**Waymo Open Dataset Challenge: 2D Detection**

Harish Kamath, Rohit Mittapalli, Aman Kishore, Ashwin Rathie, Mayank Kishore

Georgia Institute of Technology and University of Illinois Urbana Champaign

Background

We are a group of Computer Scientists and Computer Engineers from Georgia Tech and UIUC highly interested in autonomous technology. The Waymo Open Dataset challenge was an opportunity to use what we have learned through research and industry experience to train the best model possible. We had a lot of fun working through the challenges of this problem and wish the best of luck to all of the other teams!

Dataset & Pre-processing

Dataset was provided to us in tfrecords and we decided to convert it to a COCO format for more ease of use with existing models.

* Only data provided by Waymo was used, no external data sources
* Used a Waymo Dataset Tool created by Github user, RalphMao [1] to convert the data to a KITTI format. Afterwards, we were then able to write a python script to convert the data to COCO format for easy use with the mmdetection library [2].

Initial Visualization with Out of Box Models

To make sure we were selecting a good out of box model, we tested the models with a small subset of the data to determine if our model would be viable. We were able to create a python script that allowed us to visualize our bounding box results. After testing a few initial models from paperswithcode.com, we were able to decide on the Cascade R-CNN model. With this initial step done, we dove deeper into the research to ensure that our model’s initial parameters would ensure the best results.

Model Selection

We selected the Cascade R-CNN model and the implementation is based off of the popular mmdetection library [2], available with a lot of out of box models for easy deployment. We reviewed the top models on <https://paperswithcode.com/task/object-detection> and decided that Cascade R-CNN would give us the best results without having to change a lot of the base model settings due to our initial visualizations. That being said, our pipeline consisted of the following:

* Backbone model of SpineNet [3]
  + The architecture consists of a fixed stem network followed by a learned scale permuted network. A scale-permuted network is built with a list of building blocks where each block has an associated feature map. Then, each block can be scaled and adjusted accordingly to output the best possible results.
  + Beyond a convincing architecture, we chose this backbone because it is a recent model released by the Google Brain team particularly well suited for object detection. In addition, the model has posted results of outpacing the traditionally excellent ResNet model as a great backbone model.
* Cascade R-CNN [4]
  + We chose this model because the use of cascade regression as a resamploing mechanism allows for an IoU threshold of 0.7, which is significantly higher than the traditional threshold of 0.5 that most CNN models use.
  + This cascade learning has three important consequences for detector training. First, the potential for overfitting at large IoU thresholds u is reduced, since positive examples become plentiful at all stages. Second, detectors of deeper stages are optimal for higher IoU thresholds. Third, because some outliers are removed as the IoU threshold increases, the learning effectiveness of bounding box regression increases in the later stages.
  + Resampling progressively improves hypothesis quality, guaranteeing a positive training set of equivalent size for all detectors and minimizing overfitting. The same cascade is applied at inference, to eliminate quality mismatches between hypotheses and detectors.

Model Improvement

Our initial test yielded poor results with SpineNet, so we reverted back to a more familiar model that has been tried and tested, ResNet. In addition, we scaled the resolution of the images down to allow for much faster run time. This allowed the model to train much faster and give us a higher precision in the end. We also changed it so that the model would run for 30 epochs on every camera

* ResNet [5]
  + We switched back to ResNet as the backbone for our Cascade R-CNN as we determined that it would be a more reliable way to train the model. Cascade R-CNN had been thoroughly tested with a ResNet backbone and seemed to have astounding Average Precision scores. We tested ResNet over one epoch and it had much more accurate results as compared to SpineNet.
  + We also had to decide how many layers we wanted ResNet to have. Based on the Cascade R-CNN Paper [7], ResNet-50 had an AP of 41.3 for object detection and ResNet-101 had an AP of 43.3 for object detection. We decided to train on ResNet-50 as we felt the added complexity of ResNet-101 would be too inefficient to be worth the higher precision.

Considerations

Moving forward, if given more time, we would have liked to try to train our model for an increased number of epochs and an increased number of images. We were only able to train on about 80% of the data and were only able to run our Cascade R-CNN model for about 30 epochs per camera which is less than desirable. We would have loved to use the entirety of the data and run it for an increased number of epochs to maximize the mAP for the requested classes.

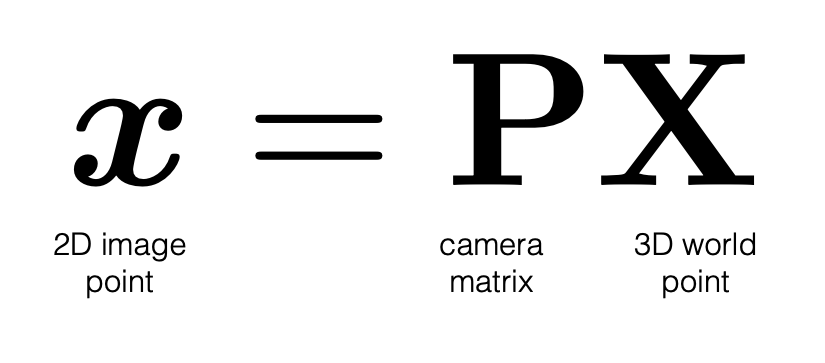
In addition, we would have liked to use Test Time Augmentation (TTA) to create multiple augmented copies of each image in the test set. Then, once we have a model make a prediction for each, we could ensemble the best models together to get the best possible combination of results. This could have drastically improved our model as it would have allowed us to potentially use non-max suppression to develop the best possible bounding boxes with an overall advantageous IoU threshold which would have ultimately allowed for an increase in overall mAP over the Vehicle, Pedestrian, and Cyclist classes.

3D Object Detection and Domain Adaptation

In addition to creating a robust 2D detection model, we also wanted to see if we could create a depth map with our inputted images in order to tackle the problem of 3D Detection and domain adaptation. In order to use our 2D detection model to give depth values as well we:

As stated in [6], “One interesting possibility that can be explored using the dataset is the prediction of 3D boxes using camera only.” This was the task we decided to attempt. To generate 3D bboxes, we leveraged our 2D detection model alongside a monocular and stereo hybrid depth estimation. This enables depth estimation on just a single source image, as opposed to stereo methods [8].

In order to map our 2D results, which are in the Camera frame, to 3D results, which are in the Vehicle Frame, we utilized the intrinsics and extrinsics according to the following formulas.



Eq. 1 [9]



Eq. 2: Where K is the calibration (intrinsic) matrix and [R|t] is the extrinsic matrix

Results

We trained our model on 5 GCP instances to maximize the amount of data that was processed. We used the Tesla V100 GPU to train our models to minimize the time spent training and were able to finish the 30 epochs within a week. Each GCP instance took one of the 5 cameras and ran for 30 epochs which yielded an mAP of \_\_\_ for Vehicles, \_\_\_ for Pedestrians, and \_\_\_ for cyclists.

References

[1] <https://github.com/RalphMao/Waymo-Dataset-Tool>

[2] <https://github.com/open-mmlab/mmdetection>

[3] <https://arxiv.org/abs/1912.05027>

[4] [https://arxiv.org/pdf/1712.00726](https://arxiv.org/pdf/1712.00726.pdf)

[5] <https://arxiv.org/abs/1512.03385>

[6] <https://arxiv.org/pdf/1912.04838.pdf>

[7] [https://arxiv.org/pdf/1906.09756](https://arxiv.org/pdf/1906.09756.pdf)

[8] <https://arxiv.org/pdf/1806.01260.pdf>

[9] <http://www.cs.cmu.edu/~16385/s17/Slides/11.1_Camera_matrix.pdf>