Summary of the problem statement

The objective of this project is to build a car classification and detection model with GUI. When an image is input through the GUI, the model would draw a bounding box and provide information about the make and the model of the car. The dataset used to train the model is Stanford Car Dataset. It contains multiple attributes about the car that can be used to classify it.

This application be used to identify cars from images that has application across different fields like -

1. Crime investigation - When they have to identify a car from a video feed, they could use this model. That way they can get the make and model of the car quickly.
2. Used car portals - Use case - When users upload the picture of their car to sell, the UI Model could complete the other fields automatically for them. Also, this can be used to provide recommendations for similar cars for the users.
3. Journalism - Can be used to identify and describe a photo. Like in the caption, it could be detailed, user passing before a car.
4. Car Counting - Could be used by the manufacturers for market study, counting cars of a particular type that are running in an area. This could be used by the authorities as well. Since they might be interested in understanding the age of the cars that is currently plying the road.
5. Car Services - Could be used to provide after sales service.
6. Car Garage - Car Garage door manufactures could implement this model, to understand the type of the cars and track it centrally.

Summary of the Approach to EDA and Pre-processing

Following are the steps undertaken to pre-process the data and to perform EDA including the Visualization:

1. Exploring the data set / folder structure to understand how to parse the information.
2. Parsing the bounding box information provided.
3. Balance of the dataset (split between test and train)
4. Different labels available
5. Resolution / metadata of the images

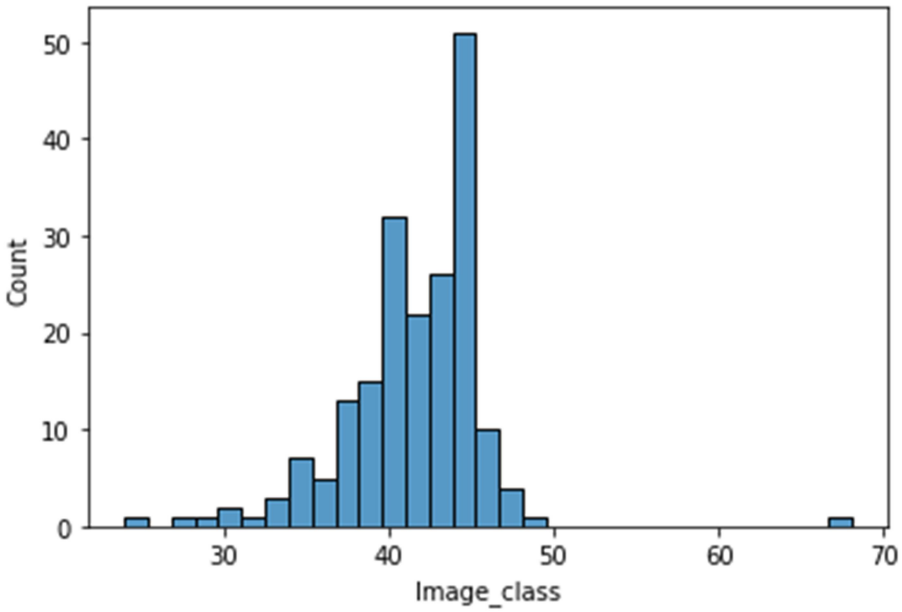
6. Detecting multiple cars.
7. Detecting anomaly and outliers in the dataset.
8. Detecting black and white images. If the no. of images are small they can be ignored.
9. Display multiple photos randomly selected from train and test folders to help in eyeballing.
    i. Eyeballing to see if photos of the cars cover all the sides.
    ii. Check if all angles are covered - from top / bottom, etc.

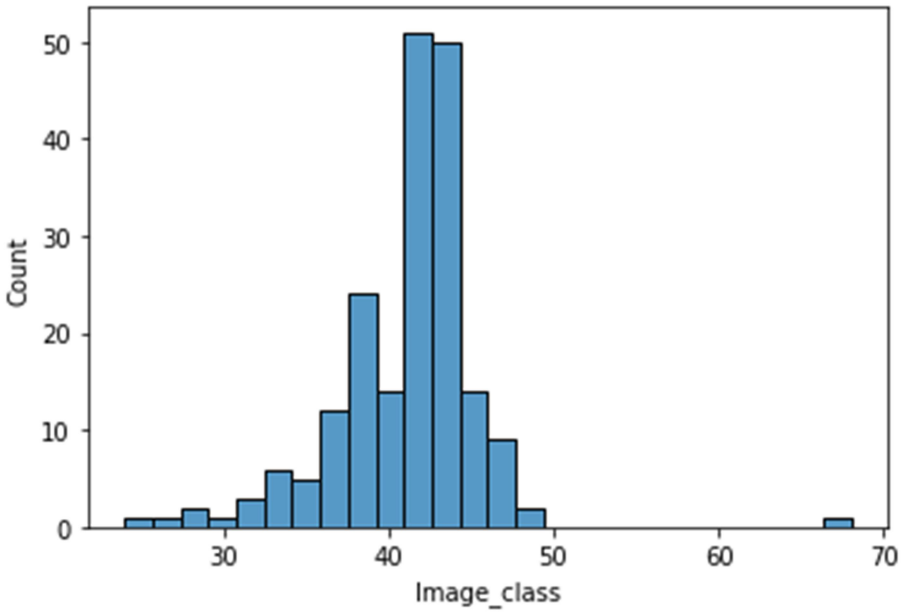
No. of classes

The dataset consists of 196 classes of cars.

Number of classes in Train and Test Dataset

In the train dataset, we have 8144 pictures of cars and in the test dataset, we have 8041 pictures of cars.





Eyeballing the cars

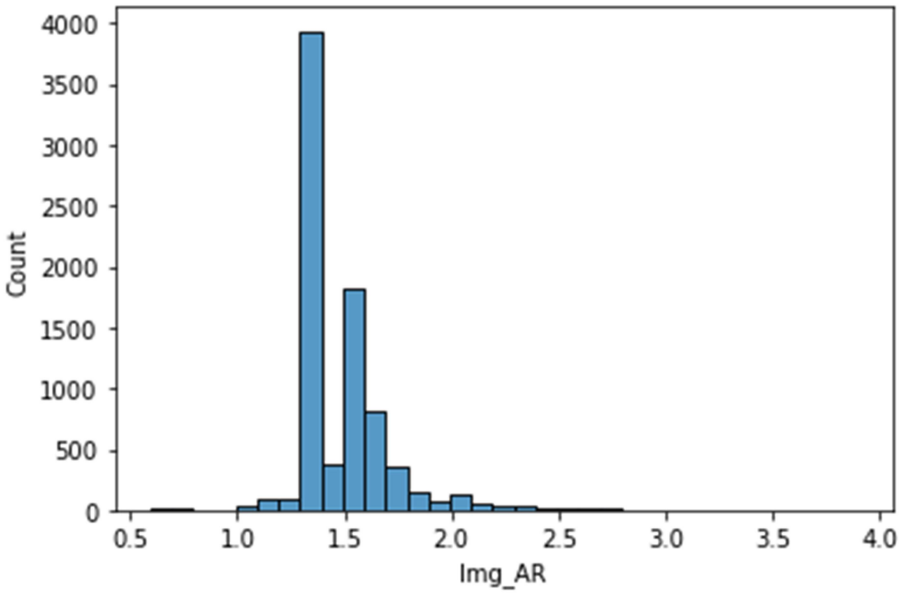
Train Dataset of images with the bounding boxes

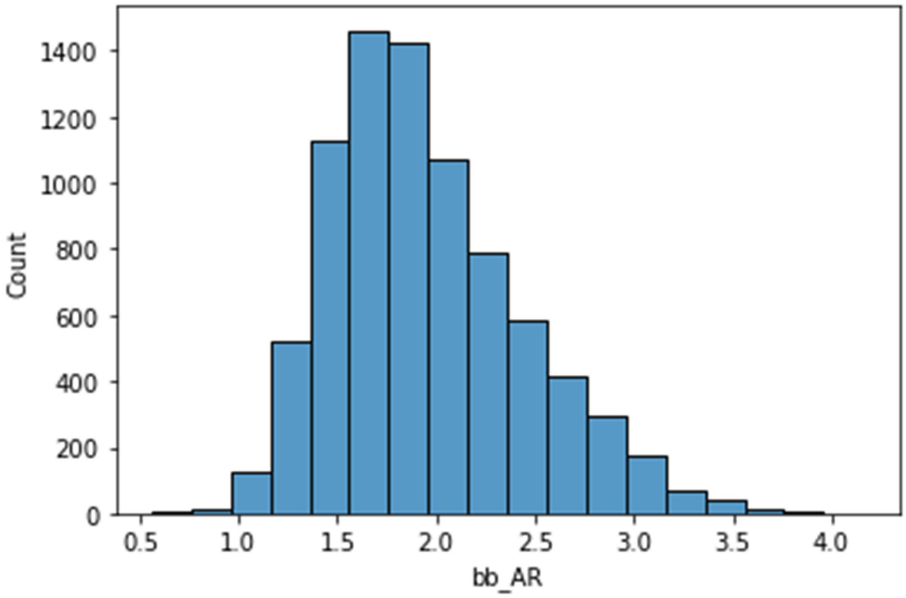

Acura Integra Type R 2001
Land Rover Range Rover SUV 2012
Dodge Dakota Crew Cab 2010
Toyota Sequoia SUV 2012
Chevrolet Silverado 1500 Regular Cab 2012
GMC Acadia SUV 2012
Jeep Liberty SUV 2012
Chevrolet Cobalt SS 2010
Toyota Corolla Sedan 2012
Spyker C8 Convertible 2009
Ford Focus Sedan 2007
Volvo 240 Sedan 1993
BMW 1 Series Convertible 2012
Scion xD Hatchback 2012
Chevrolet Sonic Sedan 2012
McLaren MP4-12C Coupe 2012
Jaguar XK XKR 2012
BMW 3 Series Wagon 2012
AM General Hummer SUV 2000
Bugatti Veyron 16.4 Convertible 2009
Lamborghini Gallardo LP570-4 Superleggera 2012
Hyundai Veracruz SUV 2012
Dodge Dakota Crew Cab 2010
BMW X3 SUV 2012
Lincoln Town Car Sedan 2011
Audi TTS Coupe 2012
Acura Integra Type R 2001
GMC Canyon Extended Cab 2012
Chevrolet Silverado 1500 Hybrid Crew Cab 2012
Dodge Durango SUV 2012
Chevrolet Silverado 2500HD Regular Cab 2012
Ford GT Coupe 2006
Chevrolet Express Cargo Van 2007
Bugatti Veyron 16.4 Convertible 2009
Chevrolet Trailblazer SS 2009
Infiniti QX56 SUV 2011
Ford Freestar Minivan 2007
GT Coupe 2006
Aston Martin Virage Coupe 2012
Dodge Dakota Crew Cab 2010
Geo Metro Convertible 1993
Buick Enclave SUV 2012
Toyota Sequoia SUV 2012
BMW 1 Series Convertible 2012
GMC Acadia SUV 2012
Chevrolet Avalanche Crew Cab 2012
Nissan 240SX Coupe 1998
Ram Pickup 3500 Crew Cab 2010
Lamborghini Reventon Coupe 2008
Chrysler 300 SRT-8 2010
Ford Ranger SuperCab 2011
BMW 3 Series Sedan 2012
Audi A5 Coupe 2012
GMC Savana Van 2012
Bentley Continental Flying Spur Sedan 2007
GMC Terrain SUV 2012
Mitsubishi Lancer Sedan 2012
Ferrari California Convertible 2012
Chrysler Crossfire Convertible 2008
Lamborghini Diablo Coupe 2001
GMC Yukon Hybrid SUV 2012
Mercedes-Benz 300-Class Convertible 1993
Rolls-Royce Phantom Sedan 2012
Jeep Liberty SUV 2012
Jaguar XK XKR 2012
Plymouth Neon Coupe 1999
Audi 100 Convertible 1994
Hyundai Sonata Sedan 2012
GMC Yukon Hybrid SUV 2012
Spyker C8 Coupe 2009
Audi 100 Sedan 1994
Chevrolet TrailBlazer SS 2009
Chevrolet Aspen SUV 2009
Audi V8 Sedan 1994
BMW 3 Series Sedan 2012
Audi TT RS Coupe 2012
Chevrolet Traverse SUV 2012
Chevrolet Malibu Sedan 2007
Chevrolet Tahoe Hybrid SUV 2012
Audi S5 Convertible 2012
Ferrari California Convertible 2012
Mercedes-Benz 300-Class Convertible 1993
Ferrari FF Coupe 2012
BMW X3 SUV 2012
Audi S5 Convertible 2012
HUMMER H3T Crew Cab 2010
Suzuki Kizashi Sedan 2012
Audi R8 Coupe 2012
Lincoln Town Car Sedan 2011

Removal of outliers and anomaly

We looked into the aspect ratio of the images and their bounding boxes. The images and bounding boxes with extreme aspect ratios could be outliers or an anomaly. Following are the histogram for Aspect Ratio(AR)

Aspect ratio of Images and bounding boxes



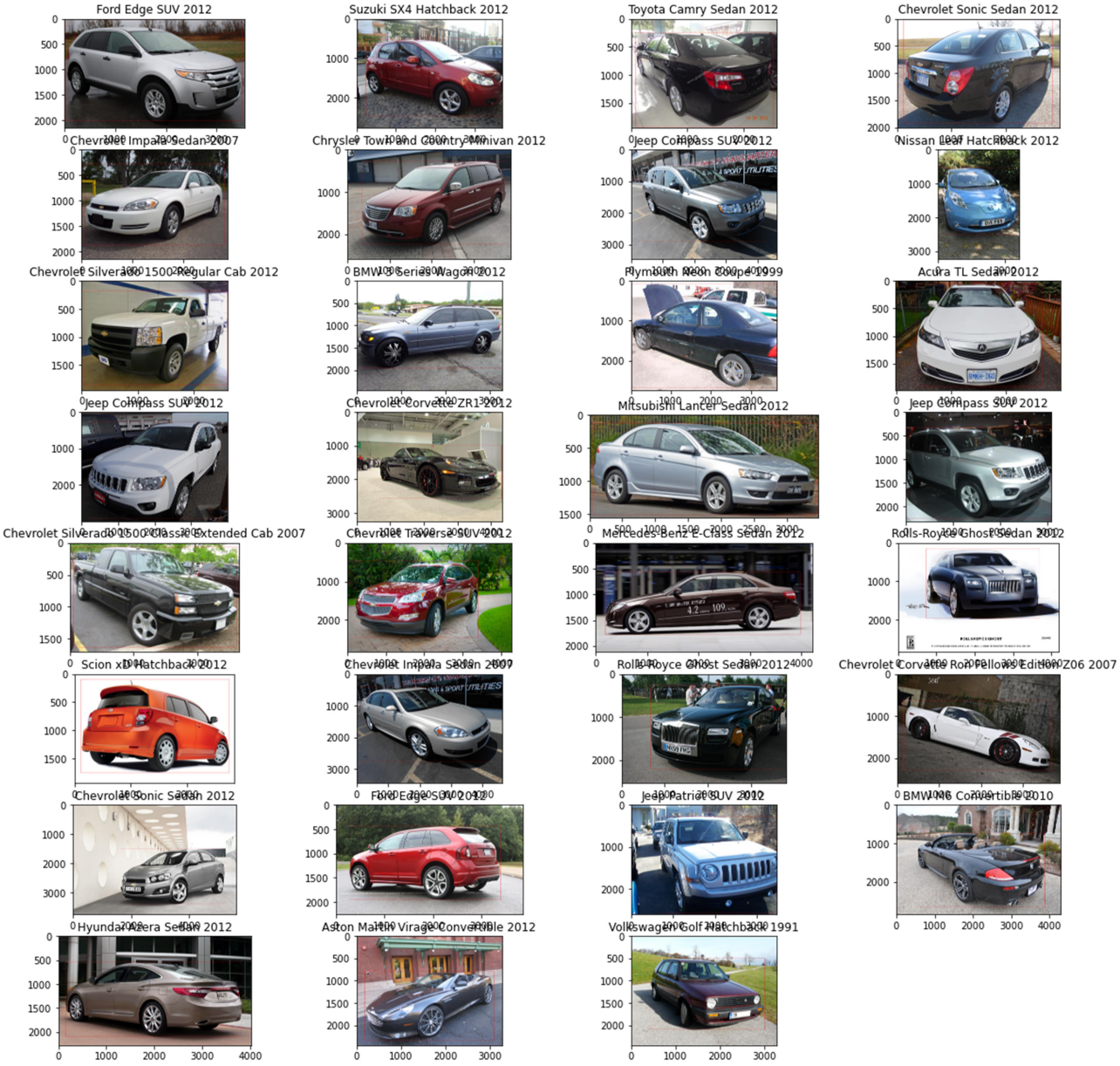


Based on the above diagrams we could infer that there there are few outliers in terms of the aspect ratio of bounding boxes. This could be because of the shape of cars. The sedans and sports cars would have a very different aspect ratio to cars like Minivans, and others. It also could be that the angle of the photographs would cover the entire picture. So the outlier aspect ratios are looked into.

Images with high aspect ratio


Chevrolet Corvette Ron Fellows Edition Z06 2007
GMC Savana Van 2012
AM General Hummer SUV 2000
Dodge Sprinter Cargo Van 2009
Geo Metro Convertible 1993
Chevrolet Corvette Ron Fellows Edition Z06 2007
Audi 100 Wagon 1994
Geo Metro Convertible 1993
Audi 100 Wagon 1994
Geo Metro Convertible 1993
Chevrolet Tahoe Hybrid SUV 2012
Geo Metro Convertible 1993
AM General Hummer SUV 2000
AM General Hummer SUV 2000
AM General Hummer SUV 2000
Chevrolet Corvette Ron Fellows Edition Z06 2007
Audi 100 Wagon 1994
Geo Metro Convertible 1993
Geo Metro Convertible 1993
Chevrolet Corvette Ron Fellows Edition Z06 2007
Geo Metro Convertible 1993

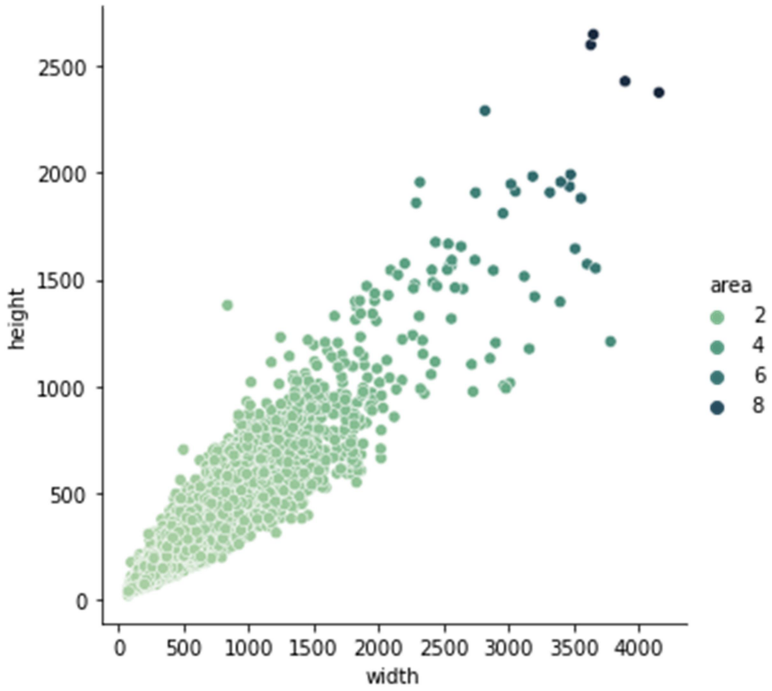
Images with low aspect ratio



From the above we could see that though these are outliers, they are not wrongly classified. Just that the images are different and different angles. We have chosen to include them, since there is a chance we could detect similar objects during testing.

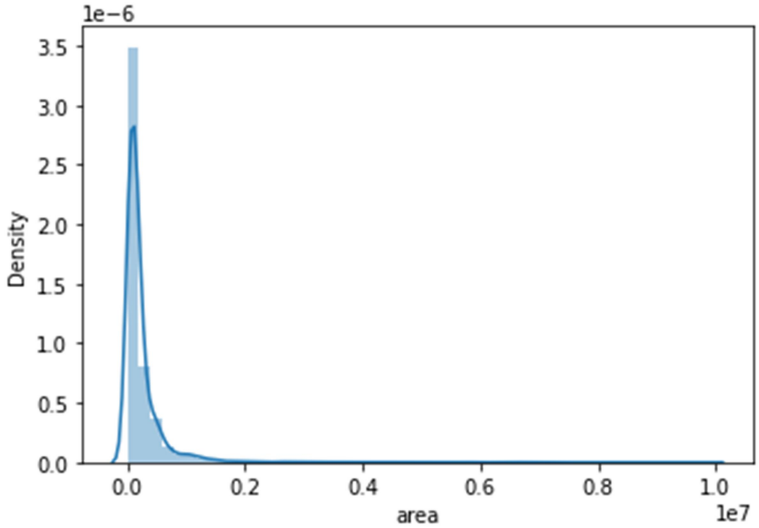
Scatter plot for h and w

Scatter plot for h and w would provide us a way to see if there are



outliers.

Histogram of area of the bounding box



Histogram for area.

Inference from EDA

Following are the inference from EDA:

1. We have class imbalance, need to do data augmentation of minority classes.
2. We have both black and white and colored images.
3. We have images with different aspect ratios. Most of the images lie 1.3 and 2.0. when we resize the image for model training, we need to take care of maintaining the high aspect ratio.
4. The aspect ratios of the bounding boxes fall in the range of 1.0 to 3.5. The reason for the variance of this bounding box could be because of the nature of the cars. For ex. Sedans and sports cars would have a higher width to height ratio compared to hatchbacks.

Deciding Models and Model Building

Two major factors to evaluate an object detection algorithm are inference speed and accuracy.

Based on the problem statement and above factors, the following algorithms can be used:

1. CNN Based custom model for predicting car class and bounding box.
2. SSD Mobilenet v1 based custom detection model
3. Faster - RCNN Based Model
4. YOLO Based Model

Design of Experiments

Experiment 1 -

Convert every image into black and white images. We think changing the image to grey scale would not affect much for this dataset. Since it is about the car and the structure is important.

Experiment 2 -

Ignore all the black and white image. We will compare the results we recieved on grayscale images with that of the colored images at a later point of time.

Experiment 3 – Data Augmentation - using CV2.

Horizontal Flip. Shifting position of image, left and right. Initially we would augment the data with horizontal flip only. At a later point of time, we would also include augmenting the data with shifting object position in image, random cropping the image, changing image contrast, changing image to blur etc. If we are not getting desired results, we would go to the next step of rotating the images, and recalculating the bounding boxes.

Experiment 4 - Class Weights

In the given dataset, we have got high number of classes - 196. Assigning weights to all the classes and then training the model might introduce errors and so this approach will be undertaken only if the data augmentation and pre-processing does not give satisfactory results.

Experiment 5 - Multiple Models - CNN, SSD, RCNN and YOLO V5

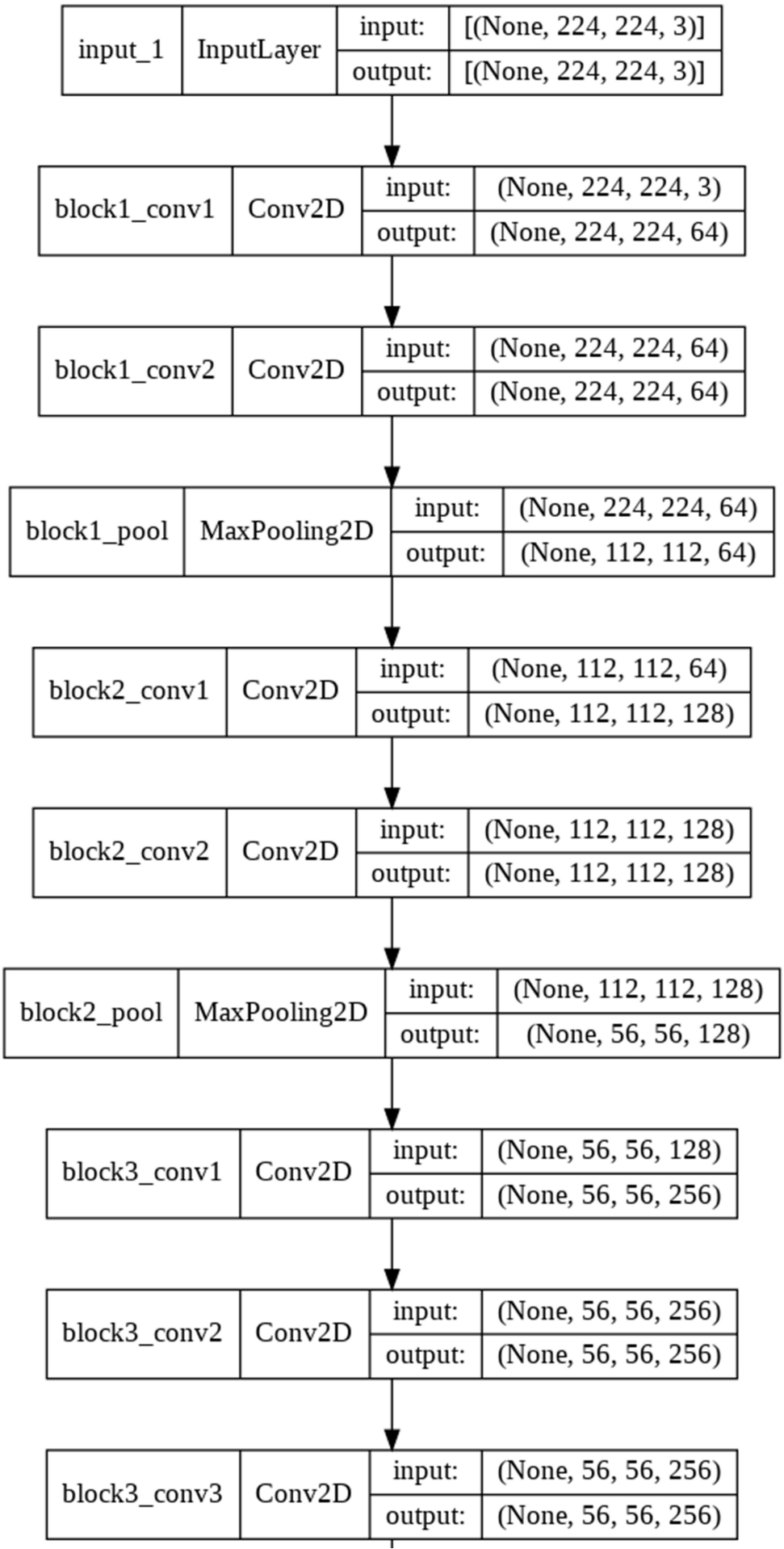
Finally, we will train multiple models with the curated dataset created in previous steps:

Custom CNN

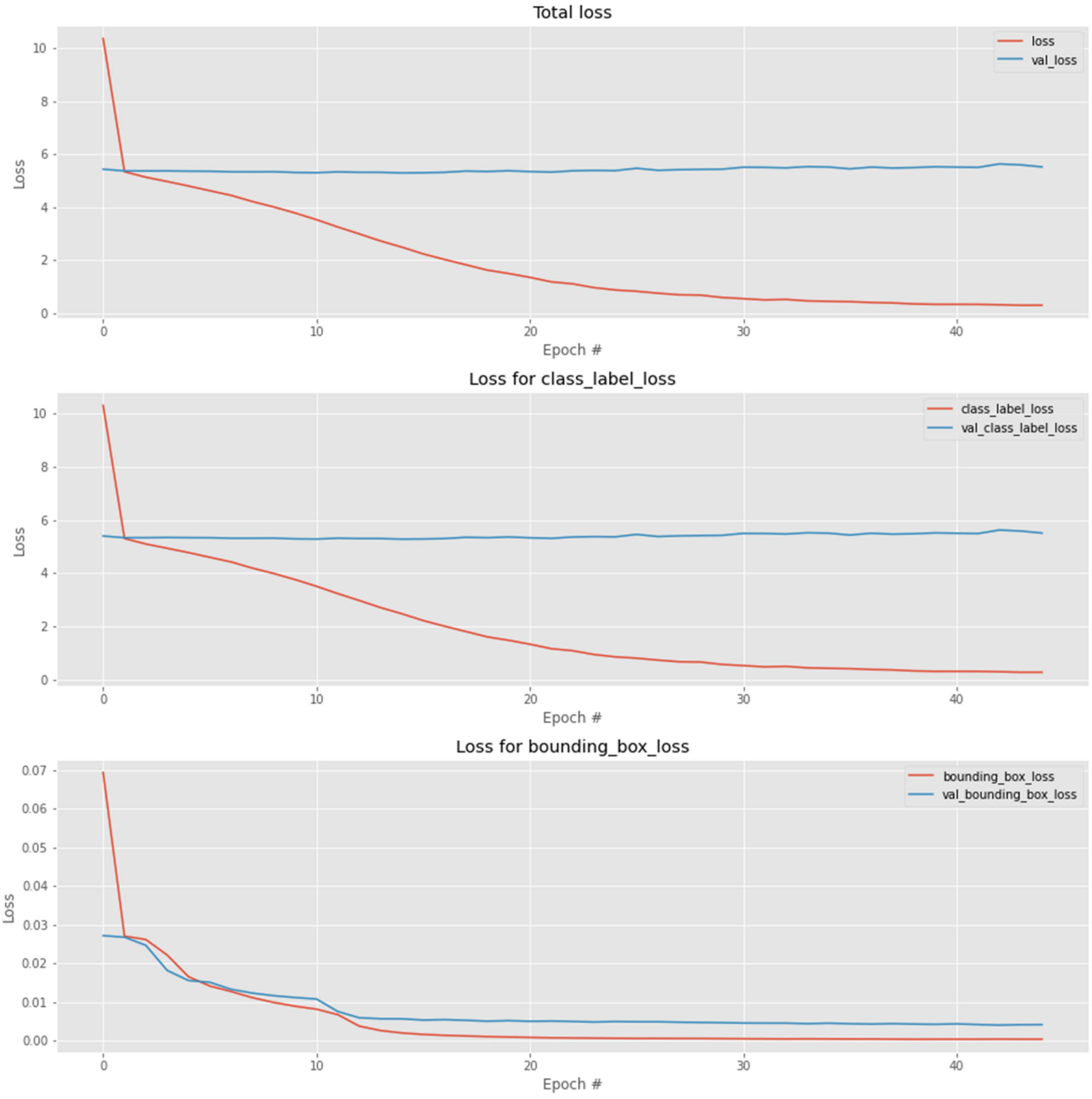
We have used a custom CNN object detection model. This is a vanilla model. We have used this as a baseline model for further experiments. In this model, we have added two output sub networks from VGG output - one classification network for predicting the class and one regression network for predicting bounding box.

We will compare the results with the state of the art models like SSD, YOLO and Faster RCNN.

VGG-16 is used as backbone model for this experiment.


input_1
InputLayer
input: [(None, 224, 224, 3)]
output: [(None, 224, 224, 3)]
block1_conv1
Conv2D
input: (None, 224, 224, 3)
output: (None, 224, 224, 64)
block1_conv2
Conv2D
input: (None, 224, 224, 64)
output: (None, 224, 224, 64)
block1_pool
MaxPooling2D
input: (None, 224, 224, 64)
output: (None, 112, 112, 64)
block2_conv1
Conv2D
input: (None, 112, 112, 64)
output: (None, 112, 112, 128)
block2_conv2
Conv2D
input: (None, 112, 112, 128)
output: (None, 112, 112, 128)
block2_pool
MaxPooling2D
input: (None, 112, 112, 128)
output: (None, 56, 56, 128)
block3_conv1
Conv2D
input: (None, 56, 56, 128)
output: (None, 56, 56, 256)
block3_conv2
Conv2D
input: (None, 56, 56, 256)
output: (None, 56, 56, 256)
block3_conv3
Conv2D
input: (None, 56, 56, 256)
output: (None, 56, 56, 256)

Performance of the custom CNN model



SSD Mobilenet v1 coco based car detection model

SSD (Single Shot MultiBox Detector) is a popular algorithm in object detection. It's generally faster than Faster RCNN. The SSD architecture is a single convolution network that learns to predict bounding box locations and classify these locations in one pass. Hence, SSD can be trained end-to-end. The SSD network consists of base architecture (MobileNet in this case) followed by several convolution layers:

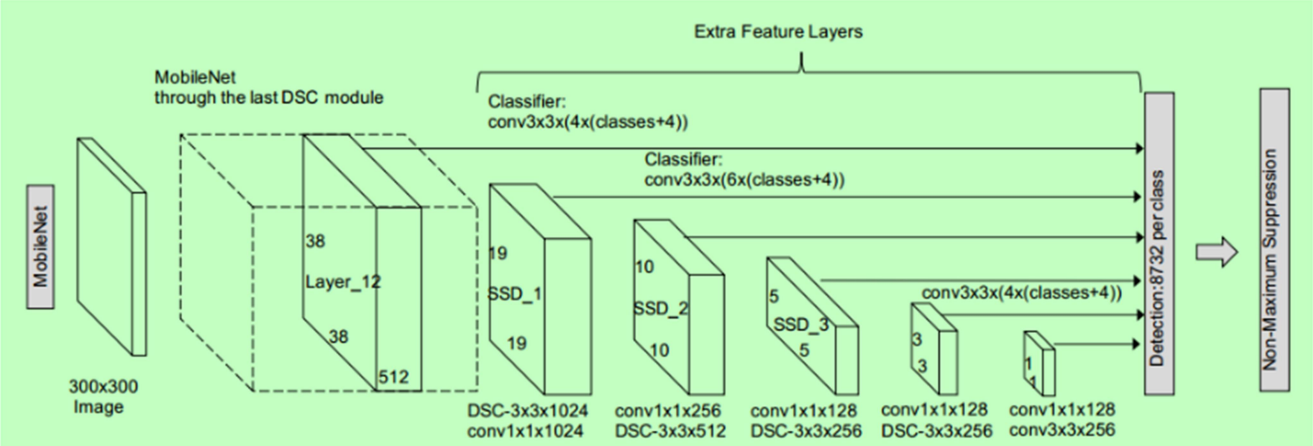


Figure 6. Mobile-Det: SSD-based detection with MobileNet as backbone, modified based on [14]

By using SSD, we only need to take one single shot to detect multiple objects within the image, while regional proposal network (RPN) based approaches such as R-CNN series that need two shots, one for generating region proposals, one for detecting the object of each proposal. Thus, SSD is much faster compared with two-shot RPN-based approaches.

We are using tensorflow object detection API for this model training.

YOLO

YOLO is a state of art algorithm for object detection. This is the fastest and accurate object detection algorithm. This is because it is oneshot learning algorithm. It skips the region proposals layer.

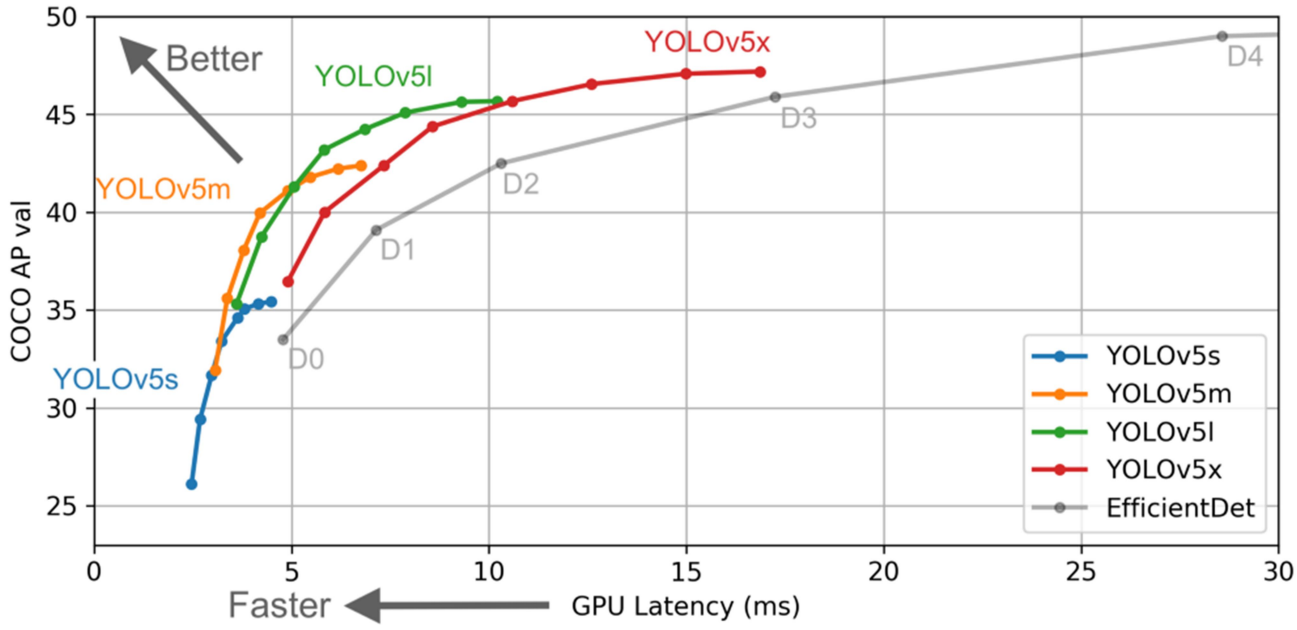
Reasons for Choosing YOLO V5

YOLO is a state of art algorithm for object detection. This is the fastest and accurate object detection algorithm. This is because it is oneshot learning algorithm. It skips the region proposals layer.

For this project we need an algorithm to work in real time, rather than batch processing, hence speed is significant.

The biggest draw back of YOLO is that it cannot detect small and very small images. However, in the dataset we have cars which occupy more than one grid of YOLO. Hence, this drawback is not significant with YOLO.

We are using a transfer learning based approach for YOLO. We use the pretrained model



The Method

1. Prepare the dataset to be used for YOLO V5.
    i. Creating the dataset.yaml
    ii. Creating the Annotations - the bounding box and the labels
2. Organize Directories
    i. Create /coco128 in the /datasets directory.
    ii. Place the labels and images as prescribed by yolov5.
3. Choosing the Model
    i. YOLOv5s - the smallest to provide the quick inference times, but the accuracy is smaller
    ii. YOLOv5x is the biggest with highest accuracy, but slower inference time.
    iii. YOLOv5m YOLOv5l in-between.
4. Training and Logging the different tests.

Reasons for choosing Faster R-CNN

Following are the major reasons for choosing Faster R-CNN over RCNN:

1. R-CNN would need to classify 2000 region proposals.
2. R-CNN takes around a minute for each test image, hence cannot be used for real time.
3. Learning even at the RoI pooling layer.

How to improve your model performance

The model performance could be improved with the following:

1. Data Augmentation - different techniques like cropping, padding, and horizontal flipping, which are commonly used to train large neural networks
2. Moving it to cloud - By moving the CNN to cloud using something like databricks architecture, we could do wonders.
3. Deploying an cloud based piplelines that allow us to quickly experiment with different CNNs.
4. For YOLO, we could use different models like V5n, V5s, V5m, V5l, V5x. We have to find the sweet spot between accuracy and inference speed between these models.
5. Hyper parameter Tuning.