PointPillars: Complete Point Cloud to Dense Tensor Implementation

Anuj Patel

July 28, 2025

1 Introduction

This report presents a complete implementation of "PointPillars: Fast Encoders for Object Detection from Point Clouds" from the PointPillars paper, including both 9D feature augmentation and dense tensor creation for neural network processing.

2 2.1. Pointcloud to PseudoImage

2.1 Objective

Convert sparse 3D LiDAR point clouds into 9-dimensional feature vectors and create dense tensors suitable for PointNet processing, following the PointPillars methodology.

2.2 Phase 1: Grid Discretization and Feature Augmentation

Given input point cloud $\mathcal{P} = \{p_1, p_2, \ldots, p_N\}$ where $p_i = (x_i, y_i, z_i, r_i)$:

2.2.1 Step 1: Grid Discretization

Discretize the x-y plane into pillars with resolution $\Delta x = \Delta y = 0.16$ m:

$$\text{pillar}_x = \left\lfloor \frac{x}{\Delta x} \right\rfloor, \quad \text{pillar}_y = \left\lfloor \frac{y}{\Delta y} \right\rfloor \tag{1}$$

Implementation Example:

$$\text{Point } (18.324, 0.049) \rightarrow \left(\left\lfloor \frac{18.324}{0.16} \right\rfloor, \left\lfloor \frac{0.049}{0.16} \right\rfloor \right) = (114, 0) \tag{2}$$

$$\text{Point } (51.299, 0.505) \rightarrow \left(\left\lfloor \frac{51.299}{0.16} \right\rfloor, \left\lfloor \frac{0.505}{0.16} \right\rfloor \right) = (320, 3) \tag{3}$$

2.2.2 Step 2: Pillar Center Calculation

For each pillar $(\text{pillar}_x, \text{pillar}_y)$:

$$\text{center}_x = \text{pillar}_x \times \Delta x + \frac{\Delta x}{2}, \quad \text{center}_y = \text{pillar}_y \times \Delta y + \frac{\Delta y}{2} \tag{4}$$

Implementation Example:

$$\text{Pillar } (114, 0) \rightarrow \text{center} = (114 \times 0.16 + 0.08, 0 \times 0.16 + 0.08) = (18.32, 0.08) \tag{5}$$

$$\text{Pillar } (320, 3) \rightarrow \text{center} = (320 \times 0.16 + 0.08, 3 \times 0.16 + 0.08) = (51.28, 0.56) \tag{6}$$

2.2.3 Step 3: Arithmetic Mean Computation

For each pillar containing k points:

$$\bar{x} = \frac{1}{k}\sum_{j=1}^{k} x_j, \quad \bar{y} = \frac{1}{k}\sum_{j=1}^{k} y_j, \quad \bar{z} = \frac{1}{k}\sum_{j=1}^{k} z_j \tag{7}$$

2.2.4 Step 4: Feature Augmentation

Transform each point to 9D vector:

$$\mathbf{f}_i = [x_i, y_i, z_i, r_i, x_{c_i}, y_{c_i}, z_{c_i}, x_{p_i}, y_{p_i}] \tag{8}$$

where:

$$x_{c_i} = x_i - \bar{x}, \quad y_{c_i} = y_i - \bar{y}, \quad z_{c_i} = z_i - \bar{z} \quad \text{(centroid-relative)} \tag{9}$$

$$x_{p_i} = x_i - \text{center}_x, \quad y_{p_i} = y_i - \text{center}_y \quad \text{(pillar-center-relative)} \tag{10}$$

3 Phase 2: Dense Tensor Creation for Sparsity Handling

3.1 Sparsity Problem

LiDAR point clouds exhibit extreme sparsity. For KITTI dataset with 0.16^2 m^2 bins:

- Total possible pillars: ∼160,000
- Non-empty pillars: 6k-9k (∼3% occupancy)
- Points per pillar: Highly variable (1-300+ points)

3.2 Dense Tensor Formulation

Create fixed-size tensor $\mathbf{T} \in \mathbb{R}^{P \times N \times D}$ where:

$$P = \text{Maximum pillars per sample} = 12000 \tag{11}$$
$$N = \text{Maximum points per pillar} = 32 \tag{12}$$
$$D = \text{Feature dimensions} = 9 \tag{13}$$

Why fixed size? Neural networks need regular input, but LiDAR data is irregular:

- Real data: 16,249 pillars with 1-369 points each
- Fixed tensor: 12,000 pillars with exactly 32 points each

3.3 Sampling and Padding Algorithm

3.3.1 Step 1: Pillar-Level Sampling

Given M non-empty pillars, select exactly $P = 12000$ pillars:

$$\text{Selected pillars} = \begin{cases} \text{randomly pick } P \text{ pillars} & \text{if } M > P \\ \text{use all } M \text{ pillars} & \text{if } M \leq P \end{cases} \tag{14}$$

Code example:

```
if len(pillar_ids) > 12000:
    selected_pillars = random.sample(pillar_ids, 12000)  # Sample
else:
    selected_pillars = pillar_ids                        # Keep all
```

3.3.2 Step 2: Point-Level Sampling

For each pillar with n_i points, create exactly $N = 32$ points:

$$\text{Final points} = \begin{cases} \text{randomly pick } N \text{ points} & \text{if } n_i > N \\ \text{use all points + zeros} & \text{if } n_i \leq N \end{cases} \tag{15}$$

Code example:

```
if len(points_in_pillar) > 32:
    sampled_points = random.sample(points_in_pillar, 32)  # Sample
else:
    sampled_points = points_in_pillar                     # Keep all + padding
```

Result: Perfect rectangle of shape $(12000, 32, 9)$ ready for neural network processing.