Lyft 3D object detection for autonomous vehicles

Charlie Brayton
Department of Computer Engg
San Jose State University
San Jose, USA
charlier.brayton@sjsu.edu
SID: 014559415

Sowmya Mruthyunjaya
Department of Computer Engg
San Jose State University
San Jose, USA
sowmya.mruthyunjaya@sjsu.edu
SID: 002005576

I. Focus area: Autonomous driving related application

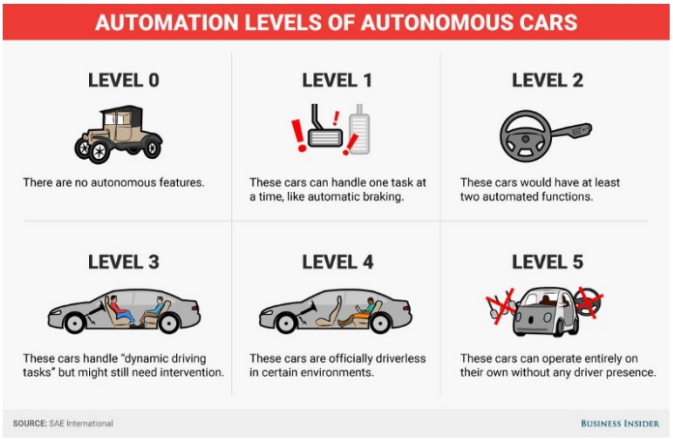
***Abstract*—Autonomous vehicles(AV) are intelligent vehicles that can function without human interference by sensing their environment. AVs can minimize accidents due to human error, are environment friendly by reducing carbon emission, can reduce travel time and transportation costs, reduce traffic congestion and increase lane capacity. A core problem to safe AVs is their ability to perceive the world around them, especially on a 3 dimensional level. Our project aims to address this problem by creating a convolutional neural network (CNN) that can process a sparse Lidar point cloud to output the 3D area that nearby objects take up. The CNN we train should be able to predict the bounding volumes of all traffic objects nearby (cars, busses, bikes, pedestrians, etc.) in a lidar point cloud as stated in the Kaggle Challenge.**

***Index Terms*—autonomous vehicle, neural networks, lidar, point cloud, object-detection, 3D bounding box**

II. Introduction

With the rapid evolution of technology, it is becoming increasingly important for the automotive industry, like every other industry, to keep up with the quick pace. Self-driving cars have gained a lot of momentum in recent years, but there is a significant difference between expectations and reality. In this vein, our research focuses on Lyft's autonomous vehicles' 3D Object Detection. Self-driving cars are one of the most talked-about inventions in the last decade. Even though many companies call their driver assistance technology "Autopilot," a fully self-driving car that operates without the assistance of a human driver on public roads has yet to be realized. Vehicles that are branded as "autonomous" are currently classified as Level 4 vehicles. Self-driving cars are classified into five levels:

- Level 0 - The lowest level is Level 0, which indicates that a vehicle does not have any driver assistance equipment. It also makes up the vast majority of vehicles on American roads.
- Level 1 - Level 1 refers to a vehicle that has driver assistance technology such as a blind spot monitoring system (BLIS), adaptive cruise control (ACC), and adaptive cruise control (ACC) and automatic braking in an emergency (AEB). It makes up the majority of new cars sold, and it has active or passive systems that only interfere when another vehicle is detected.



Source: http://post.toutptit-toutbio.com/page-3/self-driving-car-timeline-28811.html

Fig. 1. Levels of automation

- Level 2- Level 2 vehicles, which include systems like GM's Supercruise and Tesla's Autopilot, can control steering, acceleration, and braking.
- Level 3- Level 3 vehicles with devices, such as Audi Traffic Jam Assist, are fully self-driving below 37 miles per hour.
- Level 4- Level 4 vehicles include Lyft, Uber, and Waymo vehicles, which are currently being tested around the world. These vehicles are self-driving, but they must be accompanied by a person.
- Level 5 - Level 5 vehicles are fully autonomous vehicles that do not need human intervention, the most well-known of which is the Google car, which was introduced a few years ago. However, even this car can only follow pre-determined routes and therefore cannot be considered fully autonomous.

III. Dataset

We obtained the Lyft dataset from kaggle[1]. The dataset was 117 GB in size. The data was divided into two sets: training and testing. The Lyft test vehicles' 3D lidar point cloud and camera images data are included in the dataset. The data was collected on the streets of Palo Alto, California, by

ten host vehicles. On the roof of each of the host cars are seven cameras and one LiDAR sensor, as well as two small sensors underneath the headlights of the vehicle. At different fixed angles, each of the seven cameras captured photographs of its surroundings. The orange boxes in the picture below represent 55,000 human-labeled 3D annotations on items such as vehicles, pedestrians, and bicycles. Each LiDAR sensor will fire lasers in a 360-degree pattern to detect objects and obtain 3D spatial geometric data. At 10Hz, each of these LiDAR sensors produces a point cloud with 216,000 points The information was divided into 13 files, each of which acted

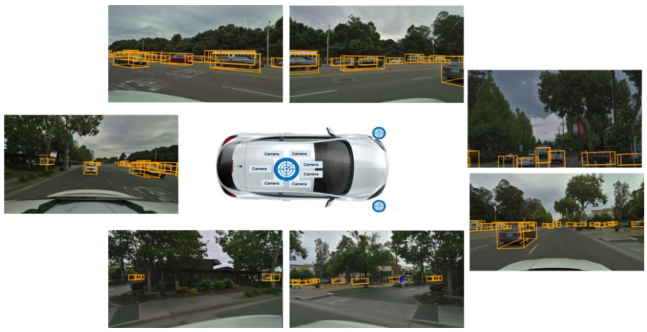


Fig. 2. Rendering of the scene

as an input to our model.

IV. Exploratory Data Analysis

The dataset consisted of raw camera images in jpeg format, 3D lidar point cloud data, and HD semantic map, data files in JSON format for both train and test set. There are 180 scenes, 25s each, 638,000 2D and 3D annotations over 18,000 objects. The metadata is in the csv file for both test and train.The figure below gives a description of the data. The

```
9 category,
18 attribute,
4 visibility,
18421 instance,
10 sensor,
148 calibrated_sensor,
177789 ego_pose,
180 log,
180 scene,
22680 sample,
189504 sample_data,
638179 sample_annotation,
1 map,
Done loading in 13.0 seconds.
======
Reverse indexing ...
Done reverse indexing in 4.8 seconds.
======
```

Fig. 3. Description of dataset

dataset had nine classes with a large class imbalance. The majority of the objects observed, as seen in figure 4 below, are cars. This is one limitation of the Lidar sensor, where it has captured objects to its side well but found it harder to capture objects in front of and behind the ego car. The categories that were not captured as much as they should have been are pedestrians, animals, trucks, bus. Autonomous driving

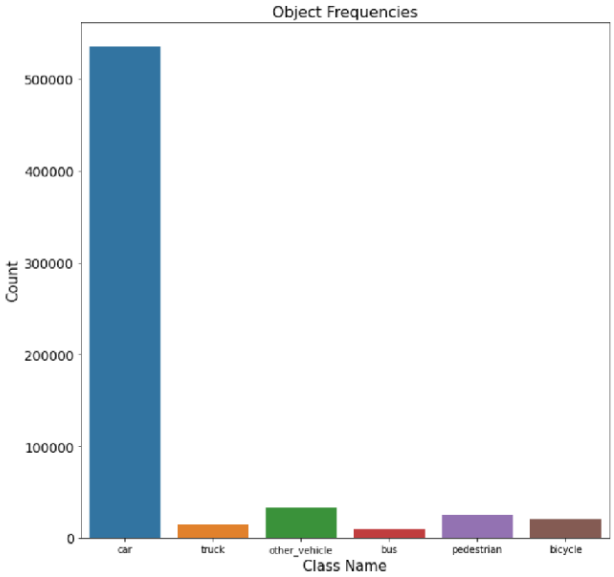
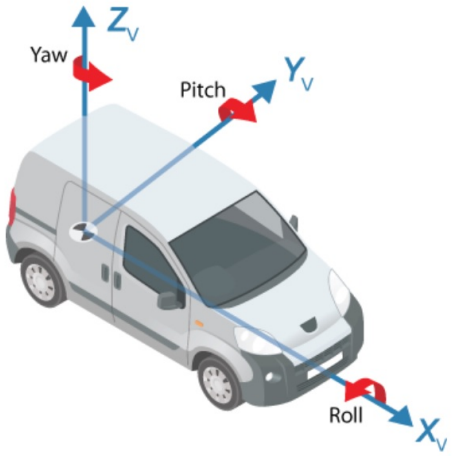


Fig. 4. Class distribution in the dataset

requires an accurate representation of the environment around the ego vehicle. The environment includes static elements such as road layout and lane structures, and also dynamic elements such as other cars, pedestrians, and other types of road users.

Lidar, an acronym for ”light detection and ranging”, is



Source: https://www.mathworks.com/help/driving/ug/coordinate-systems.html

Fig. 5. Lidar measurements

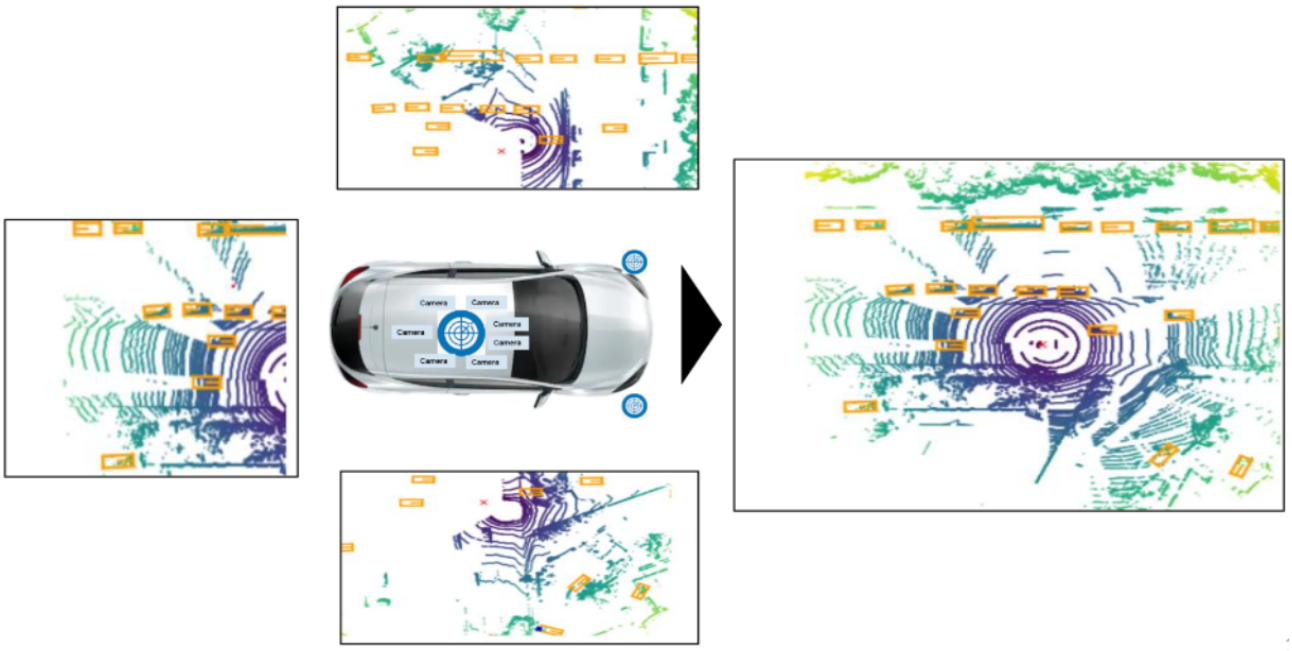


Fig. 6. Lidar to BEV conversion

essentially a sonar that uses pulsed laser waves to map the distance to surrounding objects. It is used by a large number of autonomous vehicles to navigate environments in real time. accurate depth perception, which allows LiDAR to know the distance to an object to within a few centimetres, up to 60 metres away. It's also highly suitable for 3D mapping, which means returning vehicles can then navigate the environment predictably. The LIDAR method consists of multiple sensors that capture light from different parts of the object. Lidar systems send out thousands or even hundred of thousands of laser light pulses every second. It shoots laser beams in all directions and the beams reflect off of the objects in their path and the reflected beams are collected by the sensors. The signal that is returned is processed by embedded algorithms to produce a nearly instantaneous rendering of the objects and terrain features within the view of the sensor.

For our project we have used the 3D lidar point cloud data. The x,y,z coordinates of an object, as well as its length, width, height, and yaw, are the primary measurements captured by the LiDAR, as seen in Figure 5.

The LiDAR sensors' metrics are defined as follows:

- The coordinates of an object's position on the XYZ plane are represented by centre x, centre y, and centre z.
- The orientation of the volume around the z-axis is referred to as yaw, and it is the direction in which the front of the vehicle/bounding box points when on the ground.
- The bounding volume in which the object lies is given by the length, width and height.

The lidar captures objects to the side very well compared to objects in front or behind that are mostly in-line with the car and hence difficult to capture. This is evident from figure 4 where we see high precentage of cars detected by the lidar. Humans cannot interpret lidar data with naked eyes. Since the vehicle has three separate lidar sensors, the data could not be directly fed into a neural network because the three sensors captured the same sections of the environment. We transformed our data and overlayed similar sections of the lidar points with each other because of the duplicate area issue. So we had to transform the lidar points to Bird eye view by superimposing Lidar points from three sensors into one to create Bird Eye View (BEV) as shown in figure 6. The lidar data was transformed from the sensor's frame of reference to the car's frame of reference, afterwards the car's frame of reference was transformed into the world's frame of reference to create a bird's eye view (BEV). We used the lyft SDK package for this. Our first step was to extract the lidar tokens for each scene and for each sensor. Lyft sdk package was very useful in extracting and combining the data. The next step is to voxelize the output which was a list of coordinates to an XYZ space. After this step, we created the bird's-eye view (BEV) point cloud by normalizing the voxel intensities to obtain inputs of 336 x 336 dimensions consisting of 3 channels that describe the differences in height of the lidar points.

V. Model Selection

Both segmentation and localization are essential for 3D object detection. There are two types of segmentation: instance segmentation and semantic segmentation. We used semantic segmentation because we didn't need to find different instances

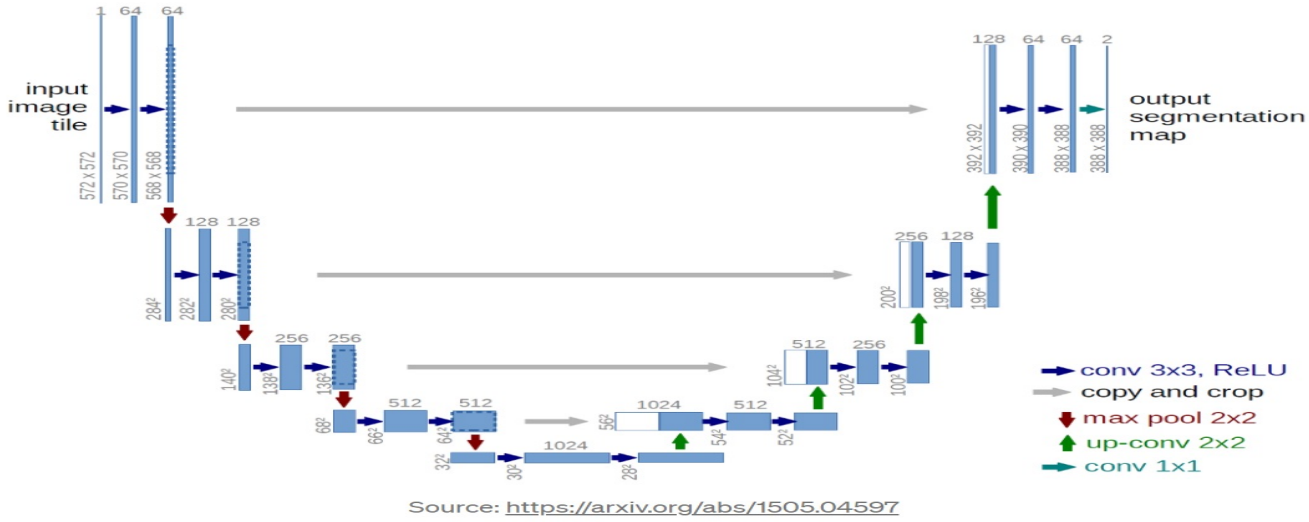


Fig. 7. Unet architecture

within a class; instead, we just needed to know whether a given object belonged to a specific class. The method of localization helps us to pinpoint the exact location of a specific object within a lidar point cloud. After exploring various CNN backbones for 3D model data, we decided to pick Unet for volumetric 3D CNN.

The U-Net is a convolutional neural network created by Olaf Ronneberger, Philipp Fischer, and Thomas Brox. They used this for biomedical image segmentation. It has produced exemplary results in image segmentation tasks. It works well with small dataset. Its symmetrical architecture gives it its name. The network does not have a fully connected layer instead uses only convolutional layers where each standard convolutional process is activated by a RELU activation process. The three key components of the U-Net architecture as show in figure 7 are:

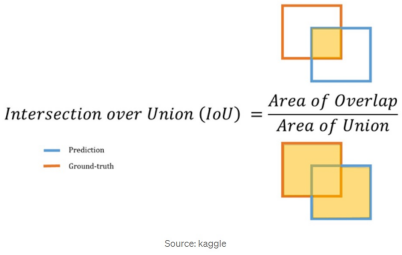


Fig. 8. IOU

$$\frac{1}{|Thresholds|}\sum_{t}\frac{TP\,(t)}{TP\,(t)+FP\,(t)}$$

mean absolute precision (mAP)

Fig. 9. mean average precision

- Downsampling or contracting path - Each of the four blocks in the contracting route has two convolutional layers and one max-pooling layer. After each block, the number of features doubles, collecting contextual details in the images. i.e., this is where the network collects the ”What” data.
- Bottleneck - Two convolutional layers with dropout make up the bottleneck.
- Upsampling path or expanding path - The expanding path, like the contracting path, has four blocks. Convolutional layers are used in blocks, and the cropped feature maps from the contracting direction are concatenated. The depth of the image decreases while the size of the image gradually increases in the expansion direction. This is how the “Where” detail in the pictures is captured.

VI. Training and Evaluation

To achieve 3D object detection, we require both segmentation as well as localization. Localization is the method that allows us to pinpoint where a certain object is inside a lidar point cloud.

We trained a U-Net fully connected convolutional neural network to predict whether an object is present in the BEV and later threshold this probability map to fit boxes around the detections. Training took approximately 8 hours from start to finish. Serveral runs had to be restarted, however, as the session crashed from running out of memory or disk space. To solve this issue, we had to reduce the dataset size, which allowed the machine to complete the run without crashing. We used lidar data from host car 7 for training which included 26 different scenes and lidar data from host car 9 for validation with 9 different scenes.

For evaluation, we used Intersection over union IOU. To test the object detection model, the average precision is determined at various Intersection over Union (IoU) thresholds. The area of the overlapping region divided by the total area of union is

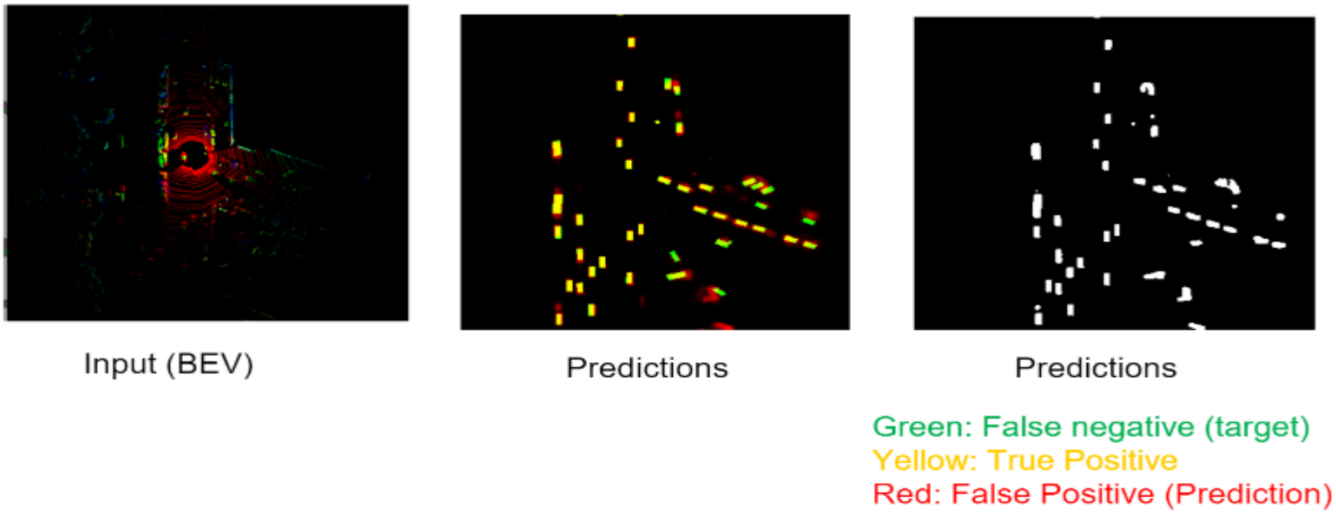


Fig. 10. Validation Predictions

the IoU of a 2D bounding box, as illustrated in figure 8. With a phase size of 0.05, the IoU is measured at thresholds ranging from 0.55 to 0.95. For example if a particular item is detected with an IoU greater than 0.55, then the item is a prediction at a 0.55 threshold. By comparing the expected object to all the ground-truth objects at each such threshold, a precision value is determined using True Positives (TP), False Positives (FP), and False Negatives (FN).

The proportion of true positives to false positives and true positives is the precision value. When the network correctly predicts the existence of an object, it is called a true positive (TP). If there is no related ground truth for the expected object, we classify it as a false positive (FP). To assess the model, we use the equation in figure 9 to measure the mean precision of the above precision values over various thresholds.

VII. Result

The Unet architecture allows us to do fast and precise segmentations. We input the top-down projection of the world around the car into our U-Net architecture to create semantic maps. The result from the architecture is the semantic map in the BEV as shown in figure 10. For these images, the left image shows the input Lidar data, where color denotes object height. The center image is an overlay of the ground truth objects in green and the predictions in red. For the center object yellow is the color of overlaps, which signify accurate predictions or true positives. Finally the right image shows the predictions alone. Similarly the effects of thresholding the predicitions can be seen in figure 11. The left image of figure 11 shows the raw predictions, the center shows those same predictions after low scoring predictions under a threshold of 128 have been filtered out. Our validation loss was around 0.11 which is not bad given we trained on partial dataset. The overall loss during training can be seen in figure 12. However we believe that if continued to train the model for more epochs and use the entire dataset the loss will come down further. To generate 3D bounding boxes from the BEV predictions we simply add the voxel height component measured by the Lidar. Unfortunately the our mean average precision (mAP) is rather low at 0.013857849213 for all classes across thresholds 0.5 to 0.95. Looking only at the cars class the mAP is 0.03459820093. While these scores aren't great, the winner of the competition had a score of 0.216 and the top 50th score was 0.053. The overall reason for the low scores is probably because with 3 dimensions the IOU bounding box threshold becomes more sensitive. Our scores could be further improved by continuing to train the model with Lidar data from other host cars. This would require more time and more computing resources, but should make our mAP scores more competitive. An inference video with 3D object detection has been submitted with this report and also attached in the presentation slides that was submitted on the day of the presentation.

VIII. Future work

With better computational power, the model can be tuned efficiently. During this project we ran into lot of problems due to run timing out several times during training. We could try models with different architectures to see if the mAP improves and and with new architecture we are able to predict small objects as well. Hopefully, the architectural variations, combined with increased computational capacity, would result in improved precision. We have used only lidar data in this project. We can probably explore the camera images data for 3D object detection. Most of the other models used in the competition are ensembles, so they combine the output of various models to improve the overall predictions. A future project could combine our Lidar CNN model with a CNN model trained on the camera images to combine the two for improved results.

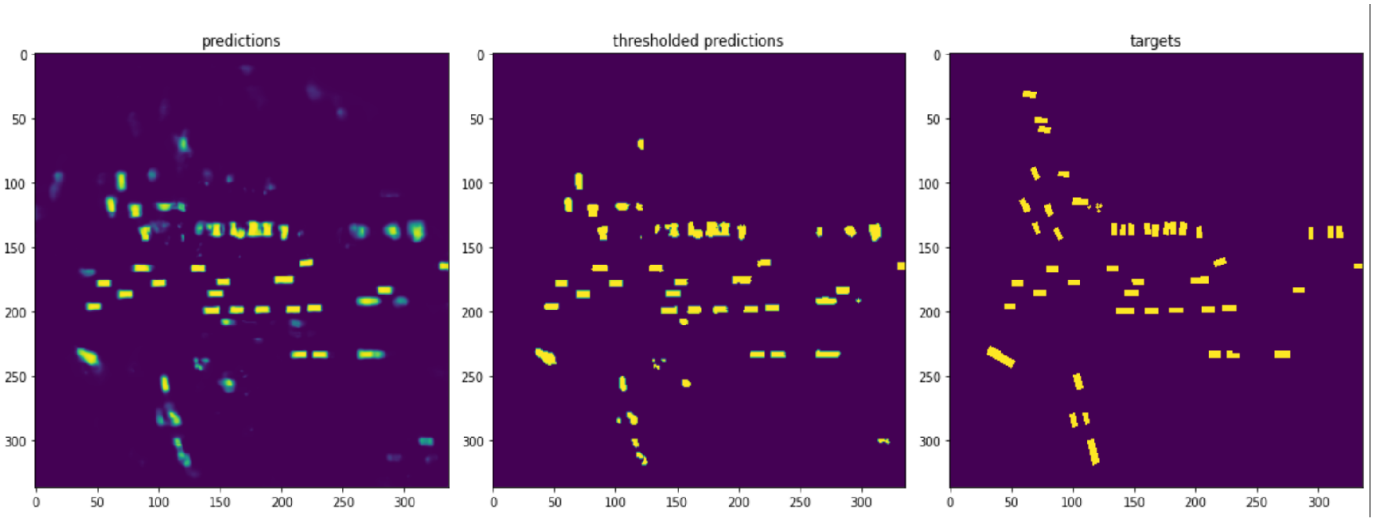


Fig. 11. Predictions after thresholding

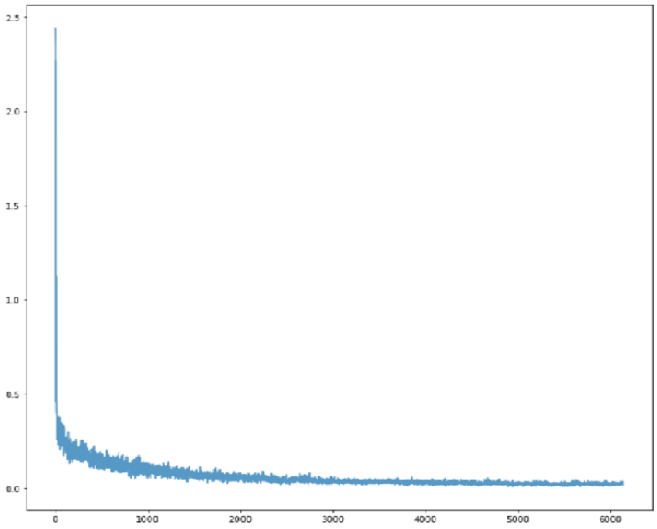


Fig. 12. Training loss function

IX. Project contribution

Charlie - Modify the starter code from Guido Zuidhof a Lyft machine learning engineer here: https://www.kaggle.com/zrstupid01/reference-model. Edit and give half of the presentation. Write and edit parts of the final report. Generated mAP results.

Sowmya - Research various datasets, creating and giving presentation, writing and formatting final report. Remote pair programming for modifying the starter code, providing insights into next steps for project/code.

Jointly, we performed EDA to understand the data and how token, scenes and Lidar data was used in the referrence code, went through the Unet architecture to understand how it works in image segmentation as this was a new CNN backbone for us and this project helped us learn it. We ran the code on google colab pro using Charlie's account (bought it for the sake of this project as the run was crashing badly on local machine). Validation and prediction was performed jointly as we worked during different timezones.

X. Conclusion

The learning was definitely steep while working on this project. This project helped us learn a lot about lidar data, U-net architecture, how to convert lidar data to Bird's eye view etc. The main problem we faced was computational capacity. Training took a long time. We could not increase the batch size beyond 8. We had to get through the initial run by using only a part of the data. The model was able to detect the 2D footprint of objects from the bird's eye view. 3D bounding boxes were then created using the Lidar's height measurements for each detected object.

Acknowledgment

We would like to express our sincere gratitude to Dr. KaiKai Liu and Dr. Wencen Wu for their knowledge and guidance throughout the course. Both the professors are subject matter experts, very student friendly and have answered all our queries with utmost patience and have guided us during times of need.

References

[1] https://www.kaggle.com/c/3d-object-detection-for-autonomous-vehicles/discussion/133895
[2] Deep Continuous Fusion for Multi-Sensor 3D Object Detection Ming Liang, Bin Yang, Shenlong Wang, Raquel Urtasun; Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 641-656
[3] Vote3Deep: Fast Object Detection in 3D Point Clouds Using Efficient Convolutional Neural Networks Martin Engelcke, Dushyant Rao, Dominic Zeng Wang, Chi Hay Tong, Ingmar Posner 2017 IEEE International Conference on Robotics and Automation (ICRA)
[4] https://towardsdatascience.com/3d-object-detection-for-autonomous-vehicles-b5f480e40856
[5] https://www.kaggle.com/zrstupid01/reference-model
[6] https://github.com/sijopkd/3d-object-detection-for-autonomous-vehicles/tree/master/codes