Module 1: Object Detection -- Part A

CSC490H1 2022: Making Your Self-driving Car Perceive the World

Gongyi Shi, Anny Runxuan Dai

Part A: Learning the Fundamentals

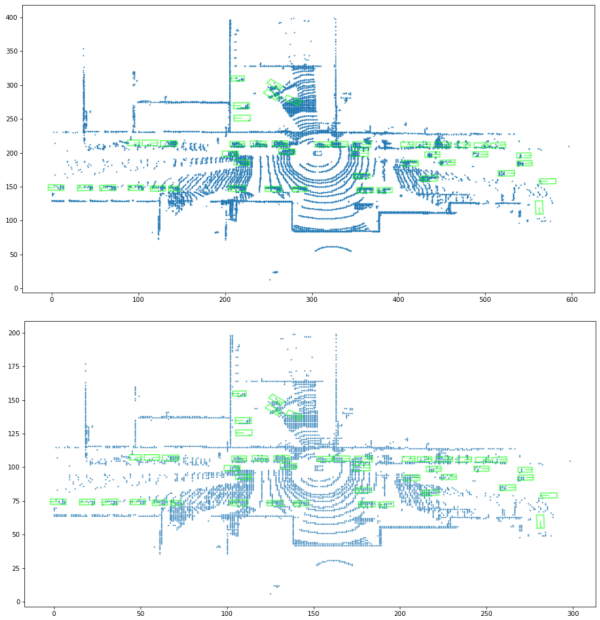
LiDAR Voxelization

Part 1

Please refer to the code in `pa_code.zip`

Part 2

Based on the output of `Voxelizer.forward`, we can see that as the step gets higher (i.e. more resolution been represented per voxel) the images appear less representative to the point cloud; the larger the step, the less voxel produced, and the less memory consumed, and vice versa. Thus, the fidelity of the voxel grid decreases as memory consumption increases.



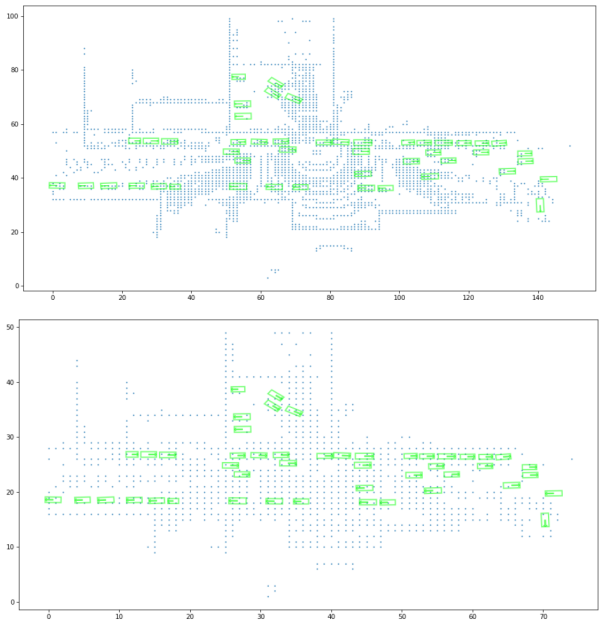


Fig. 1. Visualization of `Voxelizer.forward` on one LiDAR point cloud for resolutions step = 0.25, 0.50, 1.0, 2.

Model Training & Inference

Part 1 & 2 & 3 & 4

Please refer to the code in `pa_code.zip`

Part 5

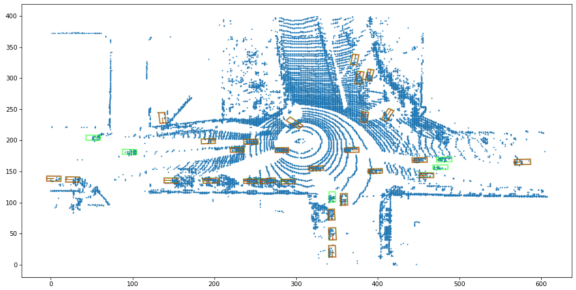


Fig. 2. Resulting detections on overfitting dataset

Part 6

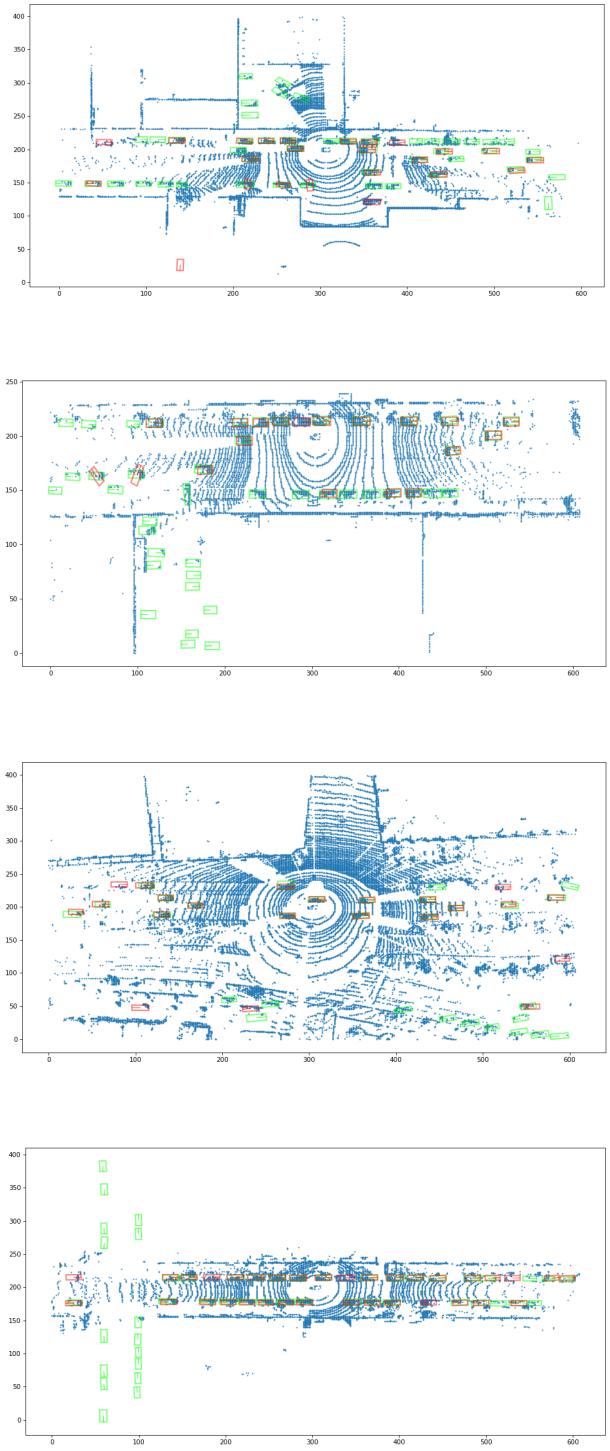


Fig. 3. Resulting detections with epoch = 10 on input frame 000, 100, 400 and 600 on the validation set

Evaluation

Part 1 & 2

Please refer to the code in `pa_code.zip`

Part 3

We notice that, as increases, AP increases as well, while the precision maintains on higher values in a wider range of larger recall values. This means the larger the τ is the more stable the detection is. However, in our scenario of auto-driving, a large τ is not realistic. A large distance from the target to its pairing prediction is logically incorrect, and dangerous.

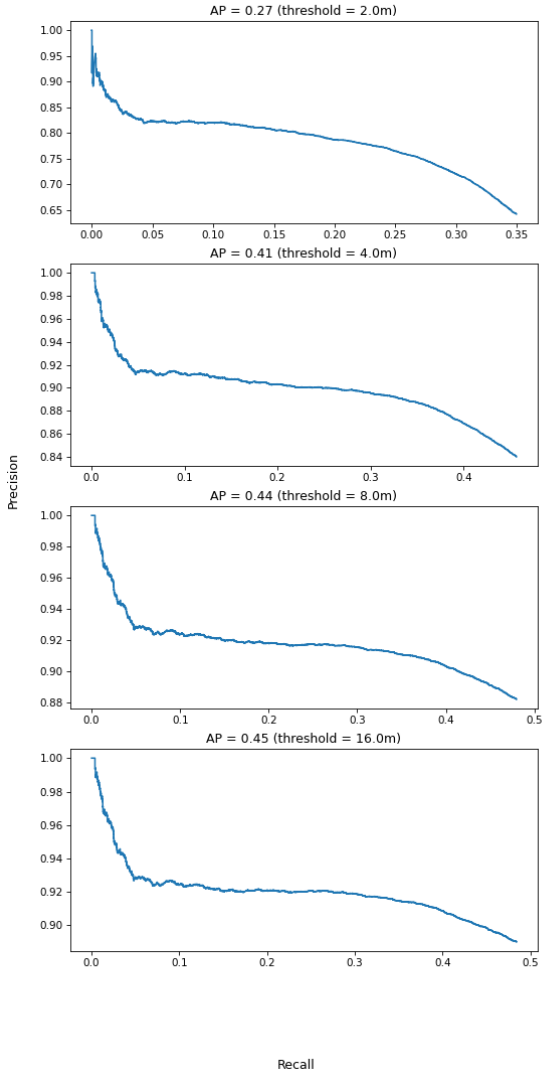


Fig. 4. PR Curve and AP with epoch = 10 on the validation set with threshold τ = 2.0, 4.0, 8.0, 16.0 m

Part 4

For each frame, we first converted the raw 3D LiDAR point cloud to $D \times H \times W$ voxels with pre-set length `Voxelizer._step` within a given range, where D, H, W are the voxel grids' size along the z-, y-, x-axis respectively.

For constructing targets, based on the centroids, headings and size of vehicles given by the data reading from `Pandaset`, construct the 7 channels' target $y^{7\times H\times W}$: detection heatmap, coordinate offsets, box sizes, and heading angles, used for prediction.

For predictions, we pass in a batch of 3D voxel, and apply a convolutional neural network to compute a feature grid $F^{C\times H\times W}$, where C is the number of features. Then apply a second convolutional neural network to predict the 7 channel prediction $x^{7\times H\times W}$.

We use the Adam optimization algorithm to minimize the square loss $||x - y||_2^2$ on training set over `num_epochs` times, and test and evaluation on the validation set. For each test frame, we plot the heatmap prediction from the 7-channel output. For evaluation, we apply different thresholds of maximum Euclidean distance between a centroid pair of prediction and target to compute the precision/recall curve and average precision to evaluate the performance of the detector.

The transformation of the raw data might result in loss of information on LiDAR point clouds, depending on the step size. For example, A dense but small area of points might indicate a small object, but converting to one single voxel might make the model misinterpret the voxel, and yield a false negative.

There are, in fact, many targets that we could not realistically expect the model to predict at inference time. As shown in the fourth plot of Figure 3, plenty of targets on the left are not even detected by the LiDAR, resulting in pointless bounding boxes. Since our input is based on LiDAR point clouds, it is unrealistic to detect targets that are far away or blocked. Thus, We should not expect our model to predict them.