GROUP 19: Report 1

Italo Nicholas Argenziano
ETH Zurich

Leonardo Pagani
ETH Zurich

Introduction

The goal of the first problem was to build a two-stage model for vehicle detection from 3D point clouds, acquired through a LiDAR. The first stage of the network is a region-proposal module that outputs bounding box proposals, which are then refined by a second stage.

1. Task 1

The implementation of the function `label2corners` closely resembles that of task 2.2 of project 1, with the only difference being that all operations are now executed using Numpy broadcasting rather than *for* loops, in order to boost the runtime efficiency. First, we obtained the coordinates of the eight corners of all bounding boxes, expressed in a reference frame whose origin is at the center of the bottom face of the bounding box and whose axes are aligned with the bounding box directions (as explained more extensively later). Then, using some basic trigonometric reasoning involving the rotation angles *ry* and the center coordinates, the corners were transformed to the rectified reference frame.
To compute the pairwise 3D IoU between each prediction and target, we used the Shapely library, which performs geometric operations quite efficiently. Since cars on the road only exhibit heading rotations (i.e., around the y-axis of the rectified reference frame), the volume of the 3D intersection can be computed as the 2D BEV intersection (defined on the xz-plane) times the intersection of the heights. Starting from the respective corner coordinates, we constructed the base polygons of prediction and target and computed their intersection using Shapely.
The intersection of heights h_{inter} was defined as follows:

$$h_{inter} = h_{hi} - h_{lo}$$

where $h_{lo} = \max\{y^{tar}_{min}, y^{pred}_{min}\}$, $h_{hi} = \min\{y^{tar}_{max}, y^{pred}_{max}\}$.

The volume of the union is just the sum of the volumes of the two boxes minus the volume of the intersection.
Finally, to compute the recall, we selected the predictions that had the maximum IoU for each target and counted how many of them had an IoU greater than the given threshold. The number of such predictions is the number of true positives. The denominator (TP+FN) is just the number of targets in the scene.
The average recall for the proposed *val-set* is 80.1%.
In the context of autonomous driving, it is of utmost importance to detect all objects on the road, without missing any. This holds true especially for other cars: a missed detection of an existing car on the road may pose a serious threat on the safety of the system, hence it is important to maintain a large number of true positives compared to the total number of instances. This is why the Recall is a fitting metric for this kind of task, in fact, a recall of 1 would indicate that all objects on the scene are detected and none is missed. On the other hand, the Precision metric focuses on how many of the predicted cars (TP+FP) are actually cars. A precision of 1 would indicate that all predictions are actually cars, but it would not account for possible undetected cars on the road. Furthermore, if considering the first stage of the model, it would not make sense to penalize false positives, since the goal of an RPN is to output raw predictions to eventually be refined by the second stage.

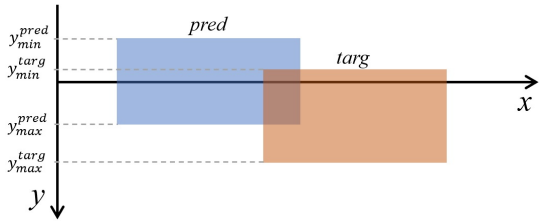

Figure 1. Intersection of heights

2. Task 2

In Task 2 the goal is to prepare the data for the second stage refinement network. In particular, we use the coarse bounding box labels from the first stage, as well as the point cloud and the extracted features, to form the region of interests for the second stage of this pipeline. First, we define a function `enlarge` which expands the output bounding box of stage 1 by the quantity $\delta = 1$ in all directions. Since we want the center of the bottom face to remain in the same position, it is sufficient to double each of the 3 dimensions of the bounding box to achieve the

newly enlarged bounding box. It is worth noting the first 3 labels of each bounding box are indeed the coordinates of the center of the bottom face. Then, we compute the ROI's by finding all points and their corresponding features that lie in each enlarged bounding box. Since the axes of the bounding boxes are not necessarily aligned with the axes of the absolute frame, checking if a 3D point belongs to a certain bounding box is not a trivial task. We decided to define a function `get_axis_aligned` which returns the 3D point cloud in the bounding box reference frame, centered in the middle of the bottom face of the bounding box. To perform this transformation we can apply the following reasoning. Let $(x_{box}, y_{box}, z_{box})$ define the coordinates of the center of the bottom face of the bounding box and let (x, y, z) be a point of the 3D point cloud, expressed in the absolute frame, then the coordinates of the same point in the canonical box frame $(x_{can}, y_{can}, z_{can})$ are computed as:

$$\begin{bmatrix} x_{can} \\ y_{can} \\ z_{can} \end{bmatrix} = \begin{bmatrix} \cos(ry) & 0 & -\sin(ry) \\ 0 & 1 & 0 \\ \sin(ry) & 0 & \cos(ry) \end{bmatrix} \begin{bmatrix} x - x_{box} \\ y - y_{box} \\ z - z_{box} \end{bmatrix} \quad (1)$$

Then, in the function `is_in_box`, it is easy to check whether a certain point belongs to the bounding box. It is worth noting that there are other possible solutions to tackle this problem, all involving non trivial geometric reasoning. Finally, by looping over the bounding boxes, we can construct the required outputs, each time by checking which points are contained in the corresponding enlarged bounding box, using the previously defined functions. If the box contains more than 512 3D points, we can just sample (without repetition) 512 of them. On the other hand, if the enlarged bounding box contains $N < 512$ points, we first make sure to sample all of the N points and then we proceed to sample $512 - N$ points among the N points contained in the box. Thus, a certain 3D points can be sampled multiple times for constructing the outputs *pooled_xyz* and *pooled_feat*. Below are three examples of pooled ROI's extracted from different scenes.

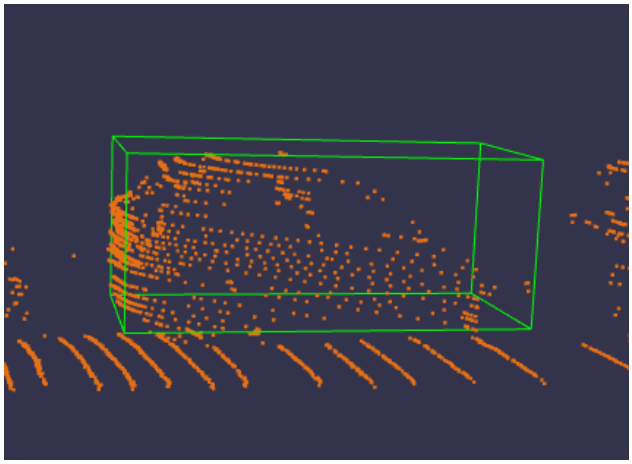

Figure 2. Pooled ROI from frame tJns0r3AFkQaF46G9tZx. Bounding box parameters: [2.6779, 1.5107, 9.2484, 1.5104, 1.5989, 3.6208, -1.5662]

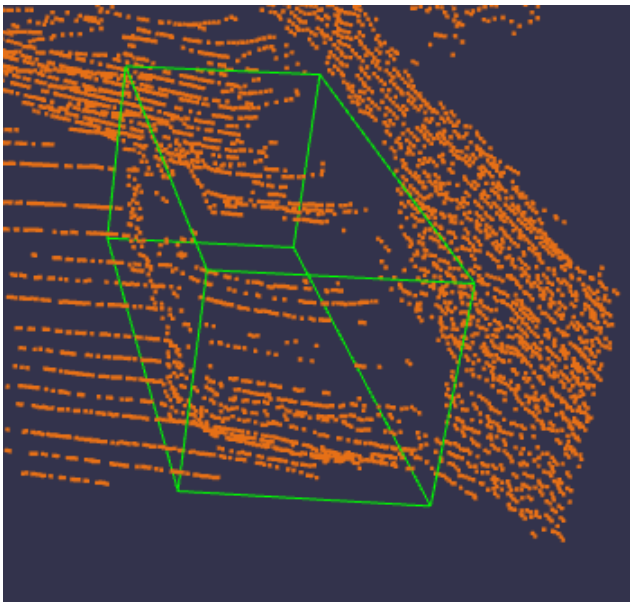

Figure 3. Pooled ROI from frame oyjL7vLBV1yE7bB3PXbm. Bounding box parameters: [-3.5126, 1.9388, 8.4936, 1.6446, 1.6658, 4.1802, 1.4937]

3. Task 3

After having generated the ROI's for the refinement stage, the next step is to build the ground truth by assigning each bounding box to a target. Thanks to the `get_iou` function from Task 1, we are able to generate an IoU matrix to store the IoU between every pair of predicted and target bounding boxes. It is then simple to assign each predicted box to a certain target according to the highest IoU. Moreover, the sampling of the bounding boxes is performed according to the criteria explained in the handout.

The main goal of this proposed sampling scheme is to balance out the number of foreground and background samples in the scene. To foster the performance of the network, it is necessary to avoid any class imbalance which would lead to the network overfitting to a specific class. If we sampled the proposals at random, more background proposals would be fed into the network, thus generating a detrimental class imbalance. In a general autonomous driving scene, foreground objects are in fact rarer than background ones.

Moreover, sampling easy background proposals (with an IoU less than 0.05 with the corresponding ground truth)

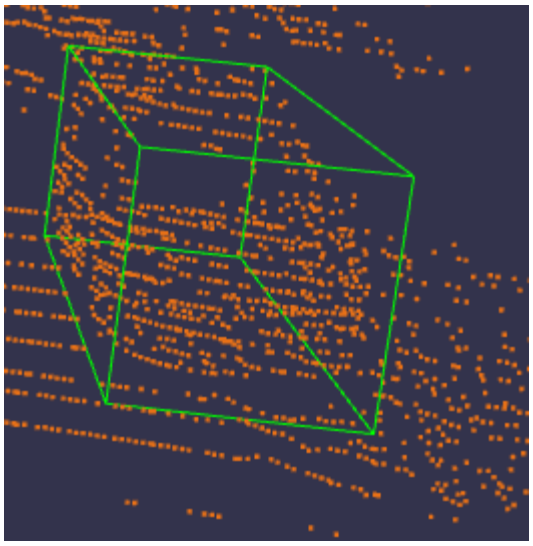

Figure 4. Pooled ROI from frame 70M703BDEV7pQ0VKEncP. Bounding box parameters: [-5.0531, 1.9290, 14.7161, 1.6735, 1.6844, 4.2581, 1.5095]

is crucial for the performance of the refinement network. It is important to feed the network with samples that are clearly non-car and therefore must not be detected. If we don't sample easy background, the network would learn how to classify as non-car a proposal with only a partial overlapping with a car, but would not know how to classify a proposal that contains a random non-car object from the scene.

The distinction between a foreground and a background proposal has to be clear and therefore there needs to be a certain margin which allows to clearly distinguish between a foreground and a background object. In this pipeline, any proposal whose highest IoU lies in the interval [0.45, 0.55] is classified neither as a foreground nor as a background. The introduction of this range guarantees that an object is categorized as such only when there is a reasonable certainty about it. If we only established a threshold of 0.5 to classify between foreground and background proposals, the distinction would be too sharp and not robust at all. The network would treat very differently proposals with an IoU of 0.49 and 0.51 whereas they may present similar features and traits. On the right is the visualization of a scene with the 64 sampled proposals.

Matching each target to the proposal with the maximum IoU is the most natural choice, as it provides with a good heuristics of the shape and location of the actual box. In this way, the training already starts from a good base and is expected to converge more quickly. It would not quite make sense to match ground truths with a proposal that has a non-maximum IoU with the ground truth itself. Finally, the highest ground truth IoU criterion is needed so that all ground truth boxes are assigned to one prediction. If this criterion was not enforced, there could be ground truth samples associated to no prediction, which clearly does not make sense.

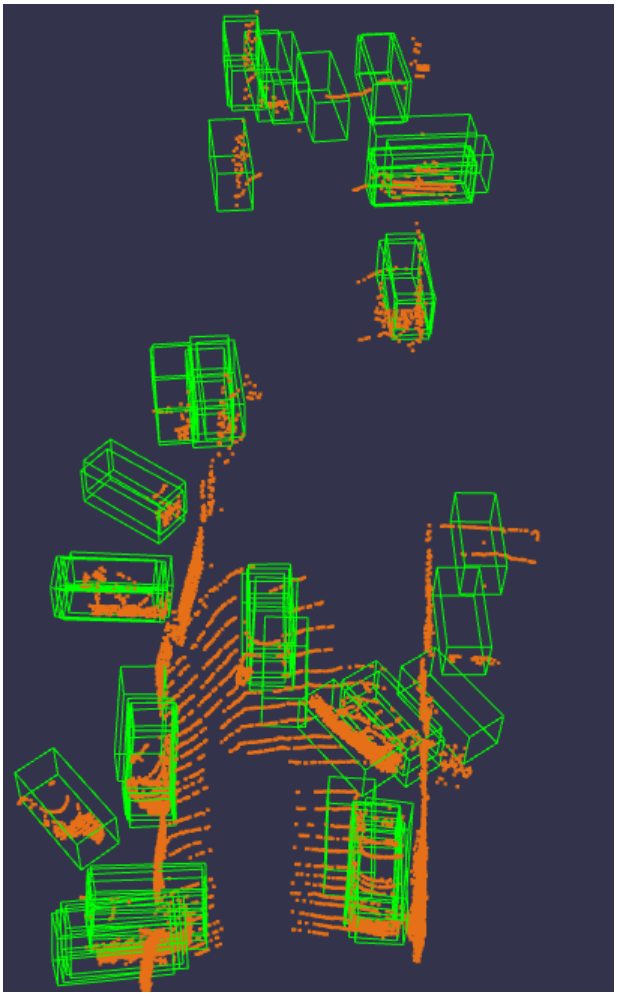

Figure 5. Visualization of frame oyjL7vLBV1yE7bB3PXbm with the 64 sampled proposals

4. Task 4

To compute the regression loss, we selected the preds and targets that have an IoU above the threshold of 0.55, since only foreground samples should contribute to the regression of bounding box coordinates. Then, we sliced them into location, size, and rotation components and computed the three loss addends.

For the classification loss, we discarded all "uncertain" predictions (with an IoU between 0.45 and 0.6). To transform the target array into a one-hot-vector, we simply rounded it to the nearest integer, so that negative targets

($IoU <= 0.45$) would be assigned a label 0 and positive ones ($IoU >= 0.6$) a label 1.

5. Task 5

Non-Maximum Suppression is used to reduce the number of predictions in a scene from the initial 64, a reasonably large amount. The criterion used for discarding predictions similar to the current maximum is the 2D IoU calculated on the BEV, as implemented in the function `get_iou_2d`, which is very similar to its 3D counterpart. One could also use the 3D IoU for this purpose, however the computation of the 2D IoU is more efficient. While it is crucial to use the accurate 3D metric to match proposals and targets or to determine whether a proposal belongs to the foreground or to the background, the 2D IoU is a sufficient similarity benchmark for this task. It is again worth noting that cars lie on the xz-plane and only exhibit heading rotations, hence the correlation between BEV and 3D IoU is sufficiently strong.

6. Task 6

Below is displayed the progression of the training loss per epoch and the mAP scores for the three difficulty levels.

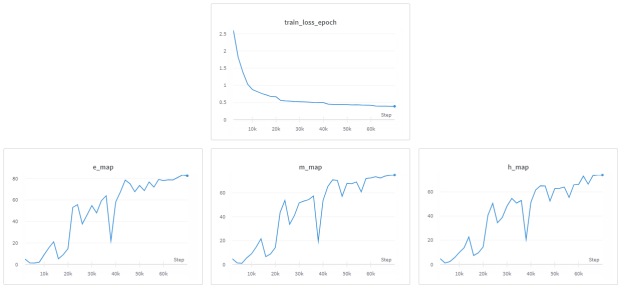

Figure 6. Plots of Training loss, Easy, Moderate, and Hard mAP

Although the training loss decreases quite steadily over the epochs, the mAP scores evolve in a quite unstable fashion. The final mAP scores of the baseline model in the validation and in the test steps are:

	Easy	Moderate	Hard
Validation	82.59	**75.09**	74.07
Test	80.88	**73.48**	72.35

On the right are some example visualizations at different stages of training.

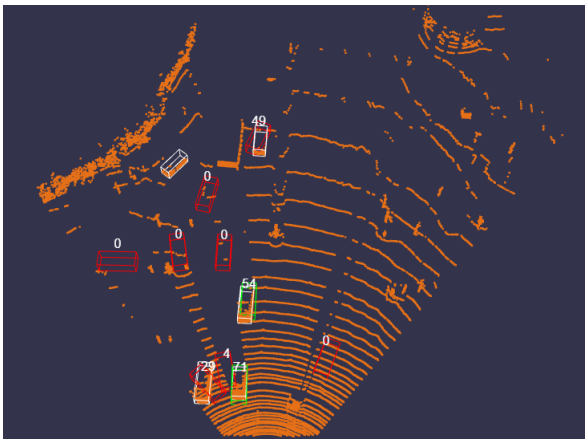

Figure 7. Visualization at epoch 0 (beginning of training)

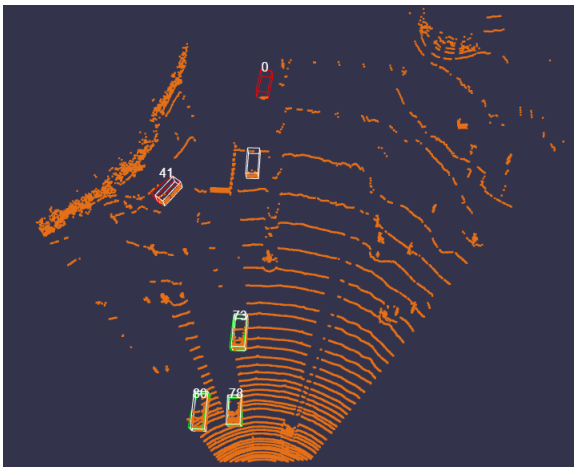

Figure 8. Visualization at epoch 16

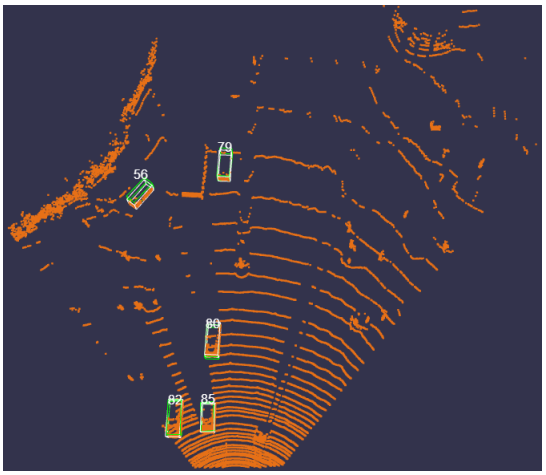

Figure 9. Visualization at epoch 34 (final predictions)