Springboard–DSC Program
Capstone Project Three: Car Model Detection

By: Kris Simino

May 26, 2022

Introduction

Object detectors scan through an image to look for particular objects. With algorithms like YOLO[1], the speed of detection can be so fast that it works on real time video. Unfortunately, the number of classes is limited to around 80 to make the model most efficient. Also, training object detection models take very large datasets and time.

Using a pre-trained YOLOv5 model, can we take its general classes and expand them to specific classes? In this project, a model will be utilized to detect the presence of automobiles and, when detected, a separate model will then classify them into their specific manufacturer makes.

These models could have wide applications in fields such as security, law enforcement, or automotive marketing. For this project, let us assume the client is a hotel chain that would like a system which can monitor a parking lot for unauthorized vehicles. Guests at the hotel must report the make and model of their vehicle at check-in. They wish to know: how many cars are in the parking lot and what types of cars are in the parking lot. The classification should achieve a precision of greater than 95%. In this way, only one out of twenty cars would need to be confirmed as authorized.

[1] YOLO official site: https://pjreddie.com/darknet/yolo/

Approach

Data Acquisition and Wrangling

Two main datasets are employed: Audi Autonomous Driving Dataset (A2D2)[2] and DVM-Car dataset. The Audi set is a publicly available set of dash cam images and sensor data. It includes 3D bounding boxes, semantic segmentation, and 3D point cloud segmentation. However, only the dashcam images and 2D boxes are used.

The DVM-Car dataset[3] is a publicly available UK automotive market based research dataset. It consists of nearly 1.5 million images of 899 car models. Each image is 300x300 and the car is set into a white background. Each car model has images of different angles and colors.

Both datasets required no cleaning and were well organized and labeled.

Storytelling and Inferential Statistics.

A2D2 dataset

The intent of the Audi dataset is to give realworld images for the final model to experiment on. It is not used to train the detection model. The YOLOv5 model that is used has pretrained weights and has had its performance tested and evaluated previously.

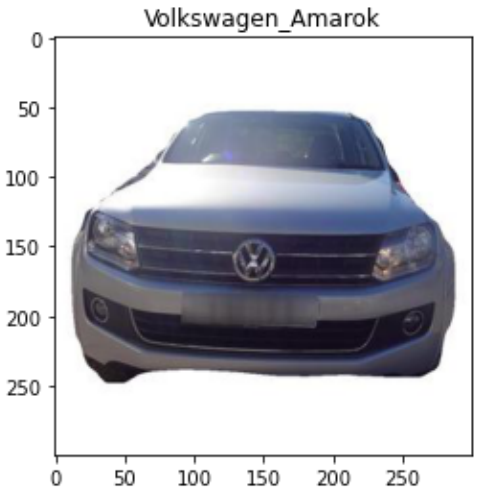
The set of images consists of 12,499 dash cam frames. Not all frames contain an automobile but each frame that does contain an automobile has labeled bounding boxes for them. What will be discovered later is cars that are obscured or very distant are not labeled.

[2] The Audi autonomous driving dataset (A2D2): https://www.a2d2.audi/a2d2/en.html
[3] "DVM-CAR: A large-scale automotive dataset for visual marketing research and applications": https://deepvisualmarketing.github.io/

Dash Cam Image Example

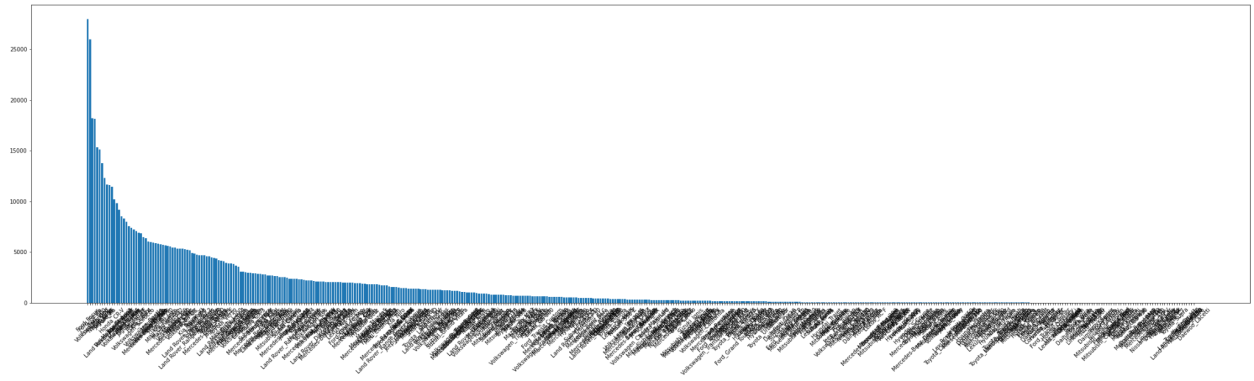
DVM-Car dataset

This set is used to train the car classifier that is used in the final model. It consists of nearly 1.5 million images of cars at various angles. The 899 different classes of cars are separated by make, model, and year. Each image is 300x300 and is a car that has been cropped along its edge and set into a white background.



Example car image

As can be seen in the chart below, the proportion of car models was extremely skewed. In order to make the most of this dataset, many of the classes that contain very little data, some with only one image, were removed.



The next step in eliminating some of the smaller classes was to remove brands that were very rare, like Ferraris and Lamborghinis, or brands that were not sold in the United States, like London Taxi or Koenigsegg. These cars are unlikely to be seen in most dashcam images. An “Unknown” class will be added. Also, each class was limited to make and model and disregarded the year. This left 715,779 images and 456 classes. The data set was still very skewed with the largest class, Ford Focus, comprising 3.9% of the data and the smallest class, Hummer H1, comprising 0.0001% of the data.

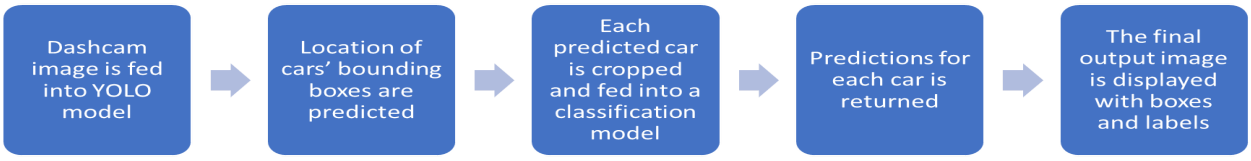
Training a model on this dataset resulted in poor performance, as seen in the next section. So, with the intent to revisit and improve the model later, the training data was further reduced in classes. This was simply achieved by only considering the car’s make. Now the number of classes was 26 which included an “Unknown” class.

In both cases, the original data was shuffled and 10% of the data was randomly selected for a test set. The rest of the data was split into an 80/20 train/validation datasets.

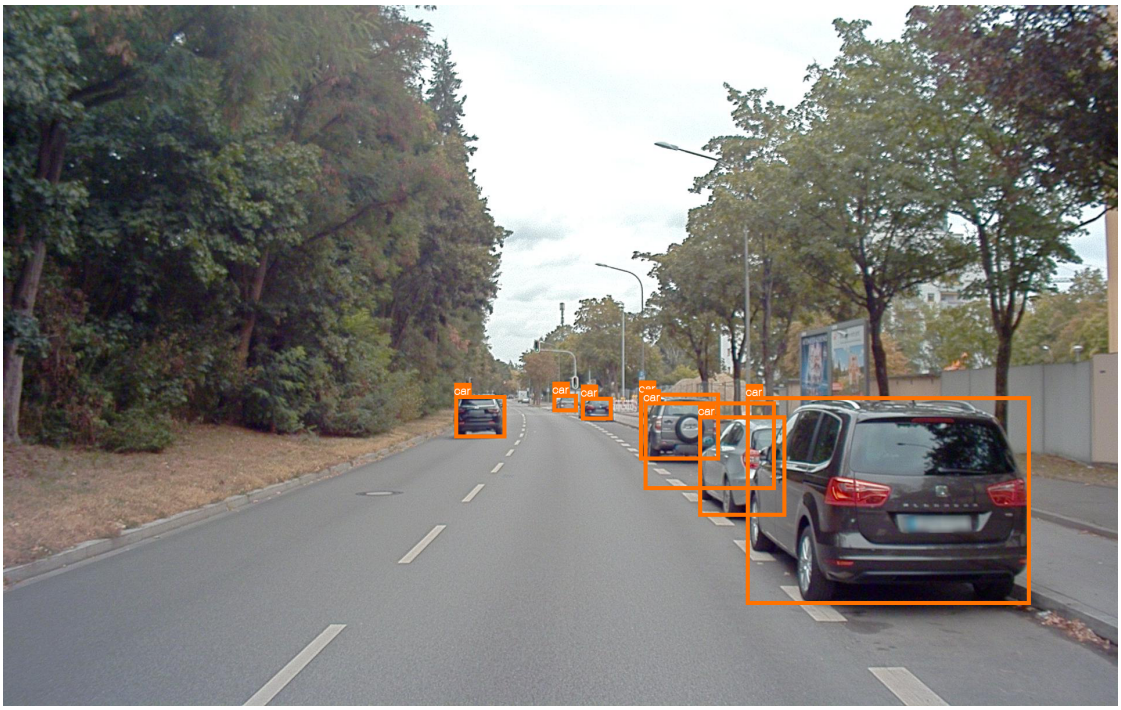
Modeling

Object Detection

The final model should take a dashcam image, search for the presence of any automobiles, isolate those portions of the picture with an automobile and crop them out, feed the cropped car images into a classifier, and then finally return the original dash cam image with each car labeled and surrounded by a bounding box.



The first step was to upload the YOLOv5 model and its pretrained weights. A random image from the Audi set was used to test the model's function.

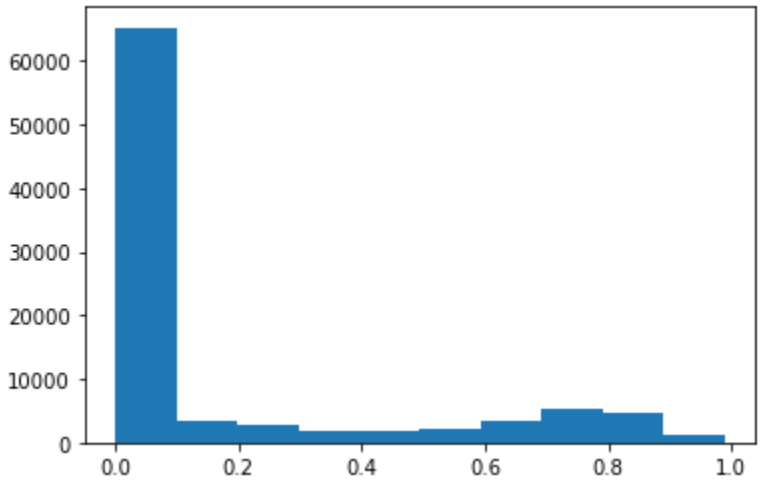


As can be seen in the above image, the YOLO model effectively detects cars in the image and boxes them. However, upon closer inspection, it appears that an additional box is present that does not represent a car. This is essentially a false positive between closely spaced vehicles. What other issues are present?

Every image in the Audi dataset was then run through the YOLO model. The cars that were detected were compared to the labeled ones for each dashcam image. On average, the model detected three more cars than were labeled. After examining some of these images, it became apparent that the labeled images only recognized cars that were not too distant or too obscured by other objects. Whereas the YOLO model reported anything that might be a car.

	Boxes_Known	Boxes_Predicted
count	12497	12497
mean	2.2	5.6
Min	0	0
Max	17	29

To further test the detection model, the IOU (intersection over union) score was calculated for each labeled box and its matching predicted box. This should tell us how well the model detects the edge of the labeled car with an IOU score of one being a perfect match and a score of zero meaning the predicted box completely missed the car.



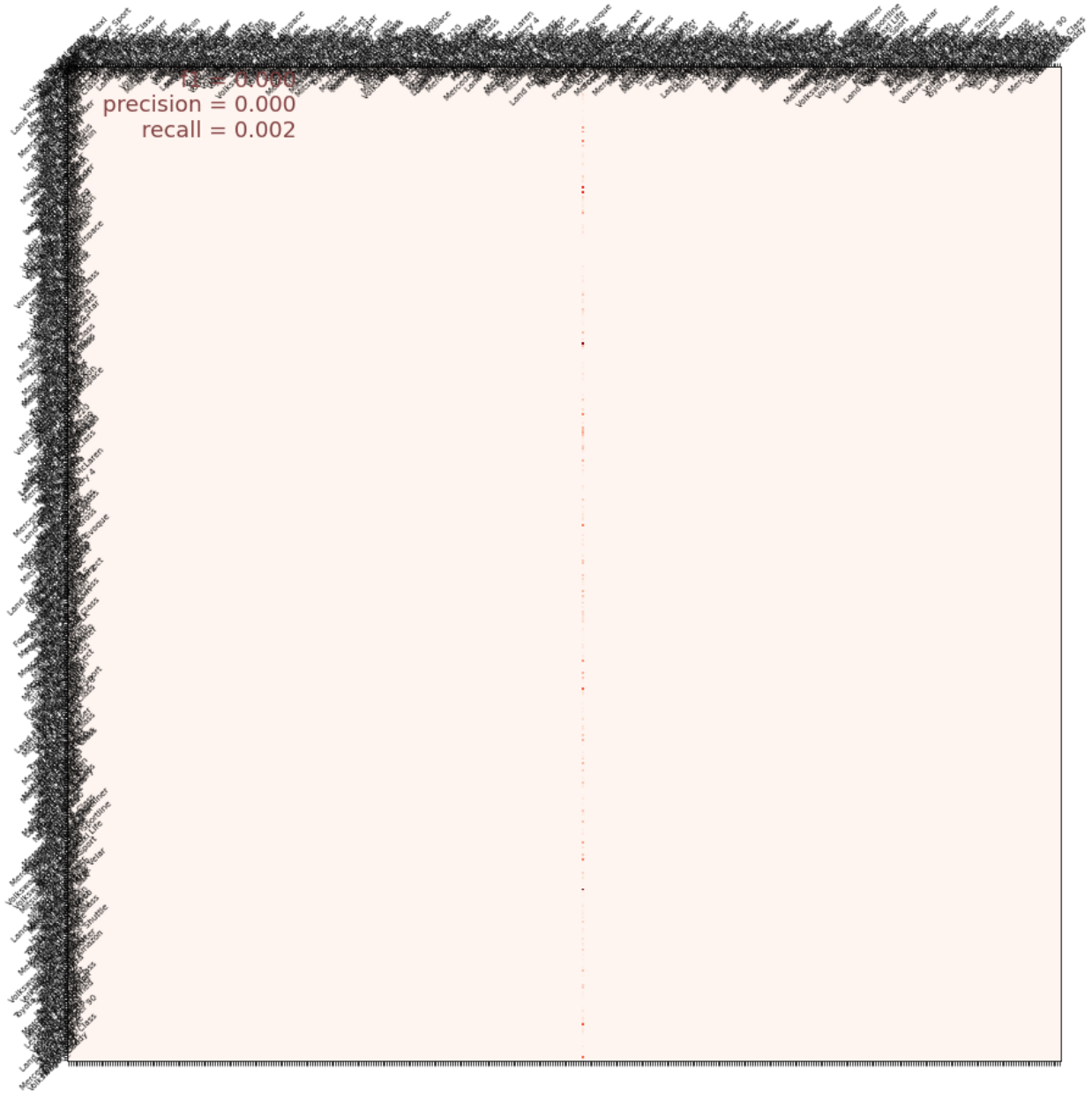
Histogram of IOU Scores

There is an obvious spike at zero. This can be explained because, as mentioned earlier, the labeled data does not recognize distant and obscured cars but the YOLO model does. So, in the many cases where the model predicts the presence of cars but none are labeled, the IOU score is zero. Beyond that, the majority of the rest of the scores are above .60. There is also a bump around 0.2 which requires some further investigation.

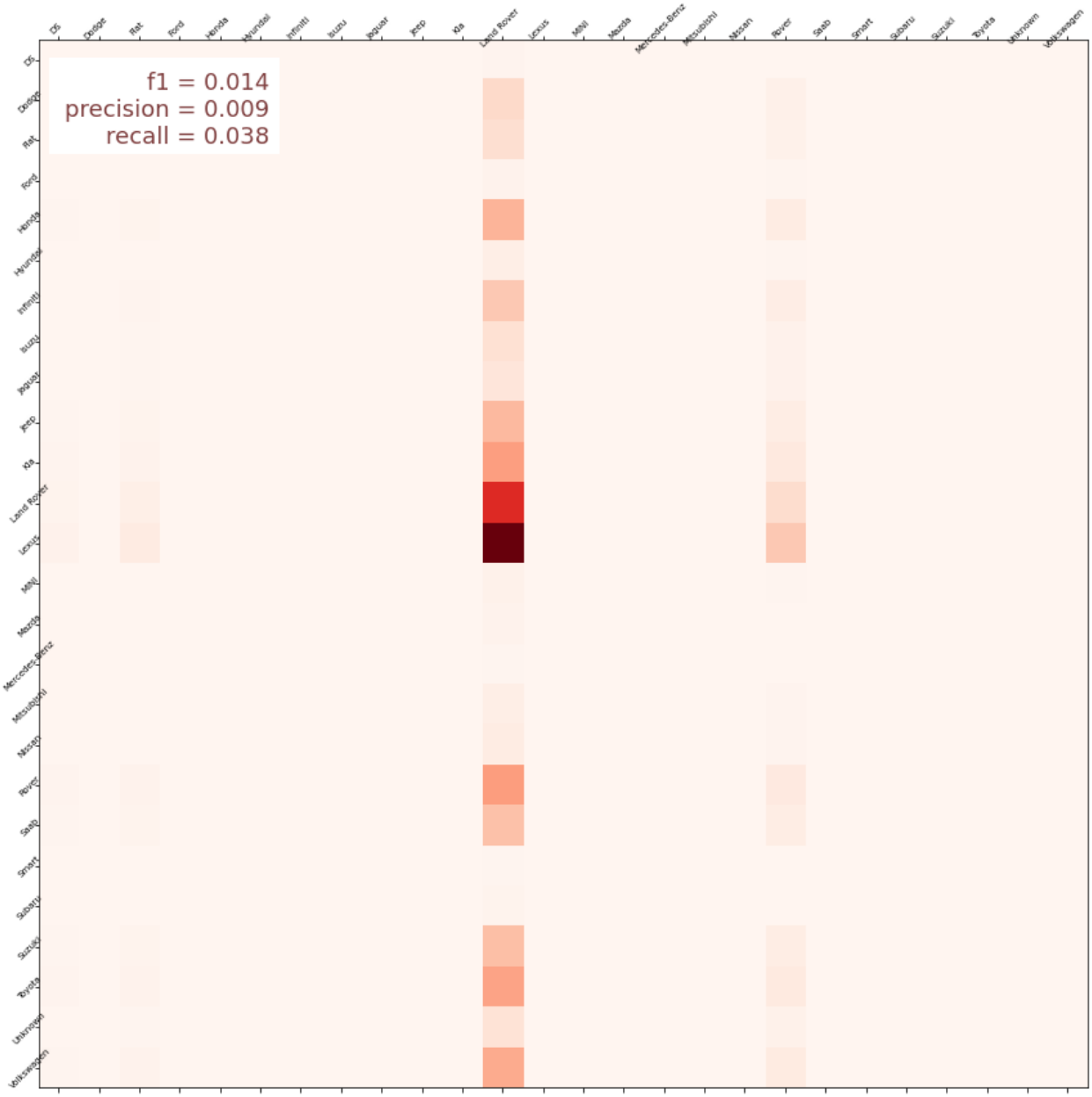
Overall, the YOLO model does seem to detect cars well. However, how well it bounds the detected car is somewhat unclear. One item to research would be the anchor boxes used in the YOLO model versus the length-width ratios used in the labeled data. The spread of the IOU scores may be explained because the YOLO model only uses a few anchor boxes with fixed ratios and the labeled data does not.

Classification

The car classification model used was the InceptionNet v3 architecture with weights pre-trained on the ImageNet dataset. The head was modified and trained on the DVM-Car data set. The first model was trained on the data set that included make and model. After 30 epochs, the validation accuracy reached 0.2%. This is worse than just predicting all cars as a Ford Focus (which was nearly 4% of the data). The precision, recall and F1 scores can be seen below on the confusion matrix heatmap. From the vertical line that can be seen on the heat map, the model did in fact simply choose Ford Focus for most of its predictions.



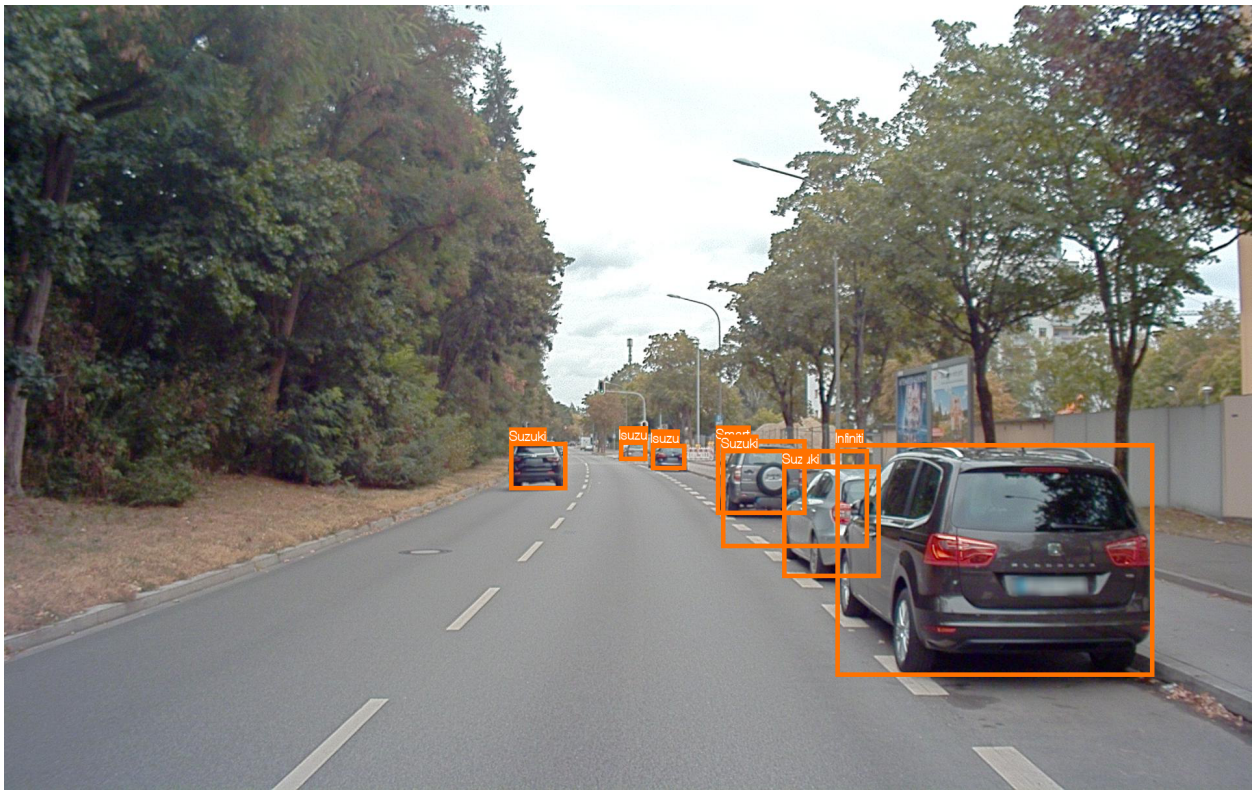
The result was disappointing and certainly due to the skewed data. In the hopes of getting some better performance, the data set with reduced classes was then used to train the model. With only 26 classes, the model should have more information per class to build on. After training, the validation accuracy reached 23%. This seems a better result but because the data was still skewed, it might indicate that the model just lumps everything into a few classes as before. The precision, recall and F1 scores can be seen below on the confusion matrix heatmap.



As expected, the model does in fact only classify into limited categories; Land Rover and Rover. In further study, more data to balance the classes can be added, image augmentation utilized, and different architectures experimented with to improve performance.

Putting the Models Together

Now both models can be used in sequence. A custom function was written that takes the detection model, the classifier model, and the image. The image passes through the detection model and the predictions are found. The predictions are in the form of coordinates in the image that correspond to the top left and bottom right of each bounding box of a suspected car. Those bounding boxes are then cropped from the image, padded to make square, resized, and fed into the classifier. Finally, the original image is returned with each predicted bounding box and class superimposed.



Example of Final Prediction

Recommendation to the Client

The system as is, can very effectively detect cars in the field of view of the camera image. However, with the best classification precision of only 0.9%, it is advised that the model should only be used to count cars in an image and not classify them.

More time should be allotted to advance the performance of the classification phase of the model.

Conclusions and Future Work

The final model does detect and classify the cars in the image.. Unfortunately, the classification phase only has a precision of 0.9%. The precision metric is nowhere near the 95% acceptability. However, the final function merely takes whatever classification model you give it. This means that focused effort on improving the classification model's performance can be made and the new version simply applied.

Other future aspects that can be improved and added upon are:

- Altering the function so that the class the YOLO model is searching for can be changed
- Optimizing the model to work in real time
- Training other classification models to use
- Packaging the entire project in a user friendly application