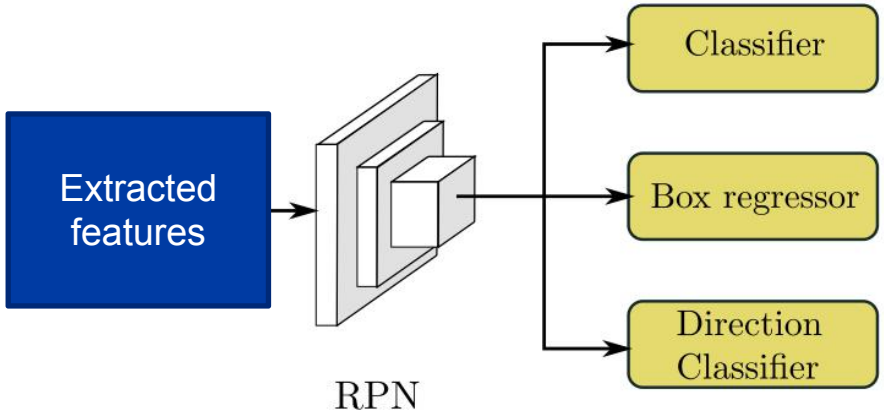
New Rule Generation Algorithm (RGA) on GPU

**Table 1.** Comparison of the execution speeds of various convolution implementations. SparseConvNet is the official implementation of submanifold convolution [27]. All benchmarks were run on a GTX 1080 Ti GPU with the data from the KITTI dataset.

| Sparse Convolution (1 layer) | | | |
|---|---|---|---|
| **Channels** | **SECOND** | **SpConvNet [31]** | **Dense** |
| $64 \times 64$ | 8.6 | 21.2 | 567 |
| $128 \times 128$ | 13.8 | 24.8 | 1250 |
| $256 \times 256$ | 25.3 | 37.4 | $N/A$ |
| $512 \times 512$ | 58.7 | 86.0 | $N/A$ |
| **Submanifold Convolution (4 layers)** | | | |
| **Channels** | **SECOND** | **SpConvNet [31]** | **Dense** |
| $64 \times 64$ | 7.1 | 16.0 | $N/A$ |
| $128 \times 128$ | 11.3 | 21.5 | $N/A$ |
| $256 \times 256$ | 20.4 | 37.0 | $N/A$ |
| $512 \times 512$ | 49.0 | 94.1 | $N/A$ |

# 2/ How?

## (3) Detection generation: SSD-based RPN with anchors



# 2/ How?

## Sine-Error Loss for Angle Regression

$$L_\theta = \text{SmoothL1}(\sin(\theta_p - \theta_t)),$$

- (1) it solves the adversarial example problem between orientations of 0 and π
- (2) it naturally models the IoU against the angle offset function

This loss treats boxes with opposite directions as being the same?

➠ **Add a direction classifier using a softmax loss**

# 2/ How?

**Focal Loss for Classification**

$$FL(p_t) = -\alpha_t(1 - p_t)^{\gamma} \log(p_t),$$

# 2/ How?

## Multitask Loss

$$L_{total} = \beta_1 L_{cls} + \beta_2 (L_{reg-\theta} + L_{reg-other}) + \beta_3 L_{dir},$$

where $L_{cls}$ is the classification loss,
$L_{reg-other}$ is the regression loss for location and dimension,
$L_{reg-\theta}$ is the angle loss,
and $L_{dir}$ is the direction classification loss

# 2/ How?

**Data augmentation:**

3 main methods for data augmentation

- **(1) Sample Ground Truths from the Database:** copy object points and labels from ground truth to training point clouds. Check collision to prevent impossible outcomes
- **(2) Object Noise:** augment each object independently with random rotations and linear transformation
- **(3) Global Rotation and Scaling**

# 3/ Ideas

- **Try anchor-free RPN?**
- **Reproduce dense properties in data augmentation?**

# References


- **SECOND** https://www.researchgate.net/publication/328158485_SECOND_Sparsely_Embedded_Convolutional_Detection
- **VoxelNet** https://ieeexplore.ieee.org/document/8578570
- **Apple’s new self-driving car tech: Voxelnet is quite Awesome** https://www.techexplorist.com/apples-new-self-driving-car-tech-voxelnet-quite-awesome/8925/

# Thanks

Do you have any questions?