# GROUP 15: Report Project 2 Problem 1

Jérôme Bonvin 20-917-332
Kai Zhang 23-947-583
Liam Achenbach 20-940-268

## Abstract

*This project began with the implementation of code designed to familiarize us with the dataset and the visualization of point clouds generated by LiDAR sensors.*

## 1. Introduction

To better interpret the outputs of our deep learning model, we require a robust method for visualizing predictions in 3D or their projections onto 2D images. Three tasks were identified: 1) projecting point cloud data onto optical images, 2) projecting bounding boxes of vehicles onto optical images, and 3) visualizing the entire scene in 3D using *vispy*.

## 2. Related Work

Camera calibration plays a crucial role in computer vision algorithms. Several methods [3, 5] have been developed for precise calibration. Although we could explore various techniques, the intrinsic, rotation, and translation matrices provided make the Direct Linear Transform (DLT) [3] a suitable choice for our implementation.

## 3. Method

Using the provided 3D LiDAR points, the rotation matrix (around the Y-axis), and the intrinsic camera matrix, we employed the DLT [3] algorithm to project the 3D points and bounding box corners onto a 2D image:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} R & T \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\implies \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Here, $X$, $Y$, and $Z$ are the 3D points observed by the LiDAR; $R$ and $T$ are the rotation and translation matrices, respectively, for converting to rectified camera coordinates; $K$ is the camera intrinsic matrix; $u$ and $v$ are the projected image coordinates; and $\lambda$ is a normalizing constant. This method is applicable for tasks 1 and 2 since projecting bounding boxes involves projecting their corners and connecting them with lines.

## 4. Results

Using the method described in Section 3, we successfully reprojected 3D point cloud data onto 2D images. Figure 1 shows the projected points, with distinct colors representing different object classes. A minor discrepancy is visible in the right part of the figure, where blue points (building) overlap with orange points (minivan). This misalignment arises from the positional differences between the optical camera (Cam 2) and the LiDAR (Velodyne laserscanner), as shown in Figure 2.

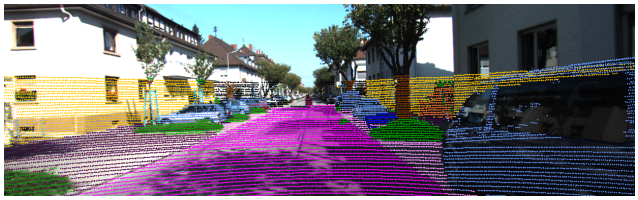


Figure 1. Projection of points from *data.p* onto the optical image

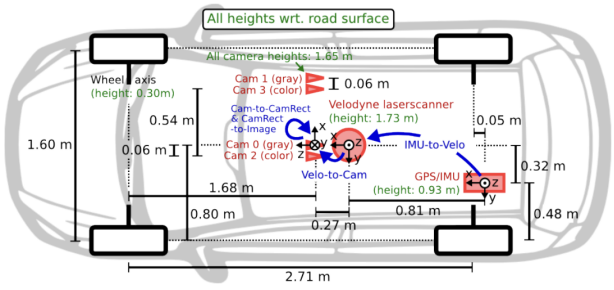



Figure 2. Placement of sensors on the car

Bounding boxes were also projected successfully, as shown in Figure 3.

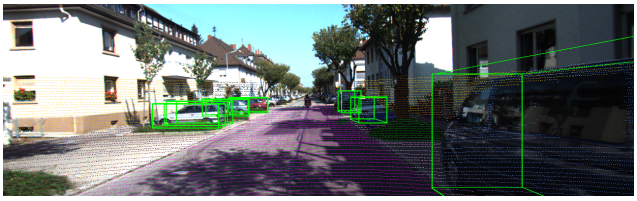


Figure 3. Projection of points and bounding boxes from *data.p* onto the optical image

For better spatial understanding, we visualized the scene in 3D. Figure 4 provides an overview of the detected bounding boxes and LiDAR-detected points.

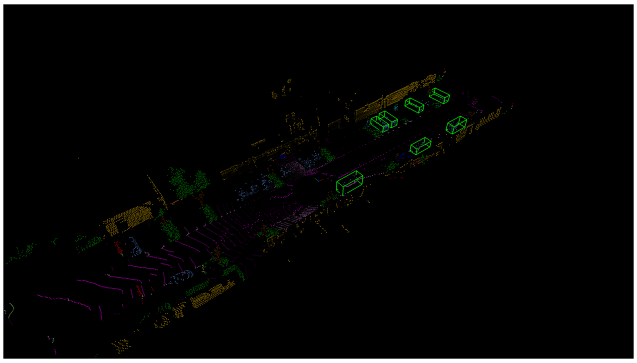


Figure 4. Projection of points and bounding boxes from *data.p* in a 3D scene

To evaluate whether bounding boxes accurately identified all vehicles within the camera's field of view (FOV), we constrained the LiDAR FOV for correspondence. This limitation is illustrated in Figure 5.

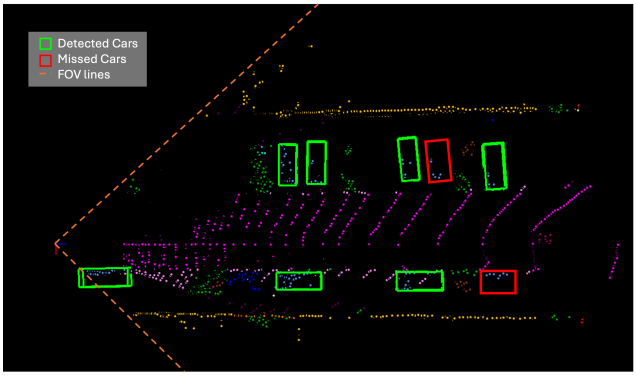



Figure 5. Projection of points in the limited field of view from a bird's eye view

From the segmented point cloud, we observed that seven out of nine cars in the FOV were correctly detected, with the missed vehicles highlighted in red. Therefore our network failed to identify two cars.

## 5. Conclusion

In conclusion, we successfully implemented a visualization pipeline that provides multiple perspectives for evaluating the outputs of our deep learning models. These visualization methods will prove invaluable for diagnosing and improving model performance in future tasks.

## Abstract

*In this problem, we implement some functions that will help train the second stage of the RPN such as, the calculation of Intersection over Union (IoU) of 3D boxes, get the pooled cloud points around a prediction, filter out the predictions and calculate the different losses.*


## 1. Introduction

In problem 2, we were given 5 tasks to implement, to successfully train the second stage of the RPN pipeline. The first task was to calculate the IoU of 2 bounding boxes as this would provide us with a metric to assess our model. Note that this cannot be used as a loss since it is not differentiable. Secondly, we have to implement a code that will pool the point cloud points in an enlarged bounding box of the predictions, this will help the model concentrate just on these points to determine if it is a car or not. Thirdly, when the model trains, we assign the predictions to their targets so that the model knows which prediction corresponds to which target so that we can successfully calculate the loss. Afterwards, we calculate both losses: the regression loss using the smooth L1 loss and the classification loss using the Binary Cross-Entropy (BCE) loss. Finally, we filter out some prediction boxes even more using the Non-Maximum Suppression (NMS).

## 2. Method and Results

In this section we will go over the methods used to implement the five tasks explained in the introduction.

### 2.1. Task 1: IoU Calculation

The first metric that we want to calculate is the Intersection over Union (IoU). As its name indicates, we calculate it by dividing the intersection of 2 volumes by its union. Given that the union of two shapes is the sum of their volumes minus their intersection, we can write this union term to avoid extra calculation.

$$IoU = \frac{I_{ij}}{U_{ij}} = \frac{I_{ij}}{V_i + V_j - I_{ij}}$$

$V_i$ is the volume of a rectangular prism, this can be quickly calculated as $l \cdot w \cdot h$ since these are given for every box. $I_{ij}$ denotes the intersection between two prisms, which is a bit more complicated to compute, that is why Figure 6 represents a visual example to help follow the up coming explanation. Since this metric will often be used, it needs to be executed as fast as possible while still being accurate. To calculate the intersection, we can utilize the restraint that two boxes can only intersect if the maximums of all three $(x, y, z)$ coordinates of one box are larger than the minimum coordinates of the other. This enables us to filter out two boxes with 0 intersection early in the stage. Furthermore, we can utilize the information that a box can only rotate along the $y$-axis, which means that we can decouple the volume with the overlap in the $y$-axis and the intersected polygon in the $xz$ 2D space. Computing intersection in the 2D space is easier and makes the process much faster. $U_{ij}$ stands for the union of the two prisms and equals the sum of volumes minus the intersection as stated before.

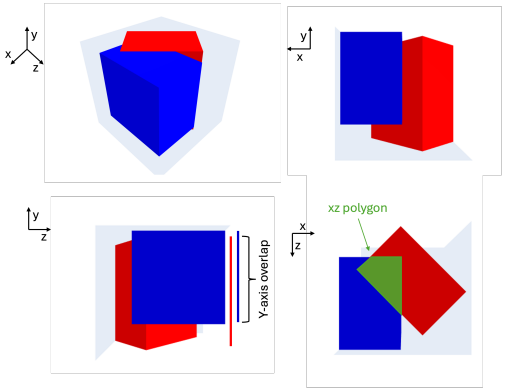



Figure 6. Representation of bounding boxes and their intersection

The average recall for the *val*-set is 0.8134, which is already quite a high recall.

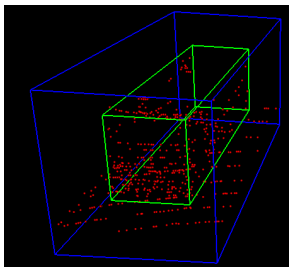
Recall fits well in our object detector task since it only focuses on the parts with positive ground truth, where it actually has an object. All the boxes the model doesn't predict are likely containing no objects within them. As a result, there are abundant True Negative (TN) in this case and it's not relevant. Our goal is to retrieve as many objects as we can, but the model can achieve high precision even if all the predictions correspond to the same ground truth box. This contains only one meaningful prediction after the non-maximum suppression, which is not informative. We want our model to have diverse predictions instead and recall is a better metric to ensure all ground truth objects are detected.

### 2.2. Task 2: Region of Interest Pooling

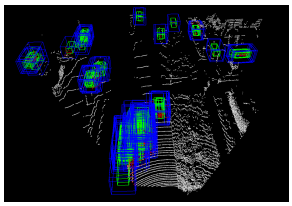
For this part of the tasks, we want to construct features for each predicted box so that they can be fed into the second stage refinement network. For each bounding box, we enlarge the box in all directions since nearby points can also be helpful for further refinement. We then filter to get those points within the enlarged box and randomly select 512 points to make all the features have the same shape. An illustration of the pooling process can be seen in Fig. 7.

### 2.3. Task 3: Input-Target Pairs Sampling

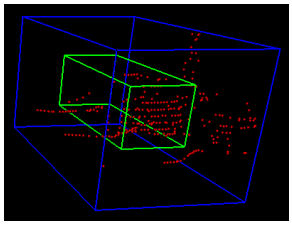
In this task, a set of valid proposals is given and we need to assign each proposal with its corresponding target to fa-



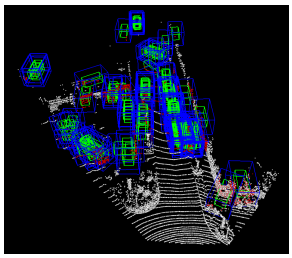
(a) Template bounding box within frame 22 with parameters $(-4.0319, 1.5823, 17.1729, 2.4839, 2.0806, 6.4482, 1.5621)$



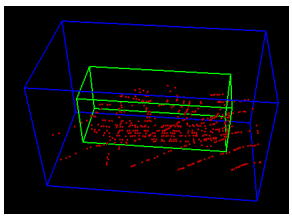
(b) Overall view of frame 22



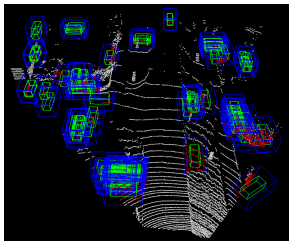
(c) Template bounding box within frame 41 with parameters $(-0.545, 1.6981, 26.8901, 1.5044, 1.6018, 3.6602, -1.7086)$



(d) Overall view of frame 41



(e) Template bounding box within frame 89 with parameters $(-8.9506, 2.0118, 15.4647, 1.6592, 1.716, 4.3955, 0.0887)$



(f) Overall view of frame 89

Figure 7. **Illustration of extracted bounding boxes and their corresponding points**. Given that `project2_val.h5` is too large to fit on our local machine, the frame indices mentioned here corresponds to the indices in `project2_minival.h5`. Frame 22, 41 and 89 are selected to demonstrate the pooling process. The right column shows the overall view of all the points in the frames and the pooled boxes and their points, and the left column represents a single valid predicted bounding box and its assigned points in the right scene. The green bounding boxes are valid proposals from first stage network. The blue bounding boxes are enlarged boxes for pooling features. The red dots are the points pooled out of the original scene and the white dots are the rest points ignored.

cilitate the training process. The target is simply the ground truth box with the highest IoU with the input. The input-target pairs are then sampled with the rules explained below, and an example of the final proposals can be seen in Fig. 8.

#### 2.3.1 Reason for sampling scheme

The input proposals are regarded as foreground proposals if their IoU with ground truth boxes are above a certain

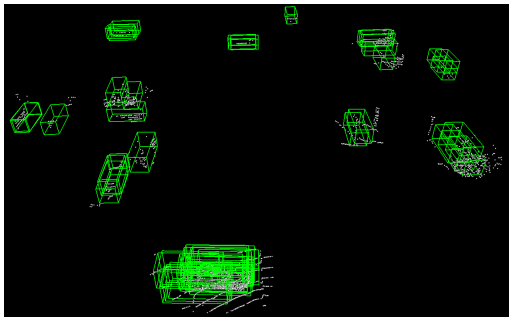


Figure 8. **Sampled Proposals**. The 64 sampled proposals for Frame 89 of `project2_minival.h5` (enabling train flag).

threshold and as background proposals if their IoU are below another threshold. When both foreground and background proposals are given and foreground proposals are enough, our sampling scheme ensures that the sampled proposals are balanced between the two categories. If there are too few foreground proposals, all those proposals are chosen, and background proposals are sampled to reach the fixed number of samples.

This sampling scheme is required to build a balanced training dataset. Given that there are only a few objects in the scene, we expect the Region Proposal Network (RPN) to have more background proposals than foreground proposals. Our goal is to accurately refine on positive proposals, thus we need to sample a balanced training set to not overfit to the more abundant class (background). This scheme also reduces the number of less informative negative proposals in most cases, which speeds up the training process.

#### 2.3.2 Comparison to random sampling

As stated before, there are usually more background proposals than foreground proposals. Randomly sampling from the full distribution without respecting the imbalanced class distribution would lead the model focusing more on background information. Even though the model might have good performance on making accurate background prediction, it won't help in our case where our goal focuses on predicting accurate bounding box for foreground proposals. Moreover, the proposals for some target ground truth are completely missed during the sampling process, making the model has longer convergence. This network would not be able to perform well with this random sampling.

#### 2.3.3 Without easy background proposals

The sampled background proposals are equally distributed with easy and hard background proposals. Easy background classes should enable the model to learn clearly non-object regions. In contrast, hard background proposals contain parts of cars, which have some similar features as those in foreground proposals. Excluding easy background pro-

posals could lead to learning an incomplete or faulty background representation. The model might start misclassifying obvious background areas as objects because it hasn't learned to recognize and ignore them effectively. This could increase the share of False Positives in the predictions. Also, this might reduce the ability to generalize to unseen data where a large share of easy background samples is proposed wrongly by RPN and the model is never trained on how to deal with them.

#### 2.3.4 Adapting FG-BG threshold

Adapting the threshold to 0.5 such that the "ignore region" is 0 instead of 0.1 as in the task, will introduce noise into the data. Since samples close to the threshold of 0.5 are ambiguous on whether they are foreground or background proposals, the difference among features of those proposals is unclear. A sharp, binary label provides inconsistent signals during training, which could make it hard for the model to distinguish between true positives and false positives and ultimately degrades performance. Instead, using an ignore region like we do in the training process of the model gives clear boundary between labels and cleaner training data, and thus more stable training as well as higher performance.

#### 2.3.5 IoU-GT proposal matching

Some ground truth objects might be very tricky and none of the proposals exceed the foreground IoU threshold, for example, objects that are far away with a few points. Without this step, those target objects are not assigned to any proposals, and the model will fail to learn about how to detect those objects. Moreover, some ground truth targets will shadow other targets where all the proposals have higher IoU with them than others. This step ensures that those shadowed targets are also assigned with some proposals. This process will probably also prevent bias towards "easier to predict" samples and thus increase performance at inference time. Including these samples therefore seems to be a good way of improving performance and reducing bias.

### 2.4. Task 4: Losses

To train the model, we need to define loss so that the model can be updated with back propagation. Our loss is composed of two parts, regression loss and classification loss. The regression loss is the smoothL1 loss to ensure that the bounding boxes of positive samples are close to ground truth, while the classification loss is the binary cross entropy loss to ensure that the confidence predictions of our model match ground truth labels.

### 2.5. Task 5: Non-maximum Suppression (NMS)

Now that we have an array of predictions of each scene, many predictions correspond to the same ground truth objects. To reduce the number of redundant predictions in the final stage, we apply a technique called Non-maximum Suppression (NMS). This is an iterative selection mechanism. On each step, the proposal with largest confidence is selected, and the rest predictions are considered as overlapping or not based on their IoU with the proposal. Those predictions with IoU higher than a threshold are discarded before visual evaluation.

The IoU can be calculated both in 3D and in 2D Birds-Eye-View (BEV) projection. Advantages of the 2D BEV include fitting more to the purpose of autonomous driving, being more computational efficient without loss of generality, and having better well-defined IoU in space. In the use case of autonomous driving, we focus more on the 2D positions of objects for navigation and obstacle avoiding. The height of an object is not so relevant as the car should move around the object anyway, and we want to avoid two predictions appearing at the same position, which make 2D BEV map fit more to our purpose. Also, since we consider cars roughly resting on the ground plane and only rotate around the y-axis, we can project to 2D without loosing constraints. Calculating areas of polygon directly in 2D space is much less complex than computing overlapping volume in 3D area. Even though we choose to reuse 3D IoU calculation function for code simplicity, computing IoU in BEV leads to higher computational efficiency in general. Finally, the space becomes sparser as the dimension grows higher. It's much easier to result in lower IoU in 3D space than 2D space, and the choose for threshold becomes more tricky.

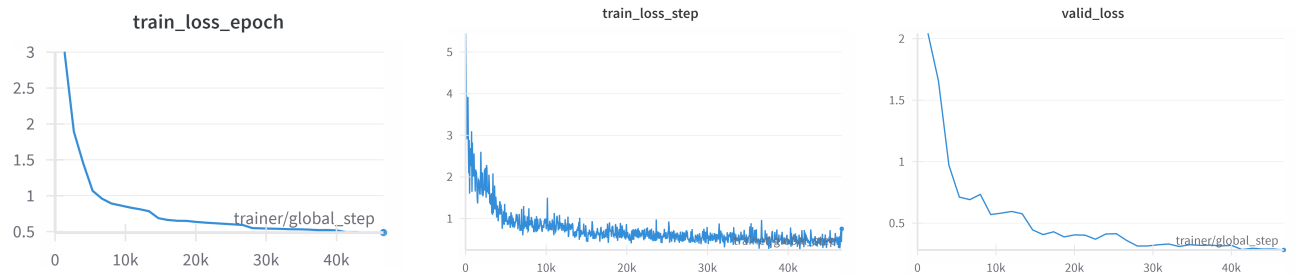
Using the 3D overlap on the other hand could improve precision because it potentially captures the 3D shape better and proposes harder constraint on suppression threshold. It would also be able to deal better with larger vertical variations, like the case where there are cars on both roads and bridges. Since LiDAR sensors only have a limited vertical range, and we focus on object detection for navigation, these benefits are negligible. We choose 2D overlap to avoid two cars detected at the same position in the output.

### 2.6. Task 6: Training

Given that we have implemented every component needed, we can finally train our model and inspect its performance. The training statistics can be seen in Figure 9 and 10. Our model has a steady decrease in losses and reaches quite satisfying precision at the end of training. A visualization of the process can be seen in Figure 11. Even though the prediction is not accurate at the beginning, it matches nicely with ground truth boxes after training. This creates a baseline solution for object detection from point cloud.

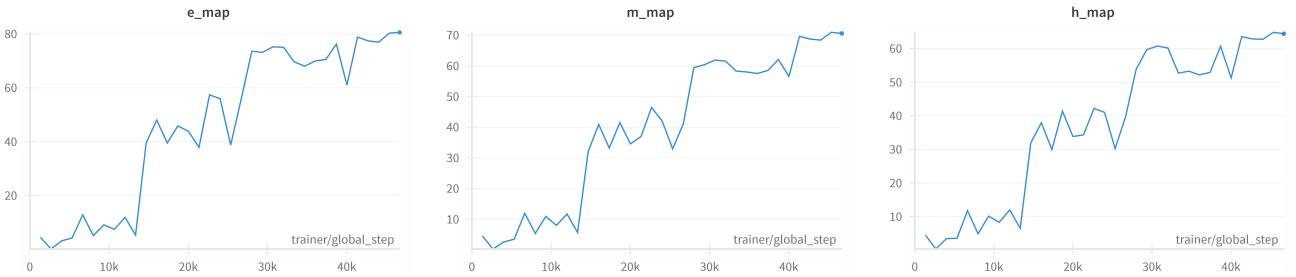
## 3. Discussion

The whole training process takes around 13 hours to finish, which is quite an efficient implementation. It achieves



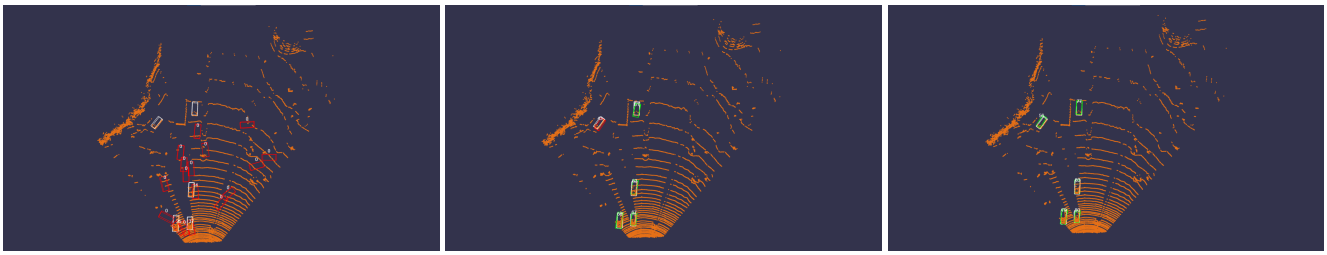

(a) train loss per epoch (b) train loss per step (c) validation loss per epoch

Figure 9. **Curves of losses** during training.




(a) precision under easy metric (b) precision under medium metric (c) precision under hard metric

Figure 10. **Curves of precisions** of prediction during training.



(a) Prediction result after 1 epoch (b) Prediction result after 18 epoch (c) Prediction result after 35 epoch

Figure 11. **Comparison of prediction** from the beginning, mid-way and end of the training cycle

85.21, 75.51 and 73.69 for easy, medium and hard %mAP respectively.

## Abstract

*In the paper, we will extend the baseline model implemented in Problem 2 and try to achieve higher performance. To achieve this goal, we first inspect our current prediction to discover the weakness of our baseline model. Based on this insight, we propose two improvements that lead to better mAP score on test set, namely, data augmentation on rotation label for rotation robustness and self-attention layer for better prediction confidence.*


## 1. Introduction

The task in Problem 3 is to work on the second stage of a two-stage 3D object detection network. The first stage of the network functions as a Region-Proposoal-Network (RPN) which proposes coarse bounding boxes. These boxes are then refined to bounding-box predictions in the second stage. The pre-implemented network for the second stage is a PointNet++ [1] style network. It feeds the bounding boxes through multiple layers of set abstractions. Afterwards, the output is split into a regression and a classification head.

## 2. Method & Results

Initially, we train the baseline network implemented in problem 2, whose scores can be seen in Table 1. Investigating the performance of the PointNet++ [1] reveals that two main issues are inhibiting performance on the task. Firstly, the network under-performs on the rotational prediction of bounding boxes and secondly, it occasionally predicts more bounding boxes than ground-truth vehicles are in the scene. Both of these issues can be observed in Figure 12. We employ various methods to improve the network performance on these two problems.

### 2.1. Rotational accuracy

The rotational accuracy has an important effect on the performance metrics, since an incorrect rotational angle reduces the overlap of two bounding boxes significantly. We follow an initial suspicion that the classes in the *train*- and *val*-dataset are too heavily biased towards certain rotations, which makes it harder for the model to generalize to rotations well. We investigate the target rotational distribution in a smaller *val*-dataset and plot it against baseline model predictions in Figure 13. As seen in the figure, the dataset includes a few rotations at which the density of samples is especially high. This is to be expected in an autonomous driving context because the majority of cars occupy similar spaces such as driving on the street or being parked at the side. However, this could diminish model generalization performance and potentially lead to a worse performance on the test set. Inspired by the performance increase achieved

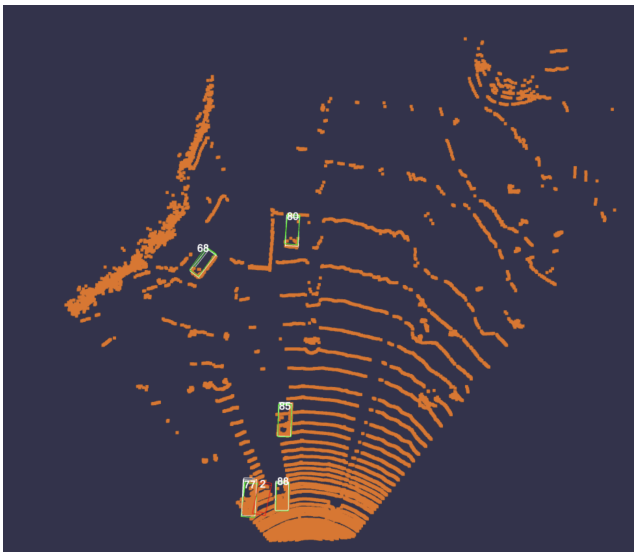



Figure 12. **Performance of the Baseline Network**. For all the bounding boxes appear in the frame, the center and size of the bounding boxes seem to fit nicely to target output, whereas the inconsistency of rotation angles significantly degenerates the IoU scores. More over, the model predicts a false negative in the bottom left of the image, which is failed to get filtered out by NMS.

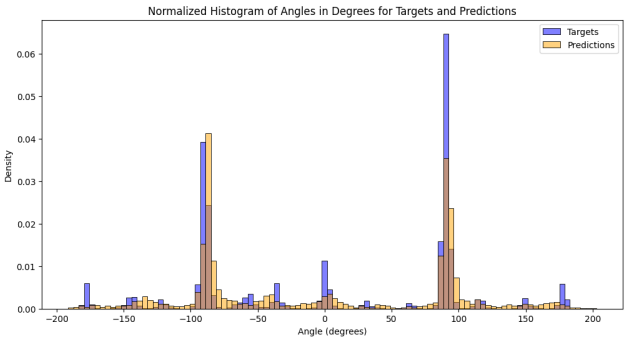



Figure 13. **Rotation Angles Contrast**. The x-axis reflects the rotation in degree while the y-axis presents the fraction of samples among total data. The target distribution in validation set is shown in blue and the predicted distribution is shown in orange. The significant disparity near $90^\circ$ demonstrate the inconsistency of the angle distribution between the training and validation dataset, which in turn worsen the total performance of the model.

on the regular PointNet++ [1] in the PointNext paper [2], we investigate potential benefits of applying data augmentation methods during the training process.

Because we only want to improve on the second part of our two stage network we can only perform data augmentation in the second stage of the 3D object detection network. Therefore, we add 5% random noise to the rotation label after the extraction of ROIs. This increases task complexity and enforces the model to more robustly reduce the ROI proposal to the correct bounding box. We compare model with this training noise to the baseline model in Table 1.

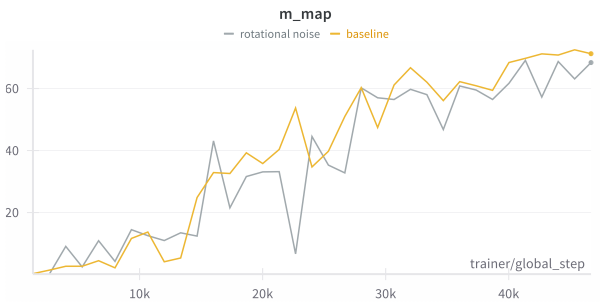



Figure 14. **Effect of augmentation on mAP curve**. The orange line is our baseline model and the gray line reflects the model with rotational noise. The noise, however, makes the model less stable.

#### 2.1.1 Rotational noise injection effects

As we can observe in the training progress plot in Figure 14, the updates to mAP estimates are less stable in the noise-injected training run. However, we cannot observe clear benefits in terms of performance after reaching epoch 35, which is as expected since we inject noise into each proposal of the RPN, and the noise on test makes it unstable.

#### 2.1.2 No-noise "fine-tuning" training effects

Injecting the noise during test-time is not necessary as we expect that the proposals of the RPN have comparatively high quality as we investigated in Problem 2. Adding random disturbance directly doesn't actually help. Therefore, training for a few more epochs without the data augmentation could in a sense "fine-tune" the model to the task at hand. We investigate this by training for a few more epochs without data augmentation from a model checkpoint that was trained with data augmentation. Unfortunately, we don't observe any additional benefits in terms of mAP. This might be caused by the difference in Adam learning rate after a new learning session. We believe this insight still being valuable and would like to investigate more if time permits.

### 2.2. Prediction confidence

The second issue we investigate is that the baseline model is occasionally predicting bounding boxes for cars not existing in the scene as seen in Fig. 12 and 15. Additionally, it performs worse on cars further away due to poor model confidence in those predictions as seen in Fig. 16. We therefore conclude that a further refinement on the features after the set abstraction layers could improve performance.

We choose Multi-head Self Attention for refining the extracted feature sets. The attention layer works as a neck layer between the PointNet++ encoder and decoder heads as shown in Fig. 18 Our choice is grounded in the fact that self attention can increase context for individual points of

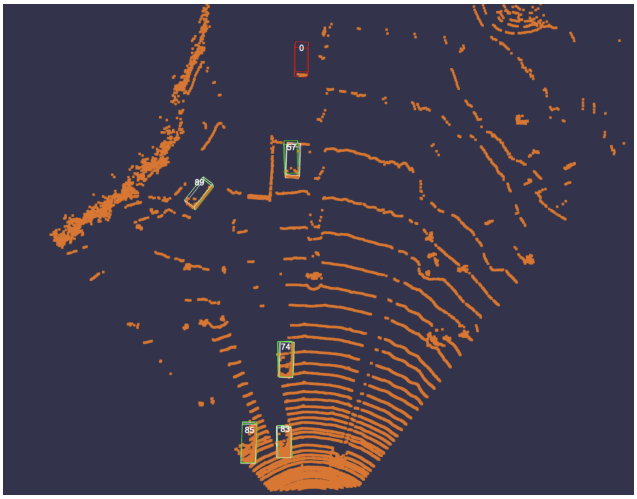



Figure 15. **Performance of the Baseline + Data Augmentation Network**. The prediction at the top has no paired ground truth.

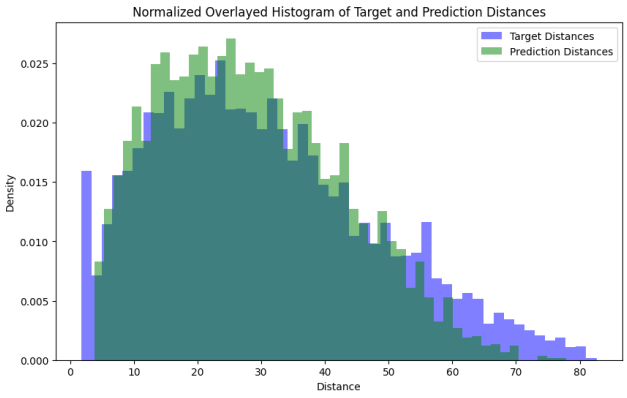



Figure 16. **Distribution of Distances of the Center of Bounding-Boxes to the Origin (0,0,0) and Their Predicted Distances**. The predicted distances match well with target distances between 5m and 50m. But the distances are predicted poorly and have large gaps to real distances after 50m.

the point cloud which we expect to decrease sensor noise and improve classification quality [4]. We think this can especially support our performance on far away cars as they have fewer data points within the ROI and thus are harder to classify with confidence by our model as shown in Figure 17. Additionally, we expect the attention mechanism to reduce inaccurate predictions of the model closer to the center of the scene since the increase in feature quality should decrease wrong predictions and sensor noise.

We also include the rotational noise from subsection 2.1 into one run to see the effect of it. We then also continue that run from a checkpoint to see the performance difference when training with rotational noise and then "fine-tuning" for a few more epochs on the clean data. The results can be observed in Figure 19. Unfortunately, we mixed two settings up during training and were not able to get clean results for a run with only self-attention. We are now running

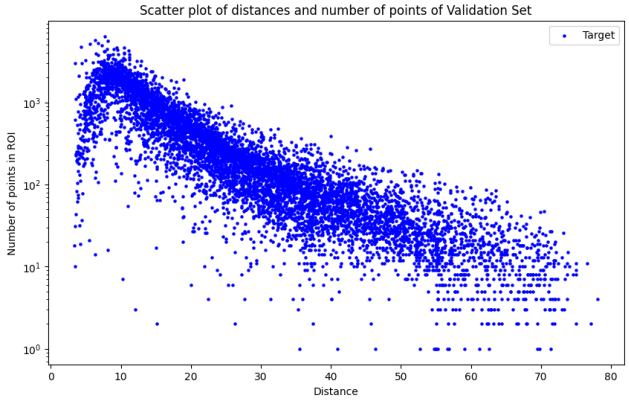



Figure 17. **Correlation of the Distance of a Bounding-Box with the Number of Points in its ROI**. Bounding boxes have most abundant features around 10m, and there are only a handle of meaningful features between 50m and 80m region.

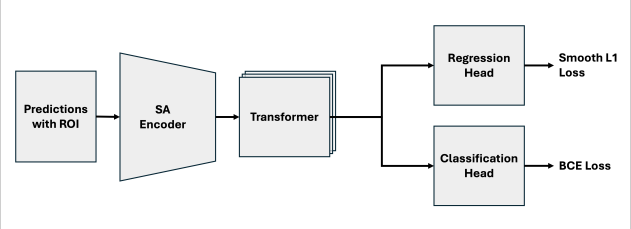



Figure 18. **Attention Based Network Architecture**.

a separate test for it but the run time exceeded the project deadline by roughly 50 percent. We can therefore only comment on the projected performance of this run.

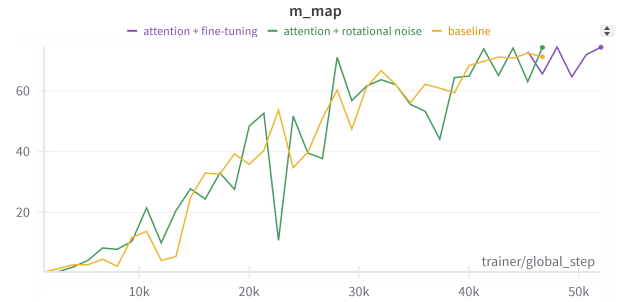



Figure 19. **Effect of both attention and augmentation on mAP curve**. The yellow line is our baseline model, the green line reflects the baseline model combined with self-attention and augmentation, and the purple line is for further fine-tuning without augmentation. The self-attention mechanism this time results in slightly better performance as shown in Tab. 1.

### 2.2.1 Effects of self-attention

Even though we are unable to show the effects of the full self-attention run without the injected noise, we can argue

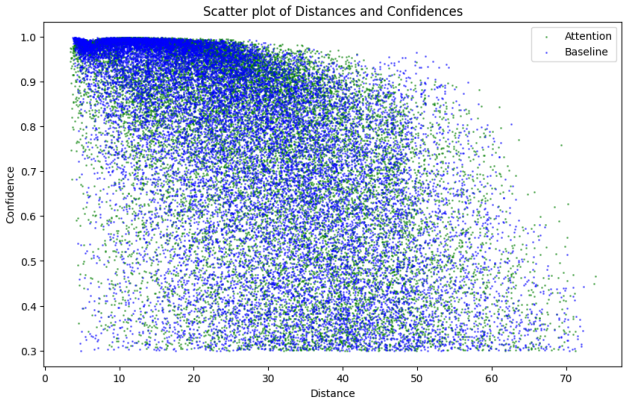



Figure 20. **Scatter plot of distance vs. confidence**. The x-axis reflects the distance of predicted bounding box from origin, and the y-axis means the confidence the model regard the prediction is correct. The blue points are predictions from baseline model while the green points reflects prediction by our newly proposed self-attention based model. For points with same x-axis, green points have higher y-axis in general, showcasing that the attention model is generally more confident on distant points.

that self attention decreases the amount of incorrect bounding box predictions compared to the baseline by using the run which includes both self-attention and noise injection as explained in 2.2.2. We can see that in Figure 20 and also in the logged sample scene predictions that we now neither observe any incorrect close nor incorrect far predictions. In Fig. 20 one can see that the confidence especially for farther predictions increases for various data points. This means that self-attention can be beneficial in refining the features of ROIs far away from the sensor.

### 2.2.2 Self-attention and data augmentation

We also have one run where we employ noise injection during the training process and use self-attention for refining features. This run is the one where we achieve the highest performance as seen in Extended Figure 2. Due to the ablated failed run we are unable to pinpoint with certainty the effect of the self-attention on this. Interestingly enough, we perform better on the easy and hard benchmarks than moderate benchmark when comparing this run to others on the test set on CodaLab.

## 2.3. Results

The detailed metrics for each of the adjustments mentioned above are listed in Tab. 1. It seems that data augmentation on rotation label of ROI proposals worsen the final improvements. However, we believe that the difference of the distributions in rotation angles between the training and testing datasets is one of the key factors that the performance of the model is not satisfying. And we believe further investigations on this disparity can finally turn into

an improvement on its performance. Despite this, our self-attention plus augmentation model still achieves a significant improvement compared to the baseline, demonstrating the superior effect of this extra self-attention layer. Details about losses and precisions over epochs can be found on Extended Figure 1 and 2.

| %mAP | Easy | Moderate | Hard |
|---|---|---|---|
| Baseline | 79.38374 | 71.22706 | 64.7868 |
| Baseline + augmentation | 79.4246 | 68.37156 | 62.76995 |
| Baseline + augmentation without noise | 75.57293 | 67.951 | 62.5289 |
| Baseline + augmentation + self attention | 83.81816 | 74.35977 | **73.12829** |
| Baseline + augmentation without noise + self attention | **85.33845** | **74.44335** | 67.47684 |

Table 1. **Performance of different models.** Baseline stands for the simple implementation shown in Problem 2. Augmentation denotes adding random noise to the rotation label as discussed in Sec. 2.1.1. "without noise" refers to "fine-tuning" mechanism in Sec. 2.1.2 and self-attention structure is shown in Sec. 2.2.2.

### 2.4. Additional Attempts

Despite data augmentation on rotation labels and extra self-attention neck structure, we also make other attempts that turn out having no improvements. First of all, we notice that there are more proposals closer to the origin while there are sparser proposals in the farther areas by RPN. Thus we try to tackle with this problem by evenly sampling each target prediction like what we did for sampling background. Such that those farther, harder proposals are requested to get trained more often. This actually worsens the performance of the model. One possible reason is that the proposals in those further regions actually have much smaller IOU with ground truth targets, making the training information inconsistent and providing bad hints for inference.

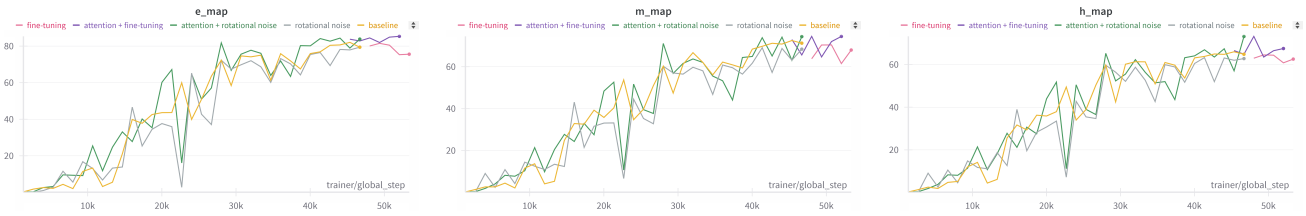
Another attempt we tried is that we noticed that the IOU used to calculate classification loss is actually the one between the RPN proposals and target ground truth. This score does not reflect the model's certainty about its particular prediction. Thus, we calculate the IOU between predicted bounding box and ground truth bounding box firstly and use this IOU as the information fed into classification loss. This, however, having 0 mAP score after a few epochs. This might because that the predicted bounding box have very little IOU with ground truth until 10 or 20 epochs, making the classification head not starting training after many epochs. This could also be the reason why the overall performance of the adjusted model behaves worse than our baseline.

## 3. Conclusion

We conclude that the feature refining using self-attention on the ROIs improves model performance. Additionally, injecting rotational noise seems to improve performance on rotational accuracy. Unfortunately, due to a mix-up we were unable to ablate the self-attention influence. Further investigation would therefore explore the self-attention in its isolated performance and additionally tuning the parameters such as number of heads and layers.


(a) train loss per epoch (b) train loss per step (c) validation loss per epoch

Extended Figure 1. **Curves of losses** during training.




(a) precision under easy metric (b) precision under medium metric (c) precision under hard metric

Extended Figure 2. **Curves of precisions** of prediction during training.

## References


[1] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space, 2017. 7

[2] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Abed Al Kader Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies, 2022. 7

[3] R. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal on Robotics and Automation*, 3(4):323–344, 1987. 1

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. 8

[5] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000. 1