GROUP 30: Report 2

Alan Savio Paul
ETH Zürich

Loïc Houmard
ETH Zürich

Abstract

In this task, we discuss the limitations of the 2-stage 3D object detector model trained in Problem 1. We discuss problems and propose improvements for the Proposal Sampling, the features used in the Stage-2 network, and the regression loss used for regressing the boundary box parameters. We succeed in running experiments for 2 variants of proposal sampling, one is a simple tiled padding based sampling and the other ensures good representation of the targets in the sampling. We achieve significant improvements in the target-based sampling, compared to our final model in Problem 1 and the tile padding based sampling.

1. Introduction

In this project, the main task we are trying to solve is to detect 3D objects from unordered and irregular 3D point cloud data. We develop a 2-stage 3D object detector to detect vehicles in autonomous driving scenes. The detections are described using a 7-dimensional vector containing parameters for the bounding boxes enclosing each object. The seven degrees of freedom are the center (x,y,z), the size (h,w,l), and the yaw rotation ry in radians.

In Problem 1, we implemented the missing parts of the second stage refinement network and trained the model by following the instructions in the handout. In this report, we describe our solution for Problem 2 where we improve upon our performance in Problem 1.

We focus on improving three main areas of the architecture in order to achieve better performance.

1. Proposal Sampling

2. Features used in training the second stage network, and

3. Regression loss used for boundary box parameter regression

We successfully make 2 incremental improvements in the performance by implementing two variants of sampling the proposals and ROI. We also implement methods to solve problems with 2. and 3. but do not succeed in getting the experiments to work as expected.

1. Proposal Sampling We notice that there exist problems with the sampling scheme described in the handout. Although the sampling follows a good strategy to mix foreground and background samples, it ignores the fact that some targets may be under-represented in the selected samples for a given scene. For example, if there are 32 background samples to be sampled from a set of 50 background samples, the sampling scheme ignores the potential target imbalance. There could exist many more proposals belonging to one target than the rest of the targets and the sampling ends up selecting more of the majority. In our work, we ensure that all targets are as well represented in the final selection as possible. This successfully improves the performance, as can be seen in the Results section (Sec 4).

2. Features used in training the second stage network Inspired by PointRCNN [2], we notice that the local spatial information when coupled with the already existing global information could help the network perform better. The authors of the paper perform ablation studies and report that this greatly increased performance. We adapt their method but do not succeed in getting a working experiment due to technical difficulties.

3. Regression loss used for boundary box parameter regression Again inspired by PointRCNN, we use their bin-based regression loss for the refinement network. The authors used this loss for both the RPN and the refinement network, but we decided to use it only for the refinement as we did not have sufficient resources to train the RPN ourselves. As with the previous experiment, we stumbled upon issues with the adaptation of this to work with our model and were not left with enough time. Based on the PointRCNN paper, we expect this feature to have improved our model's performance to some extent.

In the following Related Work section, we briefly look through some of the literature that is relevant to our work, with a focus on PointRCNN. We then look in further detail into the methods we adopt. We conclude with a Results section to demonstrate our model performance, followed by a brief discussion.

2. Related Work

Most research in this line of work can be classified into two categories: 3D object detection from *2D images* and

from *point clouds*. Our project falls under the latter category, and in this section we discuss the closest related work, PointRCNN. This paper, similar to our project, employs a bottom-up 3D proposal generation method which directly generates robust 3D proposals from point clouds. Earlier methods projected the point cloud to bird's eye view and utilized 2D techniques to learn features and generate 3D boxes [1]. However, there is clearly information lost in such techniques. There also exist other methods that learn features of voxels instead of 3d points [3].

PointRCNN interestingly combines the global semantic features from the RPN with transformed local spatial points for the refinement network. The transformation here is from the global 3D space to the proposal bounding box coordinate system where the origin is at the center of the the proposal box. The authors observe that the local features lose depth information and so they also added the distance from the sensor. They concatenated their new local features and passed them through several fully-connected layers to encode them, after which they concatenated these encoded local features with the global features to get their final features. In our work, we re-implemented this but we use just 2 hidden-layers to keep our training time as low as possible.

Another interesting feature of their model is the bin-based regression loss. This is a loss where the y value and size values (h, w, l) are regressed using Smooth L1 loss like we already did in Problem 1. For the X and Z axes localization they split the surrounding area of each point in the proposal box into uniform bins in 2D and then applied cross-entropy loss to find the centre of the proposal box. Similarly for the rotation angle, they use bin classification. The authors used this loss for both the RPN and the refinement network, but for feasibility reasons we only attempted to implement this for the refinement network.

3. Method

We implemented 3 different methods to improve the performance of the second stage refinement network. We succeeded in one method but encountered technical difficulties in the other two. In this section, we describe the methods that we succeeded and failed in implementing.

3.1. Sampling

To improve on our initial performance, we implemented two methods of sampling.

3.1.1 Tiled padding

As discussed in our report for Problem 1, while preparing the input of 64 samples using the sampling scheme described in the handout, in cases where we had fewer samples than required, we chose to generate a fully randomly sampled set of required size by taking random samples from the small sample set. An alternative would have been to use the original small sample set and pad it with random samples to obtain the required size. The reason for our choice was that we felt the random padding in the handout could be interpreted either way. However, we think on disadvantage of our choice is there is a chance that some rarely-occurring proposals could be left out. As a first attempt to improve our performance, we implemented a variant of the alternative. We used numpy's tile method to pad the array instead and this gave us a small improvement. This sampling is applied in task2 (ROI pooling) as well as in task3 (Proposal Sampling). This approach made sense because it avoids the problem of some proposals not being selected (or selected very few or too many times) due to randomness. In our implementation, we can switch to this mode of sampling by choosing the "uniform" mode in the config.

3.1.2 Sampling based on targets

This is the main successful improvement in our project.

We first try to find as equal allocations as possible using just the number of targets in the scene and the required number of samples. For example, if we have 5 targets and we require 32 samples, we sample 6 predictions belonging to each of the first 3 targets and 7 predictions from each of the remaining 2 targets. This allows us to ensure fair representation in the final prepared batch.

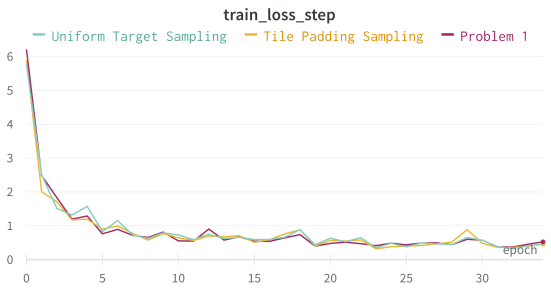
In the case where any of the targets have fewer predictions than its assigned number, say it has 3 predictions only, then we select all 3 and recompute our allocations for the remaining targets. We do this iteratively for all targets.

After all iterations, it is possible that we end up with fewer samples than required in the case that all targets have fewer predictions than the number assigned to them. In such a scenario we pad each target (using the tile method) proportionally to the number of predictions they have. This guarantees that the number of samples are selected.

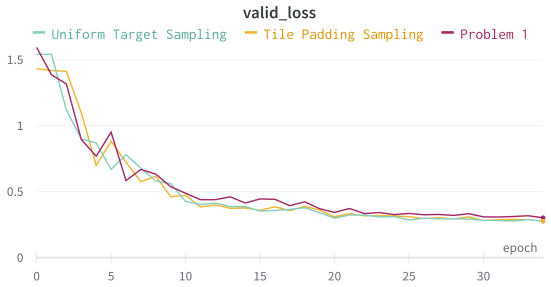
Overall it is a fairer sampling scheme than random sampling. Our experiment using this method showed a clear improvement over our previous experiments.

3.2. Adding local spatial information

One experiment that we tried was to extract local features in addition to the global ones. For that, we transformed our 3D points in their proposal bounding box coordinate system and added the distance to the origin. Using the proposed box coordinate system enables the box refinement stage to learn better local spatial features for each proposal. We then tried to use a multi layer perceptron with 2 hidden layers to have 128 channels just like we have for the global features and concatenate them together to feed our previous network. However we had many memory problems (CUDA out of memory) which we didn't manage to solve and hence

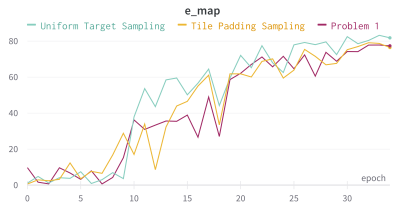


(a) Training loss

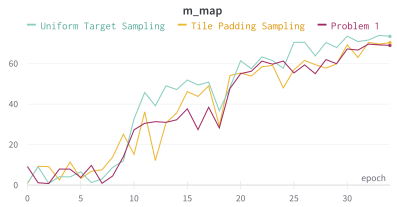


(b) Validation loss

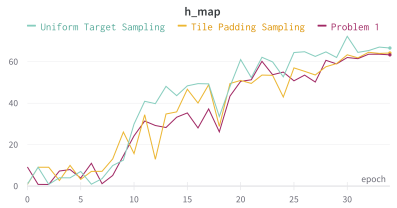
Figure 1. (a) Training loss (b) Validation loss for our 3 experiments



(a) Easy mAP



(b) Moderate mAP



(c) Hard mAP

Figure 2. (a) Easy (b) Moderate (c) Hard mAP for our 3 experiments

no results to show. The idea was taken from the paper [2], paragraph 3.3.

3.3. Bin Based Regression Loss

This is another experiment that we tried where we replaced the simple regression loss with the bin-based regression loss, used by PointRCNN [2] (Section 3.1). The authors of this paper observed that using bin-based classification with cross-entropy loss for the X and Z axes instead of direct regression with smooth L1 loss resulted in more accurate and robust center localization. The localization loss consists of bin-based classification along X and Z axes and smooth L1 loss for the Y axis. The rotation angle regression is also bin-based while the size parameters h,w,l stay the same as we had before - Smooth L1 loss We found this method interesting and adapted the code which was publicly available [1] in order to get it working with our project. However, we faced some difficulties in doing so and did not have enough time to debug the issues. We think that training with this loss will lead to some improvement as it did for the authors of PointRCNN.

4. Results

Figure 1 shows the train loss curve and validation loss curve after training the three different models. These remain very close to each other throughout. However, in Figure 2 we can see the mAP of the final experiment (Uniform Target Sampling) is clearly superior for all difficulty levels.

Table 1 shows the mAP results of the 3 experiments obtained on the public leaderboard. We can clearly see that ensuring that every training step sees all targets (almost) uniformly has helped the model performance significantly.

	Problem 1 Results	Tile Padding Sampling	Uniform Target Sampling
Easy	74.54	74.42	**81.20**
Moderate	67.52	68.59	**72.34**
Hard	61.99	63.09	**70.88**

Table 1. mAP metrics for Problem 1, Tile based sampling (Section 3.1.1), and Uniform target sampling (Section 3.1.2)

5. Conclusion

In this work, we propose different methods that we implemented and reported the results of the successful experiments. Our main contribution is a sampling technique that fairly allocates slots for each target in the final selection of 64 samples for a given scene. This method has clearly boosted our network's performance. However, the limitation of our work is that we still do not beat the expected results in the handout. We think that there may be a problem that we did not identify in Problem 1, so the best we

[1] https://github.com/sshaoshuai/PointRCNN

could do in this task is to use that model as the benchmark.

Given more time, we could have also experimented with two extra data augmentation strategies that would make the network more robust, again inspired by PointRCNN. One is to randomly augment the proposal boxes with small variations to increase the diversity of proposals. The second is to put several new ground-truth boxes and their interior points from other scenes to their same locations but in the current scene, by randomly selecting boxes that don't overlap with existing boxes.

References

[1] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3d proposal generation and object detection from view aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018. 2

[2] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 770–779, 2019. 1, 3

[3] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4490–4499, 2018. 2