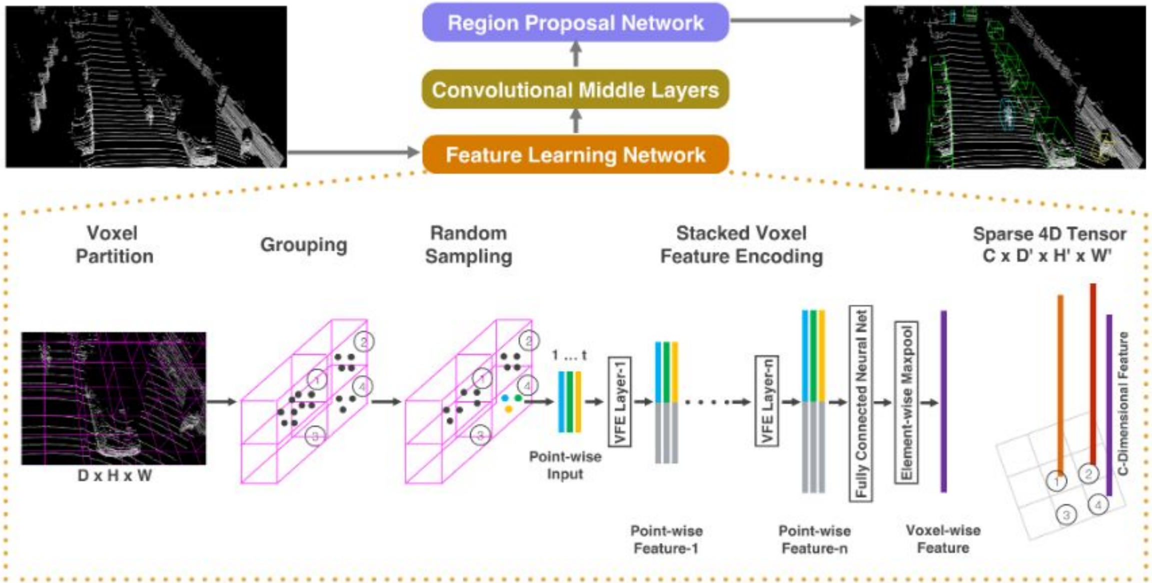
3D obstacle detection

Pointpillars

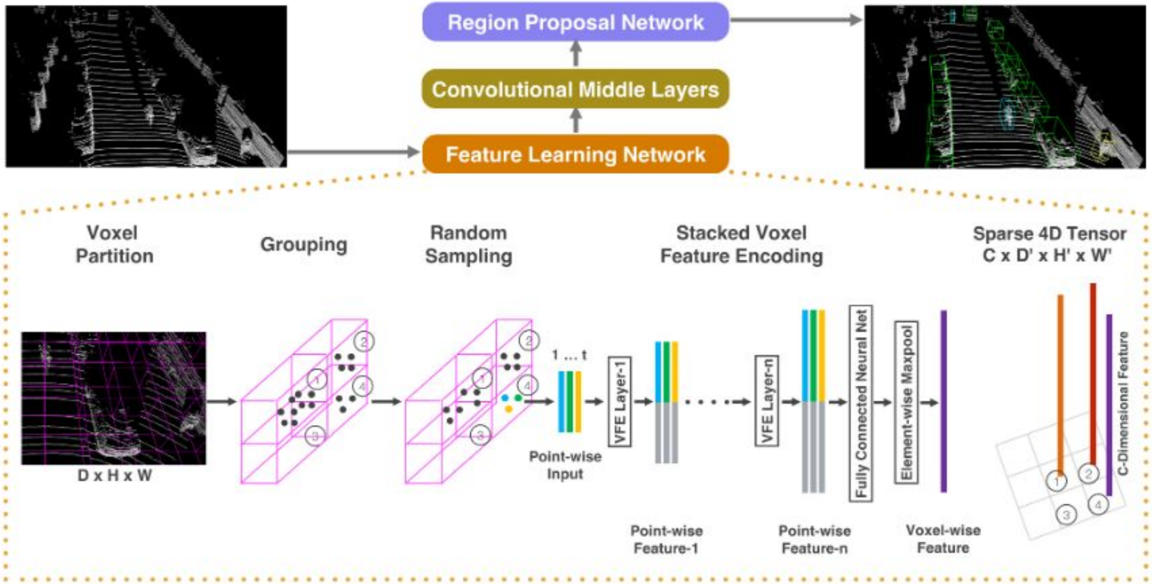
Introduction

- LiDAR
  - Reliable depth estimation
  - Accurate localization
- Very sparse, variable density
- Region Proposal Network
  - Highly optimized for efficient object detection
  - Needs input in the form of images !!
- Backbone needs to extract features reliably in pseudo-image form

VoxelNet


Region Proposal Network
Convolutional Middle Layers
Feature Learning Network
Voxel Partition
Grouping
Random Sampling
Stacked Voxel Feature Encoding
Sparse 4D Tensor C x D' x H' x W'
D x H x W
1 ... t
Point-wise Input
VFE Layer-1
VFE Layer-n
Fully Connected Neural Net
Element-wise Maxpool
Point-wise Feature-1
Point-wise Feature-n
Voxel-wise Feature
C-Dimensional Feature
1
2
3
4

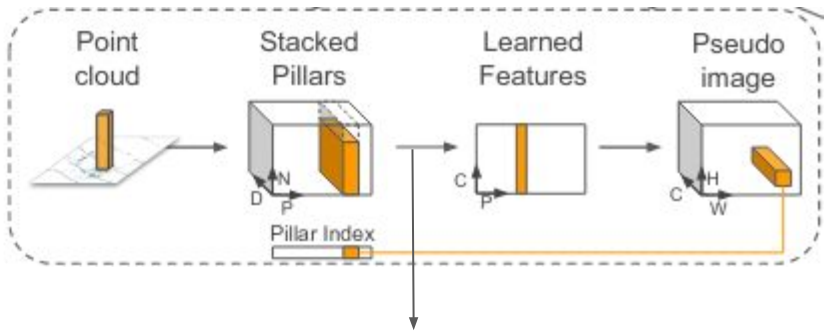
VoxelNet



- 3D convolutions in the middle layers make it too slow !

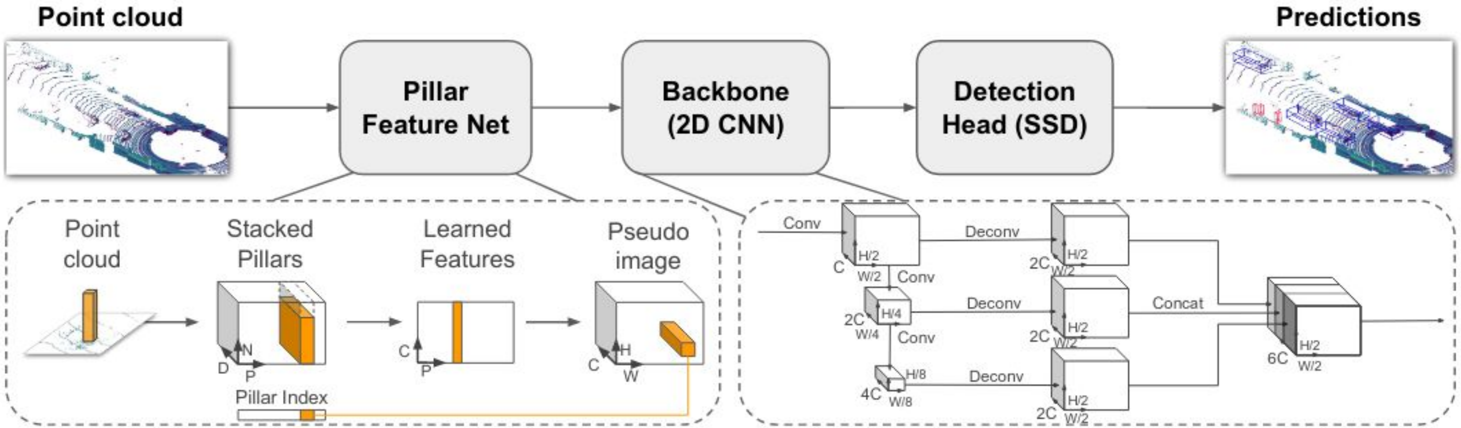
PointPillars

- Instead of voxels, learn features on pillars
- P: Number of non-empty pillars
- N: Max number of points per pillar
- D: Augmented lidar point dimension (=9)
- No 3D convolutions !!
- Very fast! (4.4 Hz -> 100 Hz)



PointNet -> Pillar-wise Max

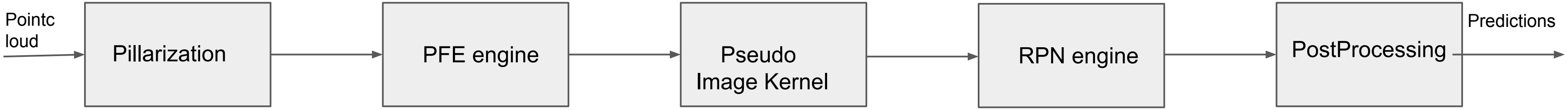
Pointpillars architecture



Runtime deployment: model and CUDA kernels

We use onnx as the model parser and use tensorRT for inference

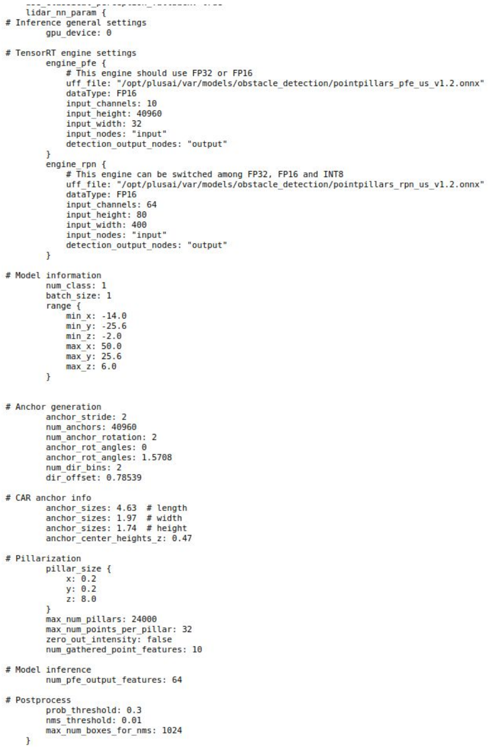
Sadly not all operations during inference are onnx compatible:
Pillarization, Pseudo Image generation, Postprocessing need their own CUDA kernels



All kernels are present: *perception/obstacle_detection/src/lidar/point_pillars/*.cu*

Runtime deployment: model config

https://github.com/PlusAI/drive/blob/master/perception/config/fusion_tracker/fusion_tracker.prototxt.paccar



```
    lidar_nn_param {
# Inference general settings
        gpu_device: 0

# TensorRT engine settings
        engine_pfe {
            # This engine should use FP32 or FP16
            uff_file: "/opt/plusai/var/models/obstacle_detection/pointpillars_pfe_us_v1.2.onnx"
            dataType: FP16
            input_channels: 10
            input_height: 40960
            input_width: 32
            input_nodes: "input"
            detection_output_nodes: "output"
        }
        engine_rpn {
            # This engine can be switched among FP32, FP16 and INT8
            uff_file: "/opt/plusai/var/models/obstacle_detection/pointpillars_rpn_us_v1.2.onnx"
            dataType: FP16
            input_channels: 64
            input_height: 80
            input_width: 400
            input_nodes: "input"
            detection_output_nodes: "output"
        }

# Model information
        num_class: 1
        batch_size: 1
        range {
            min_x: -14.0
            min_y: -25.6
            min_z: -2.0
            max_x: 50.0
            max_y: 25.6
            max_z: 6.0
        }


# Anchor generation
        anchor_stride: 2
        num_anchors: 40960
        num_anchor_rotation: 2
        anchor_rot_angles: 0
        anchor_rot_angles: 1.5708
        num_dir_bins: 2
        dir_offset: 0.78539

# CAR anchor info
        anchor_sizes: 4.63  # length
        anchor_sizes: 1.97  # width
        anchor_sizes: 1.74  # height
        anchor_center_heights_z: 0.47

# Pillarization
        pillar_size {
            x: 0.2
            y: 0.2
            z: 8.0
        }
        max_num_pillars: 24000
        max_num_points_per_pillar: 32
        zero_out_intensity: false
        num_gathered_point_features: 10

# Model inference
        num_pfe_output_features: 64

# Postprocess
        prob_threshold: 0.3
        nms_threshold: 0.01
        max_num_boxes_for_nms: 1024
    }
```

Losses

- Localization loss

$$\mathcal{L}_{loc} = \sum_{b \in (x,y,z,w,l,h,\theta)} \text{SmoothL1}(\Delta b)$$

$$\Delta x = \frac{x^{gt} - x^a}{d^a}, \Delta y = \frac{y^{gt} - y^a}{d^a}, \Delta z = \frac{z^{gt} - z^a}{h^a}$$
$$\Delta w = \log \frac{w^{gt}}{w^a}, \Delta l = \log \frac{l^{gt}}{l^a}, \Delta h = \log \frac{h^{gt}}{h^a}$$
$$\Delta \theta = \sin\left(\theta^{gt} - \theta^a\right),$$

- Bin Classification Loss:
- Object Classification Loss

Softmax

$$\mathcal{L}_{cls} = -\alpha_a \left(1 - p^a\right)^\gamma \log p^a,$$

Total Loss => Weighted sum

Quantitative results on Plus Dataset

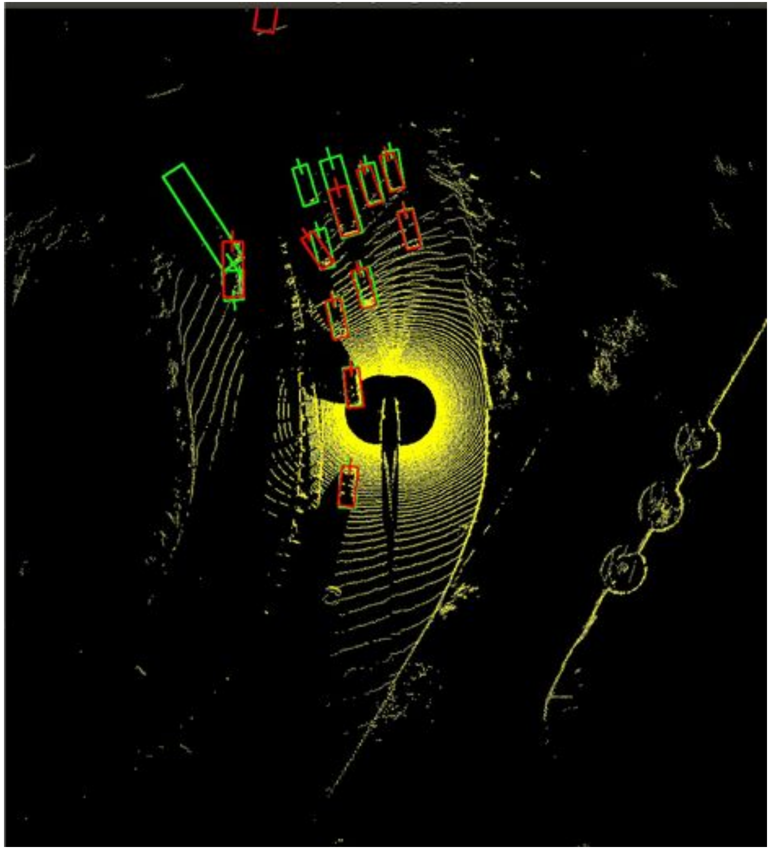
- As of 05/01/2022, our 3D dataset only has 1 Class (Car/Truck) labeled

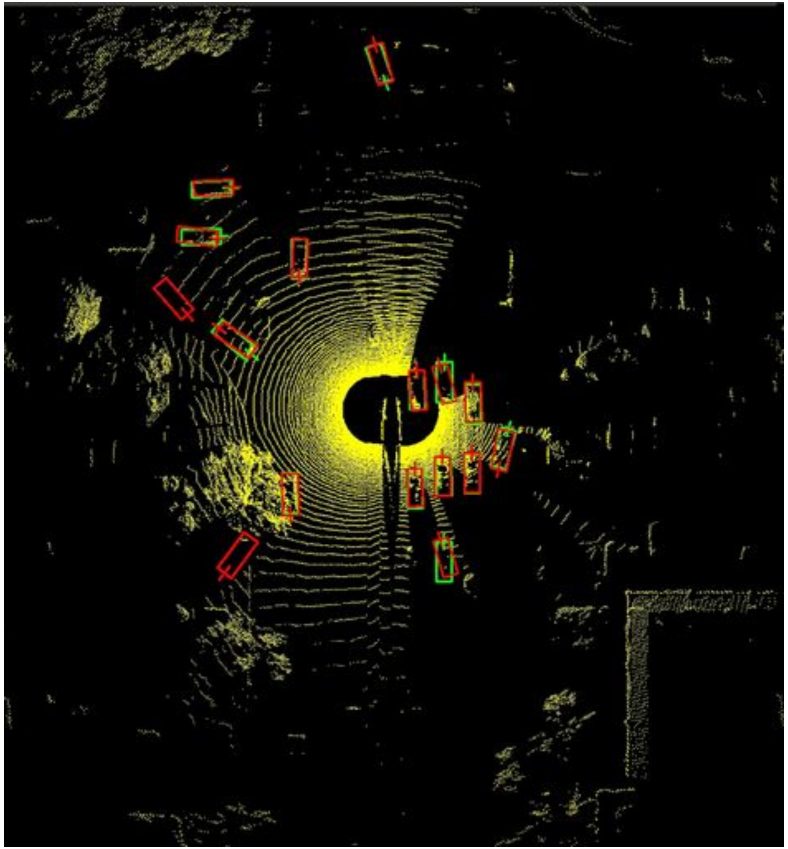
mAP	Precision	Recall
88.91	90.55	90.01

- Avg errors in prediction metrics

X_error (m)	Y_error (m)	L_error (m)	W_error (m)	Yaw_error (rad)
0.21	0.13	0.30	0.11	0.13

Qualitative Results on Plus Dataset





Tools and Repos

LidarDet repo
Repo for 3D obstacle detection training
https://github.com/PlusAI/LidarDet

Models repo
Repo containing trained models which are used at runtime
https://github.com/PlusAI/models

Labeling guideline for 3D obstacle detection
https://github.com/divimund/tools/blob/master/labeling/guidelines/3d_object_tracking_labeling_guideline.md

Labeled data available at
s3://labeling/benchmark/obstacle_tracking/data/

```
aws s3 --endpoint-url=http://172.16.0.3 ls s3://labeling/benchmark/obstacle_tracking/data/ --recursive --human-readable --summarize --page-size=100000
```

3D detection debugging

Run Pointcloud Perception Util

```
python ./perception/obstacle_detection/tools/run_pointcloud_perception.py \
-- bag {bag_path}
```

Run Perception Stack

Unified perception simulator

```
python ./perception/simulation/scripts/run_unified_simulator.py <bag_path>
```