

Lyft 3D object detection for autonomous vehicles



CMPE 249 Spring 2021
Instructor: Dr. Wencen Wu

Charles Brayton
Sowmya Mruthyunjaya

Objective

To build a machine learning model for 3D object detection in autonomous vehicles using lidar data.



AUTOMATION LEVELS OF AUTONOMOUS CARS

LEVEL 0



There are no autonomous features.

LEVEL 1



These cars can handle one task at a time, like automatic braking.

LEVEL 2



These cars would have at least two automated functions.

LEVEL 3



These cars handle "dynamic driving tasks" but might still need intervention.

LEVEL 4



These cars are officially driverless in certain environments.

LEVEL 5



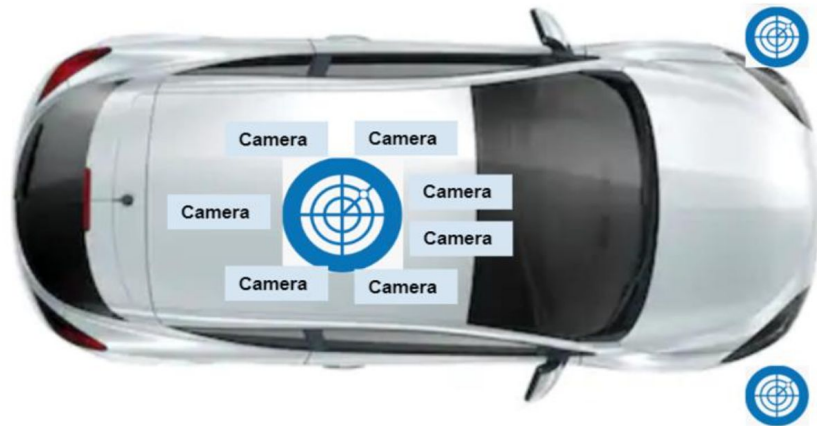
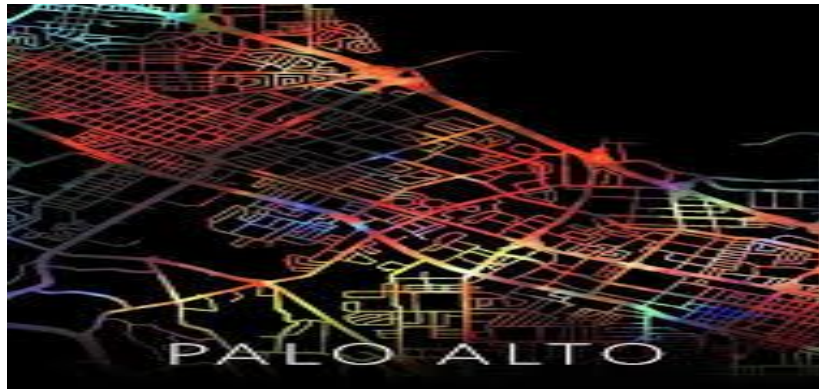
These cars can operate entirely on their own without any driver presence.

Dataset

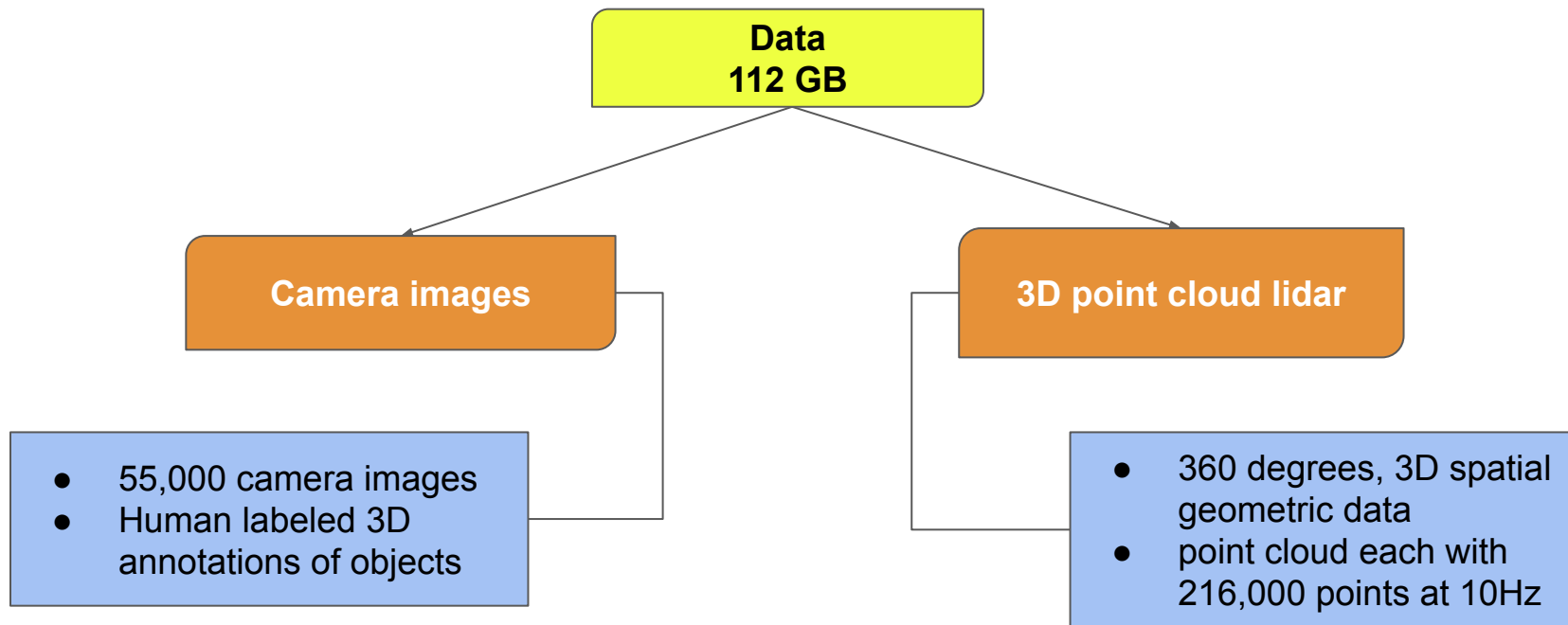
Source:

<https://www.kaggle.com/c/3d-object-detection-for-autonomous-vehicles/data>

- Data collected by 10 host cars driving on the streets of Palo Alto, CA
- Each host car had 7 cameras, one lidar sensor on the roof and 2 sensors underneath the headlight.

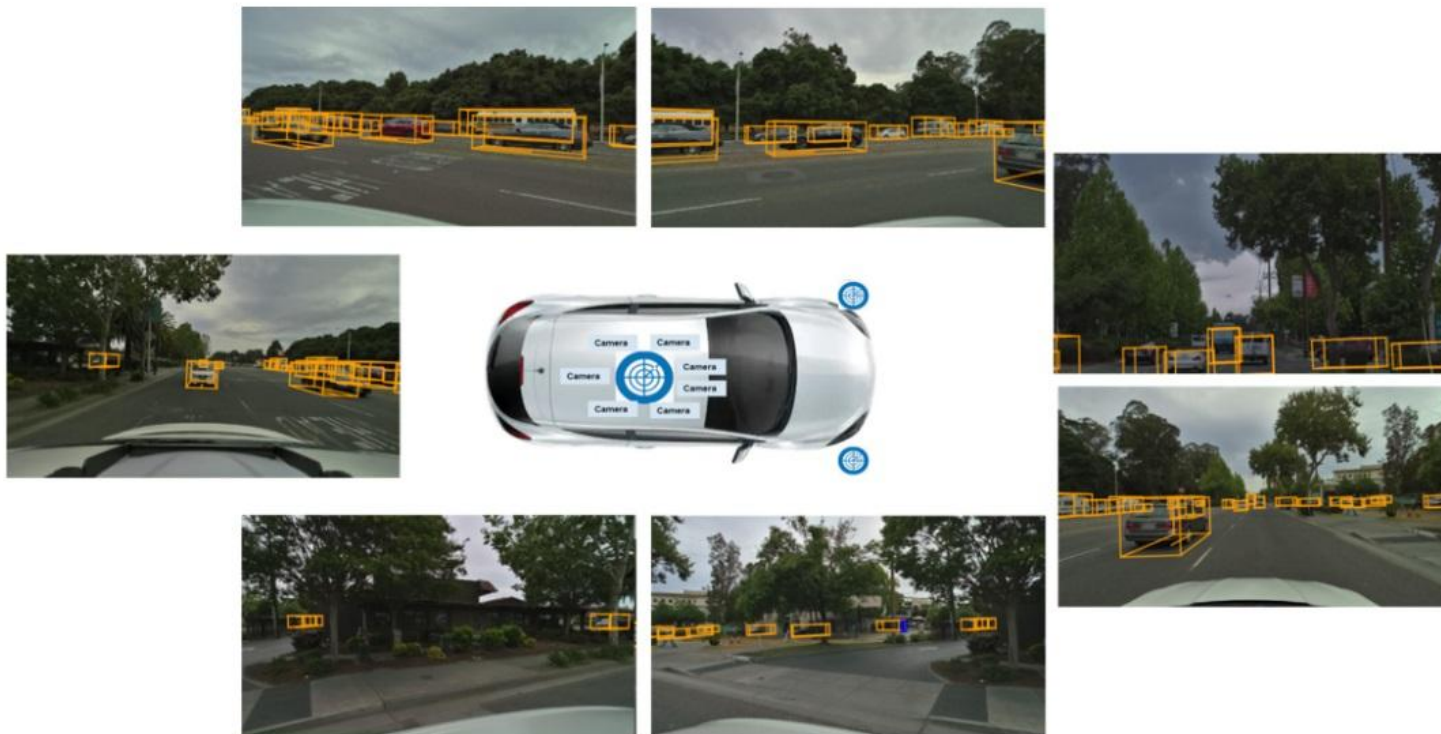


Dataset



The lidar and image data were split into training and testing sets and made up the large dataset.

Scene rendering



Lidar

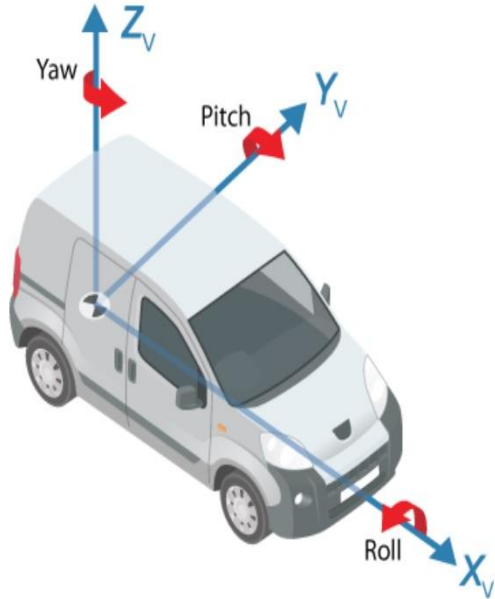


“light detection and ranging”, is essentially a sonar that uses pulsed laser waves to map the distance to surrounding objects.

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.

Primary measures captured by Lidar data



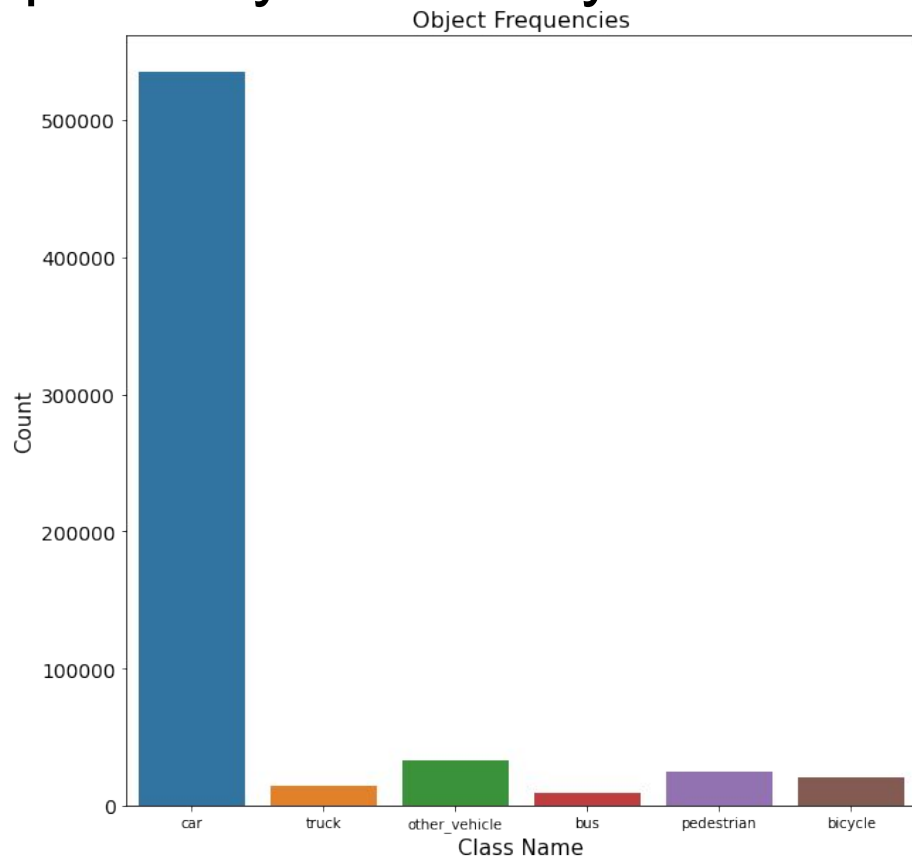
Source: <https://www.mathworks.com/help/driving/ug/coordinate-systems.html>

- x, y, z coordinates of an object

- length, width, height of an object

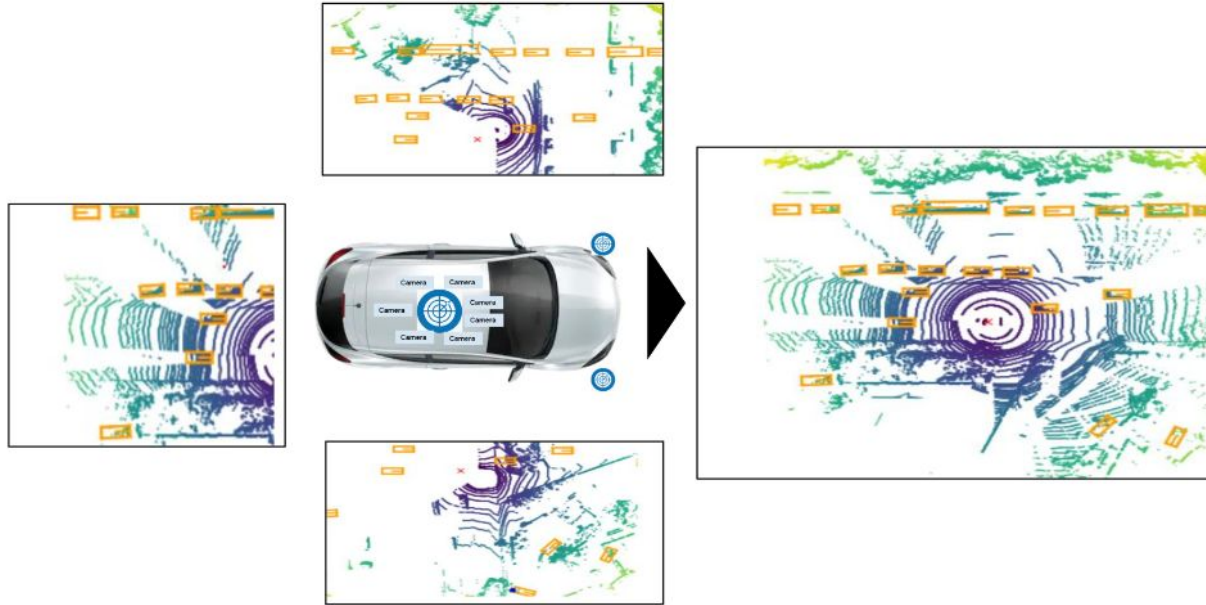
- yaw

Exploratory data analysis



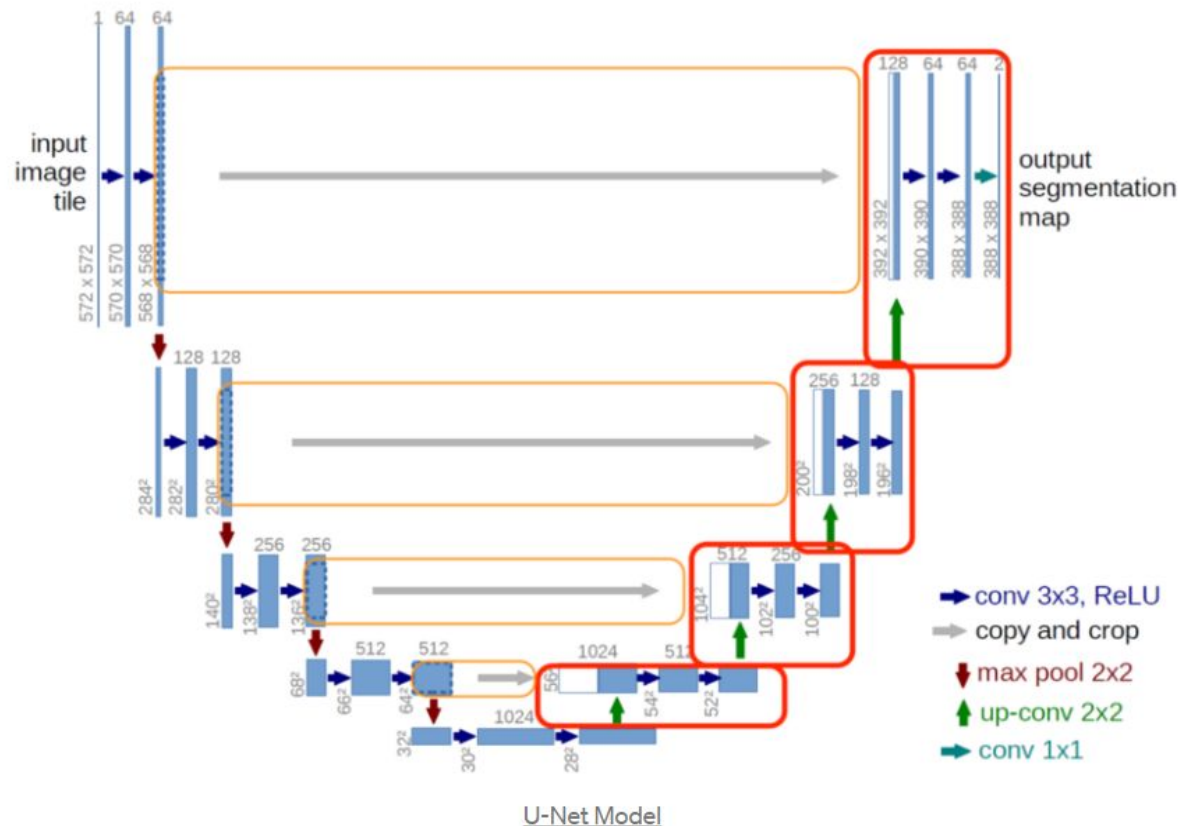
- 9 classes of objects
- High frequency of cars detected
- Imbalanced data

Data transformation - Lidar to BEV



- Hard to process Lidar data with naked eyes
- Used lyft SDK package to transform lidar data to BEV (Bird's Eye View)

Model - Unet



- CNN that models 3D data very well
- Symmetric architecture
 - Contracting
 - Bottleneck
 - Expanding

Model - Unet

```
DataParallel(  
  (module): UNet(  
    (down_path): ModuleList(  
      (0): UNetConvBlock(  
        (block): Sequential(  
          (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
          (1): ReLU()  
          (2): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
          (3): ReLU()  
        )  
      )  
      (1): UNetConvBlock(  
        (block): Sequential(  
          (0): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
          (1): ReLU()  
          (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
          (3): ReLU()  
        )  
      )  
      (2): UNetConvBlock(  
        (block): Sequential(  
          (0): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
          (1): ReLU()  
          (2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
          (3): ReLU()  
        )  
      )  
      (3): UNetConvBlock(  
        (block): Sequential(  
          (0): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
          (1): ReLU()  
          (2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
          (3): ReLU()  
        )  
      )  
    )  
  )  
)
```

```
(up_path): ModuleList(  
  (0): UNetUpBlock(  
    (up): Sequential(  
      (0): Upsample(scale_factor=2.0, mode=bilinear)  
      (1): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1))  
    )  
    (conv_block): UNetConvBlock(  
      (block): Sequential(  
        (0): Conv2d(256, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
        (1): ReLU()  
        (2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
        (3): ReLU()  
      )  
    )  
  )  
  (1): UNetUpBlock(  
    (up): Sequential(  
      (0): Upsample(scale_factor=2.0, mode=bilinear)  
      (1): Conv2d(128, 64, kernel_size=(1, 1), stride=(1, 1))  
    )  
    (conv_block): UNetConvBlock(  
      (block): Sequential(  
        (0): Conv2d(128, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
        (1): ReLU()  
        (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
        (3): ReLU()  
      )  
    )  
  )  
  (2): UNetUpBlock(  
    (up): Sequential(  
      (0): Upsample(scale_factor=2.0, mode=bilinear)  
      (1): Conv2d(64, 32, kernel_size=(1, 1), stride=(1, 1))  
    )  
    (conv_block): UNetConvBlock(  
      (block): Sequential(  
        (0): Conv2d(64, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
        (1): ReLU()  
        (2): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))  
        (3): ReLU()  
      )  
    )  
  )  
  (last): Conv2d(32, 10, kernel_size=(1, 1), stride=(1, 1))  
)
```

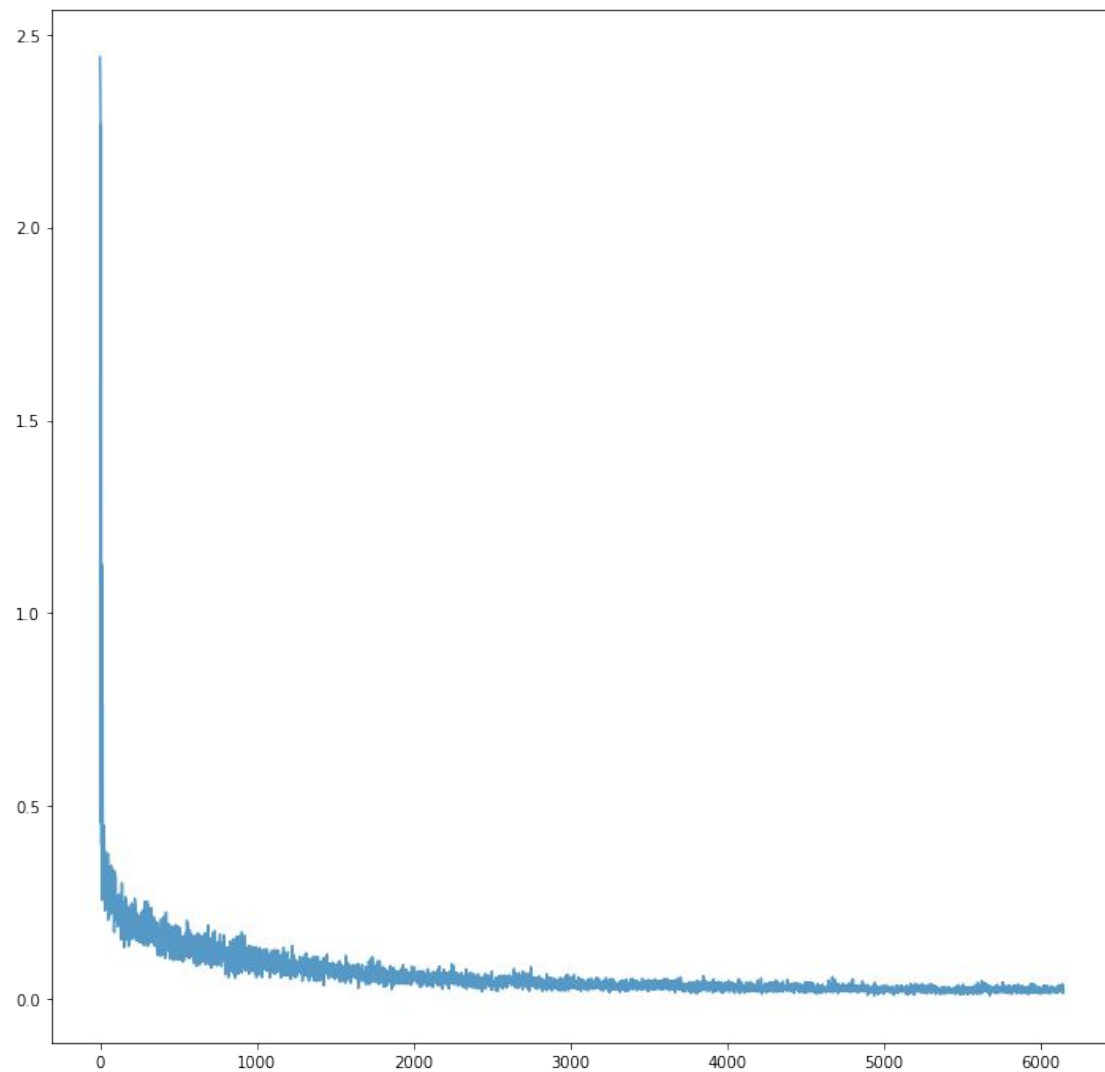
Training

- Trained U-net model to detect presence of an object.
- Threshold probability map to fit boxes around the detection.


Roadblocks

- Resource and computational constraint as the dataset was large.
- Training took around 8 hours. Had to save checkpoints.
- Used part of the data in order to overcome computational constraints and complete the project in a timely manner.

Training Loss



Evaluation


$$\text{Intersection over Union (IoU)} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

— Prediction
— Ground-truth



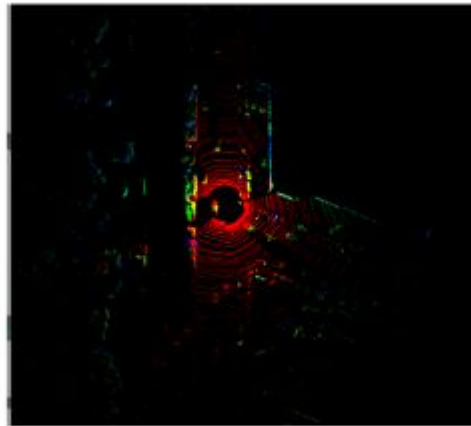
Source: kaggle

$$\frac{1}{|\text{Thresholds}|} \sum_t \frac{TP(t)}{TP(t) + FP(t)}$$

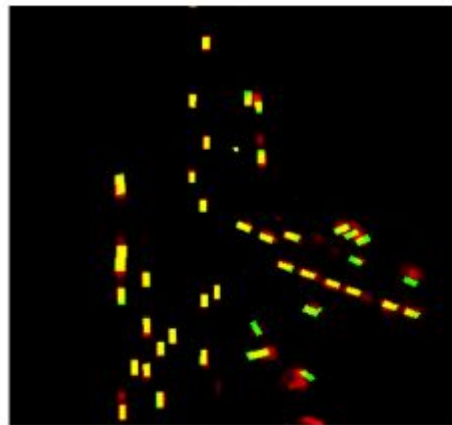
mean absolute precision (mAP)

- $\text{IoU}(A,B) = \frac{A \cap B}{A \cup B} > \text{threshold}$
- IOU measured at varying thresholds between 0.5 and 0.95
- Phase size: 0.05

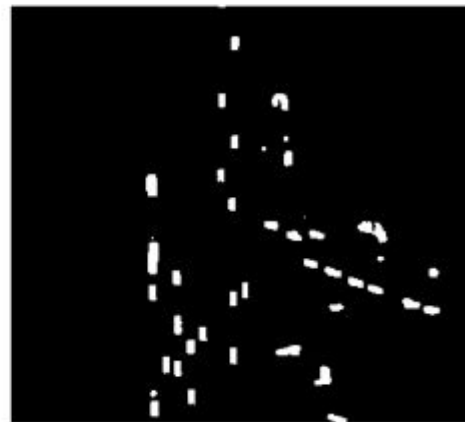
Results



Input (BEV)



Predictions

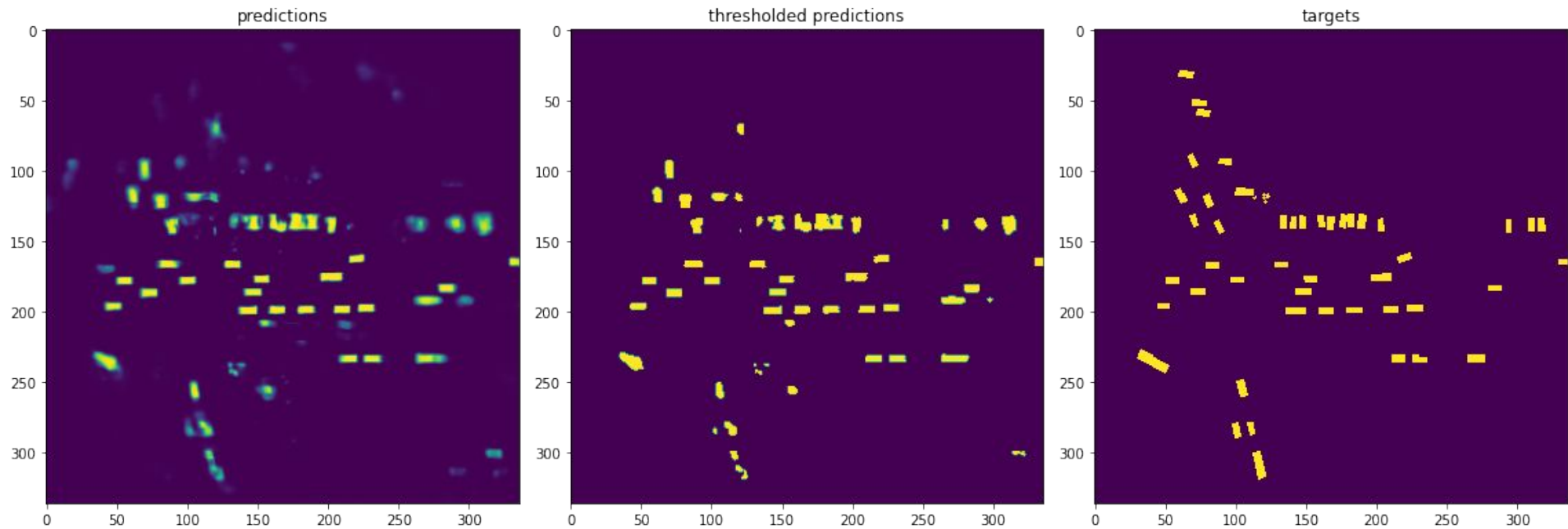


Predictions

Green: False negative (target)
Yellow: True Positive
Red: False Positive (Prediction)

Results - Validation Predictions

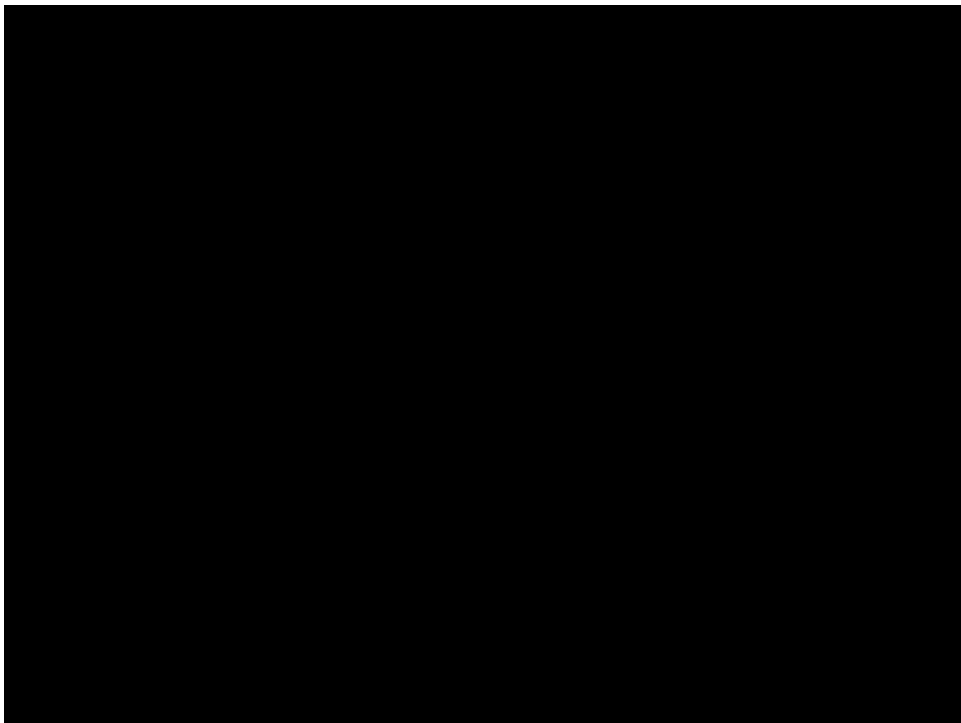
Mean validation loss: 0.110944934



Next Steps

- Add HD map to outputs for greater context
 - Potentially as input with retraining to improve accuracy
- Improve inference and visualisation
 - Ideally short video clip

Demo



Future work

- Could try models with different architectures to see if the mAP improves
- Be able to better predict small objects with new architecture
- Explore the camera images data for 3D object detection
- Try motion prediction

References

- <https://www.kaggle.com/c/3d-object-detection-for-autonomous-vehicles/>
- 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
- Stereo R-CNN based 3D Object Detection for Autonomous Driving Peiliang Li, Xiaozhi Chen, Shaojie Shen (<https://arxiv.org/abs/1902.09738>)
- 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)
- <https://towardsdatascience.com/3d-object-detection-for-autonomous-vehicles-b5f480e40856>



Thank
You !