Project 3: 3D Object Detection from Lidar Point Clouds

Deep Learning for Autonomous Driving
Team ID 18
Ivan Alberico, Nicola Loi
May 17, 2021 - July 2, 2021

Problem 1 - Building a 2 Stage 3D Object Detector

In this project a 2-stage 3D object detector is implemented and improved with the aim of detecting vehicles in autonomous driving scenes. In the first stage of the network, a Region Proposal Network computes coarse bounding boxes around each vehicle in the scene starting from point cloud data. Each bounding box is fully described by 7 parameters, which express its position, size, and orientation in the space. The second stage, instead, is aimed at refining all the previously computed proposals of the first stage, before generating the final predictions.

1

To compute the 3D IoU between the predicted and target bounding boxes, an *R-tree* graph of the bounding boxes is created in order to speed up the whole process. In this way, the IoU is computed only on those bounding boxes that are close enough to be likely to intersect each other, therefore a lot of computations are spared. Of these bounding boxes, the 2D intersections of the projection on the x-z plane are computed with the *Shapely* library. Then, the 2D intersections are easily exploited to calculate the 3D IoUs.

The average recall for the demo is 0.801 (80.1%), while the recall of the whole *val*-set is 0.813 (81.3%). The recall, calculated as True Positives/(True Positives + False Negatives), is a good metric to assess the quality of the first stage proposals since it represents how many positives (cars) are detected out of all the ground truth positives (all the cars) present. It is more important to have a low number of False Negatives (cars not detected), than having a low number of False Positives (non-car objects detected as cars). The latter are taken into account by the precision, computed as True Positives/(True Positives + False Positives). In our case the precision is not a good metric, since the main priority of the first stage is to find and propose the highest possible number of ground truth cars in the scene, without worrying about including non-car objects, since then there is the second stage network to refine and correctly classify the proposals from the first stage. A recall of 1, the maximum, means that every ground truth car in the scene is proposed by the first stage, (the best result achievable by the first stage), while a precision of 1 means that every proposal of the first stage is a car, but it does not give information on how many ground truth cars were left out by the first stage. This is a problem for an autonomous driving task, since we don't want to exclude any real cars from the second stage, for safety reason.

Since the precision is not a valuable metric for our first stage network, consequently also the average precision is not a good assessment, since it is not only dependent on the recall but also on the precision.

2

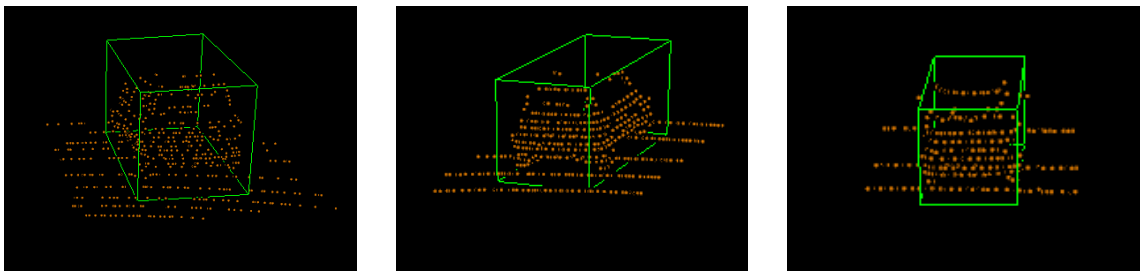
To pool adequate ROI's, each predicted bounding box from stage 1 is first enlarged, and if it is present at least 1 point within the enlarged box, a point sampling is performed, to have exactly 512 points for each prediction. If no points are present, the prediction box is discarded.

The predicted bounding box are enlarged by 1 meter in every direction: $h_{large} = h + 2m$, $w_{large} = w + 2m$, $l_{large} = l + 2m$; to check which points of the point cloud lie in it, the x-z coordinates of the whole point cloud are transformed so to be referred with respect to to a frame where the x-z coordinates of the origin are the centre of the bounding box, and whose axis are aligned with the edge of the box. Instead of transforming only the points that are approximately very close to the box, the whole point cloud is transformed. It was tested that performing a rough first filtering (using an R-tree graph or numpy), exploiting the vertices coordinates of the box, and then transform the fewer number of points, was slower than directly transform all the points. The points are first translated with a translation vector corresponding to the negative x-z coordinates of the centre of the bounding box (no translation in y direction), then they are rotated around the y-axis with a 2D rotation matrix employing the negative rotation angle of the bounding box:

$$\begin{pmatrix} \overline{x}_i \\ \overline{z}_i \end{pmatrix} = \begin{pmatrix} \cos(\text{-}\theta) & \sin(\text{-}\theta) \\ -\sin(\text{-}\theta) & \cos(\text{-}\theta) \end{pmatrix} \begin{pmatrix} x_i - x_{box} \\ z_i - z_{box} \end{pmatrix} \tag{1}$$

while $\overline{y}_i = y_i$. With these new points coordinates, it is now very simple to check if a point is within the enlarged bounding box, remembering to account for the non-translated y coordinate. These transformed coordinates are only exploited to check which point to pool, but at the end the original coordinates are pooled. If the enlarged bounding box contains at least one point, a point sampling is performed to have exactly 512 points.

Figure 1 present some examples of the generated ROI's in three different scenes, displaying the points within the enlarged bounding boxes but showing the wire-frames of the original non-enlarged bounding boxes. Figure 1a shows the bounding box with parameters (3.5176, 1.6643, 11.0931, 1.5604, 1.6181, 3.7449, -1.5691) from scene 0, Figure 1b the box with parameters (1.0739, 1.6274, 20.2643, 1.6843, 1.6877, 4.1997, -1.1749) from scene 1, and Figure 1c the box with parameters (1.1498, 1.7912, 24.3467, 1.562, 1.633, 3.7346, -1.4909) from scene 2.



(a) scene 0, bounding box parameters (3.5176, 1.6643, 11.0931, 1.5604, 1.6181, 3.7449, -1.5691)

(b) scene 1, bounding box parameters (1.0739, 1.6274, 20.2643, 1.6843, 1.6877, 4.1997, -1.1749)

(c) scene 2, bounding box parameters (1.1498, 1.7912, 24.3467, 1.562, 1.633, 3.7346, -1.4909)

Figure 1: Examples of ROI's pooling.

3

Once ROI's have been generated, next step in the pipeline is to assign each predicted bounding box to a target, and to sample them following a specific scheme, in order to supervise the training of the network. The way prediction-target pairs are generated follows a predefined procedure that assigns a ground truth to each proposal according to the highest IoU observed. Moreover, the number of candidate proposals is generally higher than the actual number of objects in a scene, hence sampling strategies are employed in order to reduce their number. A visualization of the sampling is shown in Figure 2 and Figure 3.

The sampling scheme is necessary in order to have a balanced proportion between the foreground and the background samples present in the scene. If proposals were randomly sampled in each scene, this would lead to an unbalanced set of bounding boxes on which the network will be trained later on. Generally, random sampling would generate more background proposals than foreground proposals, since in autonomous driving scenes foreground objects are more likely to be less in number with respect to the background ones. By following the sampling scheme presented in the project description, sample proposals will be drawn proportionally to the type of objects encountered in the specific scene and, in this way, over-fitting to specific categories is more likely to be avoided.

As far as background sampling is concerned, a distinction between easy and hard proposal sampling is crucial. Easy background samples, i.e. proposals with an IoU less than 0.05 with respect to the corresponding ground truth, play a significant role in 3D object detection of autonomous driving scenes. Easy background samples are in fact negative samples vital to train the network to classify correctly non-car objects. Giving equal priority to hard and easy background samples allows more variability in the prediction stage, feeding to the network negatives samples that are clearly non-cars (IoU¡0.05, easy background) and negative samples that are non-cars, but whose distinction is more difficult since their proposal box is in fact overlapping for a non-trivial amount with the ground truth box of a car (IoU¡0.45, hard background). 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.

In addition to that, the distinction between foreground and background samples is not sharp, but in general it is preferred to consider an intermediate gap between the two categories, which includes all those objects and parts of the scene that are not easy to classify and that would bring ambiguities in the training. If, for example, a sample is considered to be foreground with having an IoU above 0.5 and background vice-versa, there would be a sharp distinction between foreground and background proposals. In this way, a proposal with IoU of 0.49 and one with an IoU of 0.51 would be treated in a significantly different manner, even though they might present similar traits or shapes, and this would lead to an increase of misclassifications and to a worsening of the model's performance. Instead, introducing a range (in this case, bounding boxes having IoU between 0.45 and 0.55) in which proposals that are not clearly categorizable as foreground or background are not taken into account, makes sure that an object is categorized as such only when we are certain about it and a "safe" margin is overcome.

The reason why each ground truth bounding box is matched to the proposal with the highest IoU, is that in this way the prediction that best fits the shape of the target is selected, in order to facilitate the whole training process. In doing so, a more stable initial point from which the network starts its training is found and also it takes less time to achieve valuable results in terms of computations. It does not make sense to exploit a worse proposal to train the network, if a better one is present. For example, if a proposal with a lower IoU is matched to the ground truth, it can happen that the predictions do not converge to the wanted result if the proposal box is in a pose where its enlarged version is not including a large amount of car points in its pooling. The

prediction with the highest IoU will have within its pooling points the higher number of actual car points, or anyway the higher percentage of car points with respect to its pooled points.

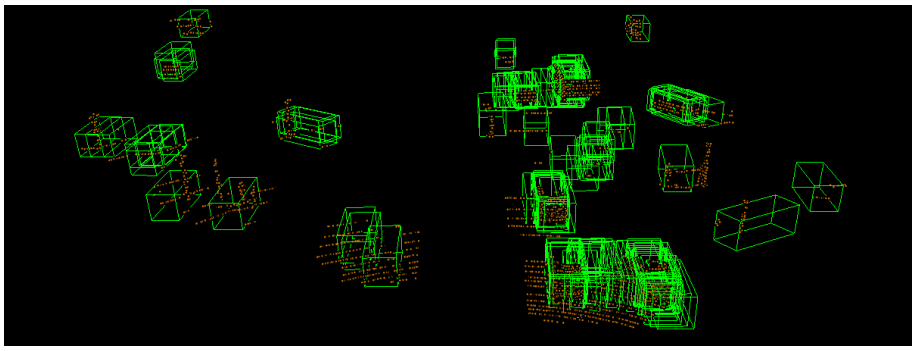


Figure 2: Visualization of scene 0 with all the proposals from *task2*

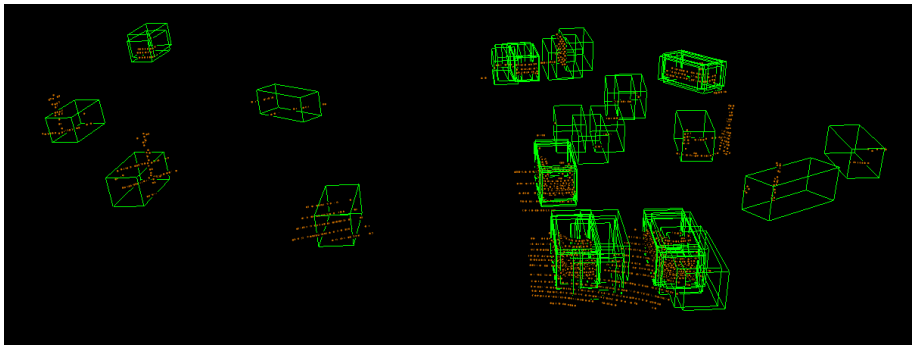


Figure 3: Visualization of scene 0 after that the 64 proposals have been sampled in *task3*

4

The aim of the fourth task is to compute the regression and classification losses for the training of the model. First of all, predictions coming from earlier stages need to be filtered and the way this is done changes according to the type of loss.

a) For the regression loss, only positive predictions are considered, i.e. predictions whose 3D-IoU is higher than the 0.55 threshold. From these positive predictions, 3 different L1-Smooth losses are computed: $L_{location}$, L_{size}, and $L_{rotation}$, which are then summed together, but with the L_{size} loss scaled by a factor of 3. In the case of no positive predictions encountered, the regression loss is automatically set to 0.

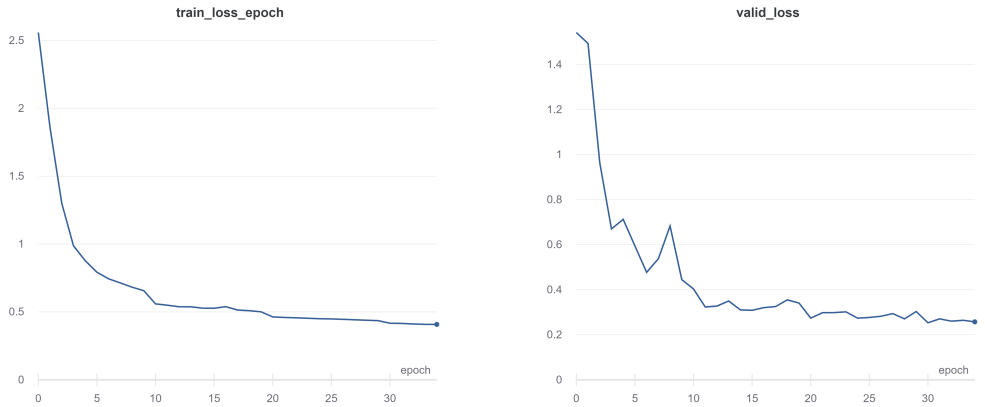
b) For the classification loss, instead, both positive and negative proposals are considered, which correspond to class 1 and 0 respectively. A prediction is considered positive if its 3D-IoU is higher than 0.6, negative if it is lower than 0.45. The classification loss is computed from a weighted sum of Binary Cross Entropy losses computed separately on the positive and negative proposals. The weights are the proportion of positive or negative proposals with respect to the total number of proposals. If no positive or no negative proposals are present, the corresponding loss is set to 0.

5

Non-Maximum Suppression (NMS) is implemented in order to remove the redundant proposals from the previous stages, employing the IoU among the different predictions and discarding the lower scored predictions that have an IoU greater than 0.1. Computing the 2D IoU is not a strict requirement for the NMS algorithm, in fact also the 3D IoU could be used. The reason why the 2D BEV IoU is preferred is that it is indeed more computational efficient, and its use does not neglect important spatial information. Even though, in general, the 3D IoU provides a more trustworthy result while comparing bounding boxes, it is not essential when performing NMS in 3D object detection of cars, where 2D IoU is sufficient for its purpose. In fact, since cars are objects that lay on the same plane in most of the cases, it is not possible to have two different cars one on top of the other. Therefore in the filtering pipeline it is enough to only check for the intersections on the x-z plane projections, without worrying about the vertical dimension.

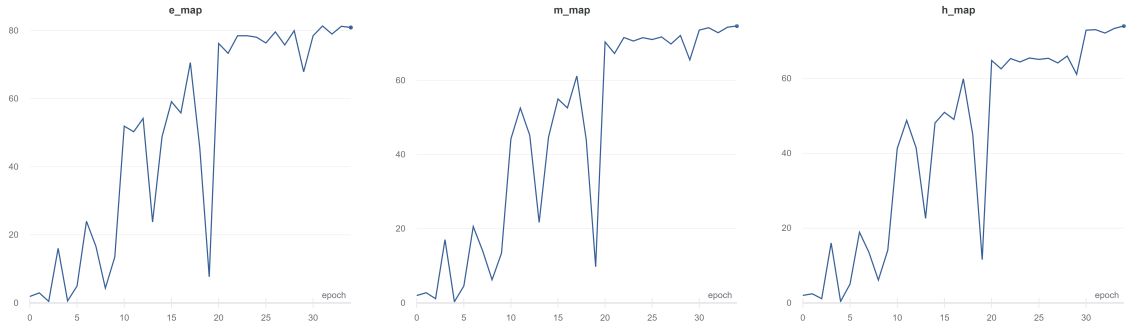
6

The network is trained for 35 epoch with a training set of 8000 scenes and a batch size of 4, for a total of 2000 steps for each epoch. The validation set is composed of 1712 scenes, while the test set of 3769 scenes. The training and validation losses graph are shown in Figure 4, where it can be seen that both losses have a rapid drop in the first part of the training, and then they steadily decrease for the rest of the training. The evolution during the training of the mAP scores of the three difficulty levels (easy, medium, and hard) are displayed in Figure 5; at the end of the training, the mAP scores achieved by the network on the validation set are `easy:` 80.93, `medium:` 74.68, `hard:` 74.04, while the ones achieved on the test set are `easy:` 79.167, `medium:` 72.43, `hard:` 71.34. For a better visualization of the results, in Figure 6 is presented the evolution of the car detection of the network in the scene 0 of the validation set; the predictions were evaluated after training the network for 1 epoch, 18 epochs, and 35 epochs (end of the training).



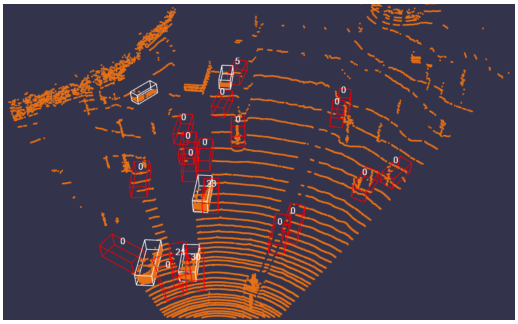
(a) (b)

Figure 4: Evolution of the losses of the network (a) training loss, (b) validation loss.

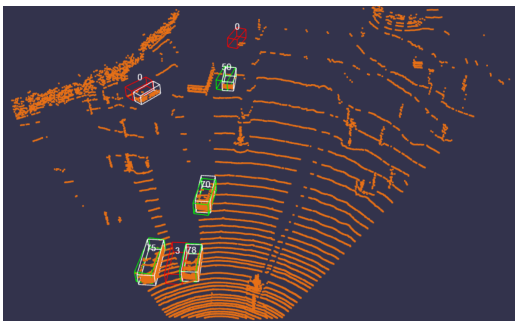


(a) (b) (c)

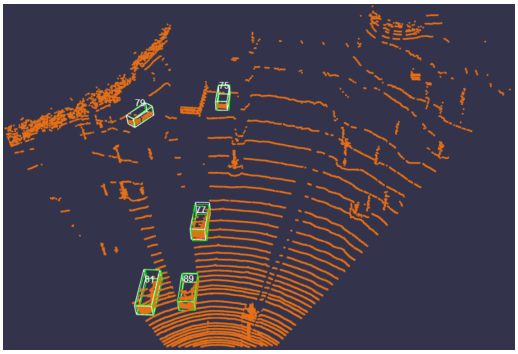
Figure 5: Evolution of the mAP scores (a) easy, (b) medium, (c) hard.



(a) car detection after training for 1 epoch



(b) car detection after training for 18 epochs



(c) car detection after training for 35 epochs (end of the training)

Figure 6: Evolution of the car detection in scene 0, after (a) 1 epoch, (b) 18 epochs, (c) 35 epochs (end of the training).