**CAPSTONE PROJECT – COMPUTER VISION – CAR DETECTION**

| Author | Group 3 members |
| --- | --- |
| Mentor | Mr.Aamit |
| Date Created | 15/01/2023 |

Table of Contents

[**PROBLEM STATEMENT**](#_heading=h.1fob9te) **3**

[**1. ABSTRACT**](#_heading=h.8chge5tdezv4) **4**

[**2. Overview**](#_heading=h.uyb8m0pmi9pr) **5**

[**Problem Solving Methodology**](#_heading=h.w0y2vh9h1i4i) **5**

[**3. Walk through of the solution (EDA and PRE-PROCESSING)**](#_heading=h.s3w8c0uo6g4c) **5**

[Data Collection and Data importing](#_heading=h.2s8eyo1) 5

[Exploratory Data Analysis](#_heading=h.17dp8vu) 6

[Data Visualization](#_heading=h.9zh6cy22l43) 7

[Data Pre-processing](#_heading=h.26in1rg) 11

[MODEL BUILDING](#_heading=h.lnxbz9) **12**

[Model Building for Car detection:](#_heading=h.35nkun2) 12

[Training the model:](#_heading=h.1ksv4uv) 13

[Validating the model](#_heading=h.44sinio) 14

[Fine tuning the model](#_heading=h.y1xy8xd0uwnz) 15

[**Transfer Learning for car detection**](#_heading=h.m8jaohajs8fm) **16**

[**VGG19 model was built using following code**](#_heading=h.r3szrfuynb4p) **16**

[**ResNET50 model was built using following code**](#_heading=h.er1mn3rrh4ma) **17**

[**Mobilenet model was built using following code**](#_heading=h.nzdsmtbfzdso) **18**

[Model Building for predicting bounding box:](#_heading=h.sfs5rhqsnuvw) 18

[**Mobilenet**](#_heading=h.qjbgtvhdffv1) **19**

[**Yolo**](#_heading=h.3bu66qoptjt6) **19**

[**Saving the Model**](#_heading=h.kzm2aikyl3z0) **20**

[**4. Model Evaluation**](#_heading=h.yjtm8blfjswz) **20**

[**Object Detection**](#_heading=h.4wpolf1y824s) **21**

[**5. Comparison with benchmark**](#_heading=h.fp6ep0oc1dkm) **21**

[**6. Visualisations of Outputs**](#_heading=h.s0uengru2i1q) **21**

[**Car model prediction with mobilnet**](#_heading=h.x1magmq4qugq) **23**

[**MobileNet Car Model Prediction**](#_heading=h.a30ubwl1ovze) **24**

[**Bounding box detection using Mobilnet model**](#_heading=h.platerh48ukw) **24**

[**Bounding box detection using yolo model**](#_heading=h.n2umo7pqdda1) **25**

[**7. Implications**](#_heading=h.hc6pjt1b9hmm) **26**

[**8. Limitations**](#_heading=h.o8nxzkee8bvk) **26**

[**9. Conclusion**](#_heading=h.66y68ili80mx) **27**

# PROBLEM STATEMENT

DOMAIN

Automotive Surveillance.

CONTEXT

Computer vision can be used to automate supervision and generate appropriate action triggers if the event is predicted from the image of interest. For example a car moving on the road can easily be identified by a camera as the make of the car, type, colour and number plates etc.

# ABSTRACT

Monitoring traffic in the Automotive Surveillance System heavily demands recognition of vehicle make, model, type, colour etc.Identifying vehicle make and model is a challenging task due to intraclass variation (Acura RL Sedan 2012, Acura TSX Sedan 2012, etc), view-point variation, and different illumination conditions. In this project, we have experimented with the classification of vehicle images by artificial vision using Keras and TensorFlow to construct a deep neural network model, Python modules, as well as a machine-learning algorithm.

DATA

The Stanford Cars dataset developed by Stanford University AI Lab specifically to create various models for classifying the car type, model, make, etc. The dataset contains 16,185 images of 196 classes of cars. The data is split into 8,144 training images and 8,041 testing images, where each class has been split roughly in a 50-50 split. Classes are typically at the level of Make, Model, and Year, e.g. 2012 Tesla Model S or 2012 BMW M3 coupe.

The Stanford Car images can also be downloaded as zip file, which contains folders for Train images and Test images. Train/Test images folders contain 196 folders each. These 196 folders hold the descriptive name of the car (Acura Integra Type R 2001, Acura RL Sedan 2012, etc). By reading the file path, these descriptive car names (classes) can be obtained. Under each folder, respective model’s car images are given in .jpg format. The number of images in each folder may vary.

Separate annotation files are also provided for train and test images in CSV format. These annotation files contain car image names, bounding box coordinates, and the numerical class value assigned for the car.

Summary of Data description:

Train Images: Consists of real images of cars as per the make and year of the car

Test Images: Consists of real images of cars as per the make and year of the car

Train Annotation: Consists of bounding box region for training images

Test Annotation: Consists of bounding box region for testing images

Findings

Accuracies of Car model detection achieved by using Basic CNN were quite low (<5%). However improved accuracy (up to 60%) can be achieved by using transfer algorithms like resnet, vgg16, mobilenet etc. Bounding boxes prediction was also performed with a dice coefficient of up to 90%. Higher epochs with decreased batch size will improve the accuracy further.

# Overview

## Problem Solving Methodology

Following steps were used to solve the problem of car detection and predicting the bounding box:

1. EDA and Preprocessing:
   1. Data collection and Importing: Import the image and csv data to environment
   2. Exploratory Data Analysis: Understand the image data and csv data
   3. Data Visualization: Visualize the cars and bounding box. Also understand the distribution of data provided.
   4. Data Preprocessing: Convert the data into required format. Resize the image data to 224\*224
2. Object Classification Modelling:
   1. Basic CNN model: Build a basic CNN model and train on images. Checked the accuracy on test images
   2. CNN model with Transfer learning: To improve the accuracy of object detection transfer learning algorithms like vgg16, ResNet, mobilenet etc were used.
3. Prediction of Bounding Box:
   1. Build a model using MobileNet, yolo3 to predict the bounding box.
      1. MobileNet predicts the object with masked area
      2. Yolo3 with bounding box

# Walk through of the solution (EDA and PRE-PROCESSING)

## Data Collection and Data importing

Following is the list of data provided for this project:

* Set of training car images (8144 files) with folder name indicating the car model
* Set of test car images  (8041 files) with folder name indicating the car model
* Set of train and test CSV files containing car image name, class of each image and bounding box for the car location
* CSV file indicating all models of cars

Import the above data to project using below commands:

**For image data:**

****

**For CSV data:**

****

## Exploratory Data Analysis

Exploratory Data Analysis is required to understand the structure of the given data and identify the patterns & issues within data.

* First step is to understand the target variable i.e., predict the parameter. In this project, the target variable is the car model or car category and coordinates of the bounding box. Take each car image as feature data to predict the target variable of car model or car category.
* Further to understand the data in csv files one of the aspects is to find the missing values in the data using following commands:

train\_data.isna().sum()

* In the given csv file, no such missing data was present in both train and test data.
* To understand the various categories available in target variable (car category or model), following command can be used:

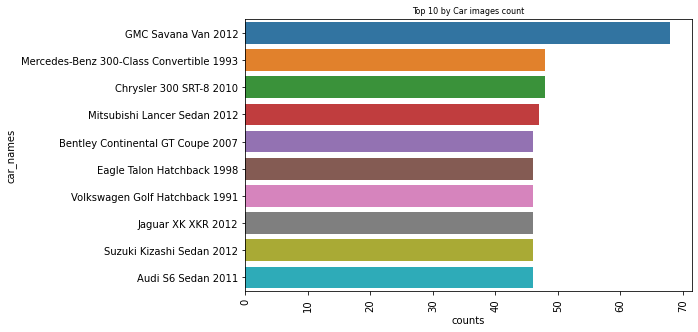
train\_data['Image class'].nunique()

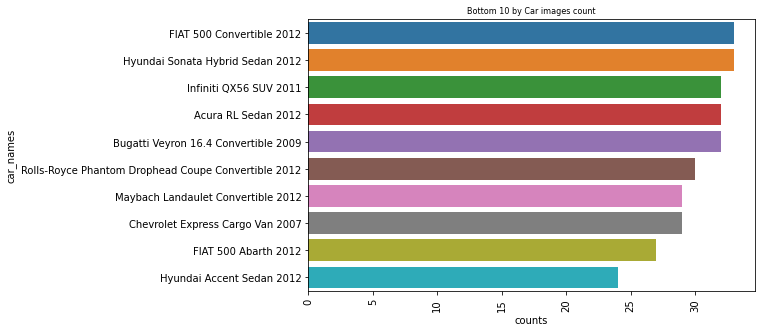
* Observed in this data that there are 196 different car categories or models.

## 

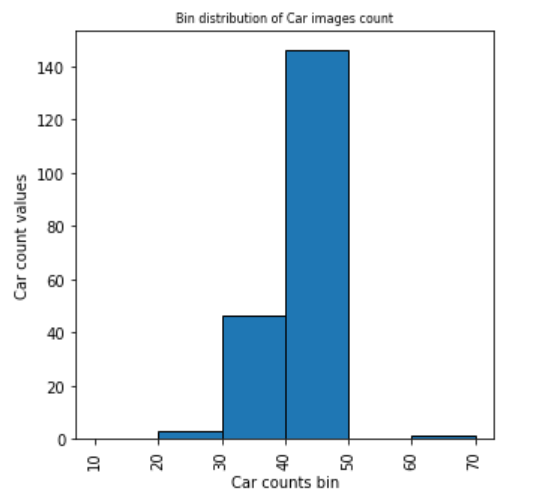
## Data Visualization

One of the ways to understand the data better is through data visualization techniques. Project has most of the data is images, following visualizations help to get better picture:

* Distribution of the number of images in car model can be depicted as follows. As the number of car models is high, the top and bottom number of models can be depicted.



The car image counts depicted in bin distribution.

* + 
* As we see above, for around 140 cars we have 40-50 images, for around 50 cars, we have 30-40 images. For a few cars, we have very few images i.e. 20-30. For just 1 car we have high number of images i.e. 60-70
* Let us check the distribution of resolution of the images. For this, let us divide the resolution into width and height components of the image. Plots of these width and height components separately can be done as follows:

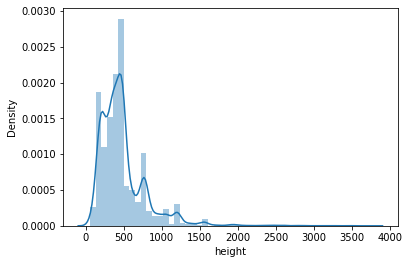


Figure1: Distribution of height of images

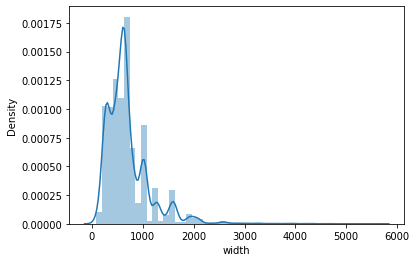
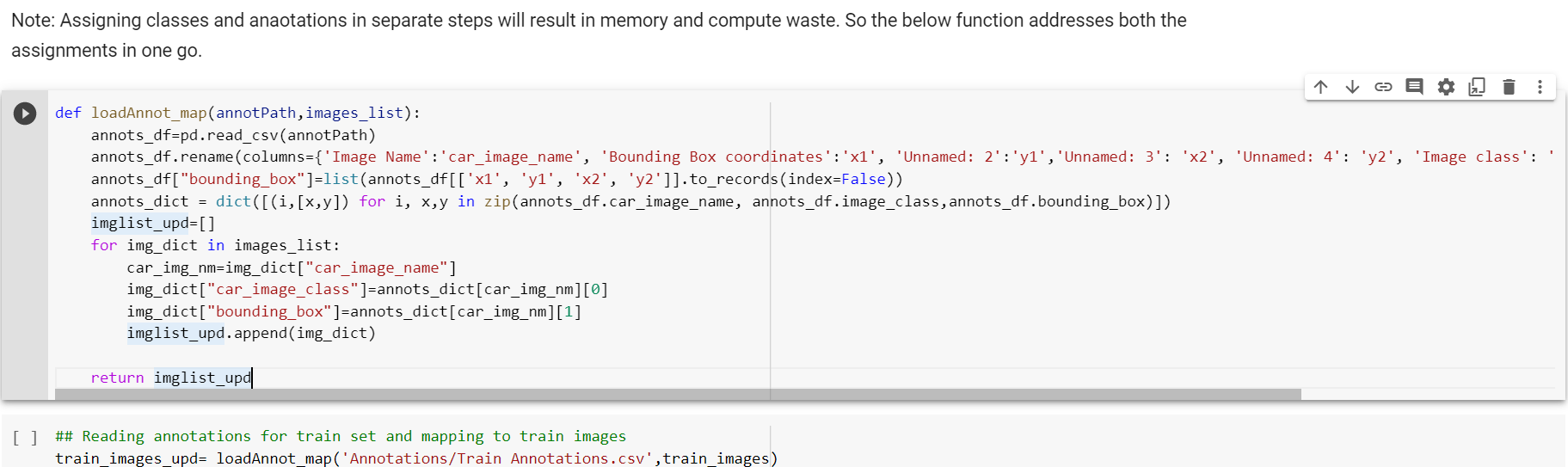
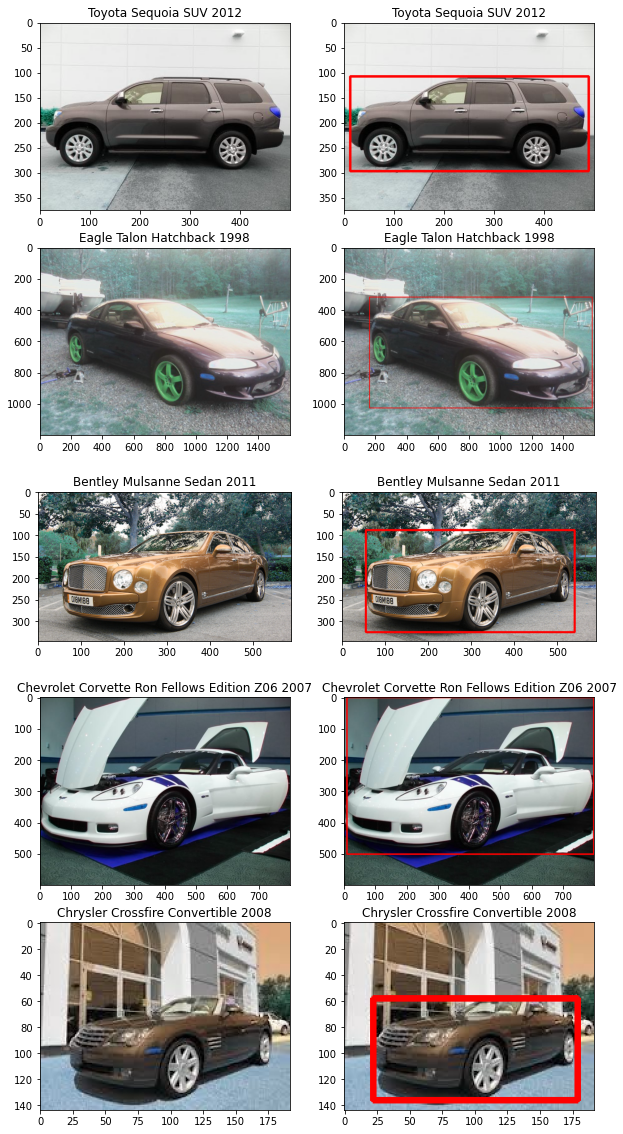


Figure2: Distribution of width of images

* From the above plots, it can be observed that most of the image data (median) has a height of 500 and width of 700. In addition, it can be observed that there is large variation in the image data. However, to train the CNN model, all the images shall be of the same resolution, as CNN cannot handle variation in resolution.
* Display of random images with bounding box and the code to show the same



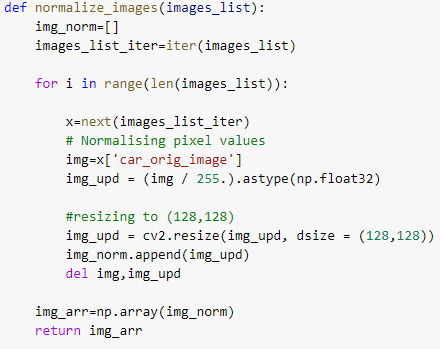


## Data Pre-processing

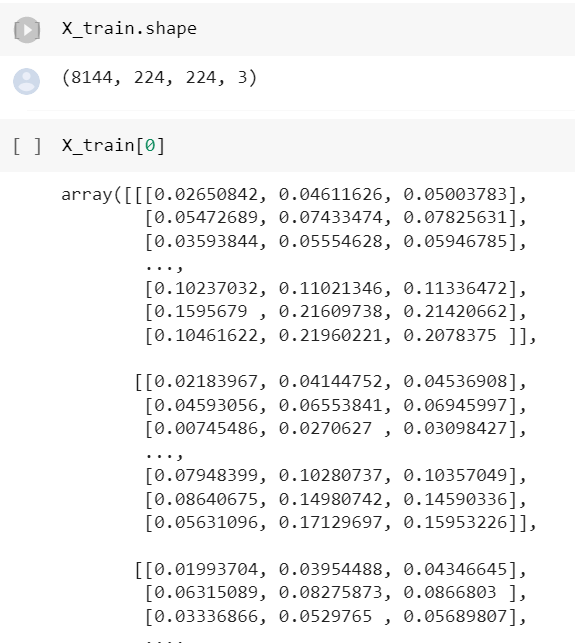
Data preprocessing is required to get the data in a better and useful format.

* As indicated above, resolution needs to be changed to a standard format so that all the images can be provided as input to CNN model. Also pixel data to be normalized. For this purpose, load the images and ‘resize ’& normalize both test and train data using following function:

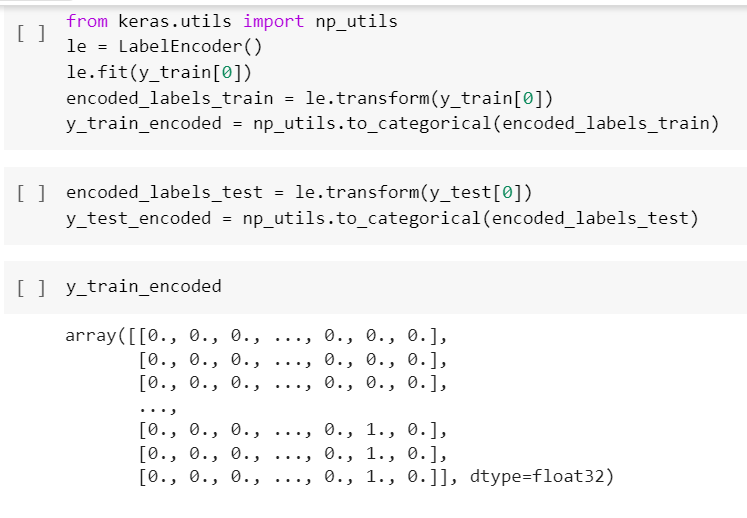




* The CV library converts the image into vector arrays as shown below both train and test data



* Next step is to one hot encode the class variable both train and test data



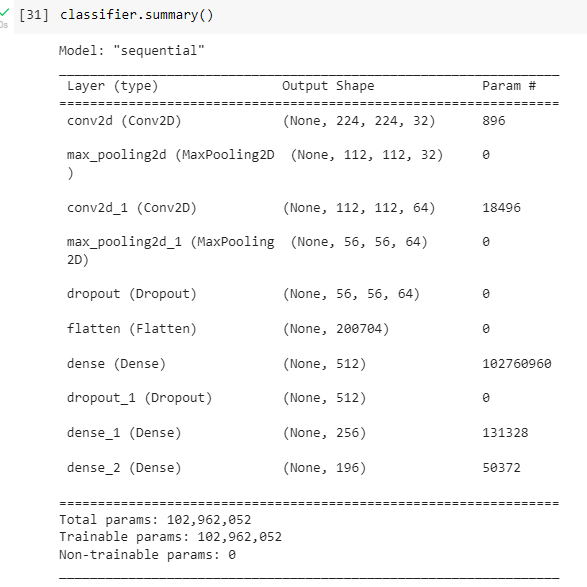
* Now the data is prepared to pass into the CNN model which is the next step

# MODEL BUILDING

## Model Building for Car detection:

Once the data is ready after all the pre-processing, it is time to build the basic CNN model for image recognition to classify the images into one of the 196 car models. There are various ways of machine learning algorithms like support vector machines, random forest etc and classic deep neural networks for classification problems. However Convolutional Neural Network works best for image recognition applications. One of the significant reasons for choosing CNN over other machine learning and deep neural networks is due to its capability to automatically extract image features.

Following based CNN model built using Tensorflow library:



Following are the features of the above model:

* No of hidden layers=6
* Max pooling and drop out layers to make training less overfitting
* Activation function for hidden layer is ’relu’ to introduce non linearity in model
* Activation function for the output layer is softmax as this is a classification problem with 196 classes.

## Training the model:

The above base CNN model trained using Adam optimizer, with 10 epochs and batch size of 32. Loss metrics were calculated using loss function categorical\_crossentropy. Accuracy metrics were also calculated for the train and test set.

opt **=** Adam()

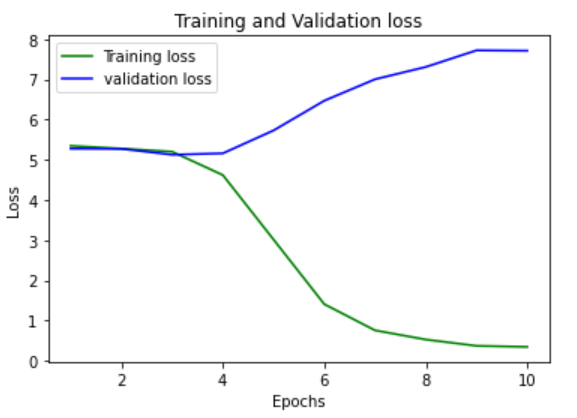
classifier**.**compile(optimizer **=** opt, loss **=** 'categorical\_crossentropy', metrics **=** ['accuracy'])

base\_model **=** classifier**.**fit(X\_train, y\_train\_encoded, validation\_data**=**(X\_test, y\_test\_encoded), epochs**=**10, batch\_size**=**32, verbose**=**1)

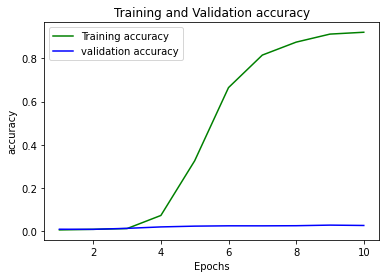
**The model has produced 92.12% accuracy in the train set and just 2.64% accuracy in the test set.**

## Validating the model

Below are the graphs showing the Train and Test (Validation loss).



Train loss increased after epoch 4 and was steady after epoch 8. Whereas Test loss started decreasing from epoch 3 and reached the lowest loss at epoch 10. The reverse in the trend/behaviour in train Vs test loss indicates the model is not predicting well in the test set.



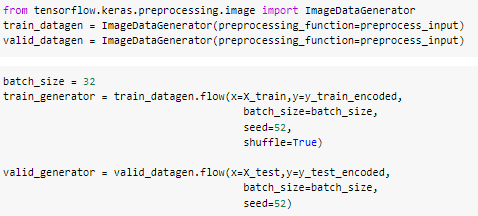
Train accuracy shows gradual increase starting from epoch 4 and reaching the maximum at epoch 10. It seems to be steady henceforth. Whereas the test accuracy was constantly around 1% or 2% throughout all the epochs.

From the above graph, it is evident that the model has overfitted in the training set, giving very low accuracy in the test set. This model needs further hyperparameter tuning. We may need to apply transfer learning/combine a few more models in order to improve accuracy.

## Fine tuning the model

Fine tuning of the base CNN model was performed by changing the number of epoch, batch size. image size etc. There was no significant improvement in the test accuracy.

Also image augmentation methods were used to generate new images from the original dataset to increase the variation in training of the base CNN model. Following is the code for generating the new images:



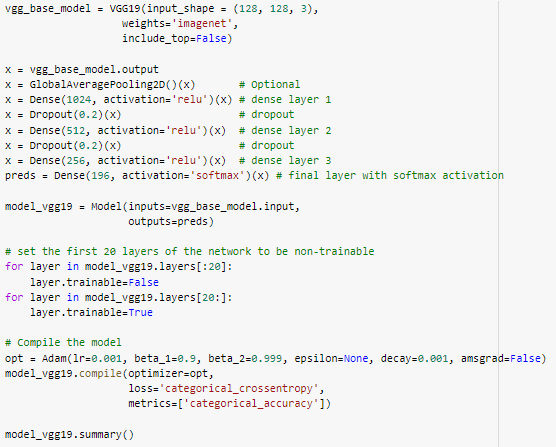
Base model was trained with these generated images also. However no considerable improvement in test accuracy was observed.

# Transfer Learning for car detection

Using various transfer learning methods, the object detection model was trained to improve the accuracy.

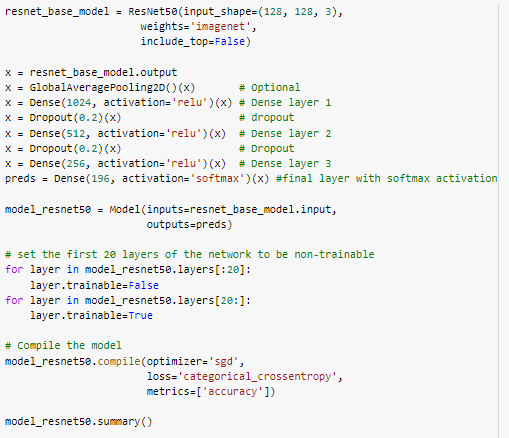
## VGG19 model was built using following code

Base VGG19 model with 3 hidden layers was built. Optimizer used was ‘adam’.



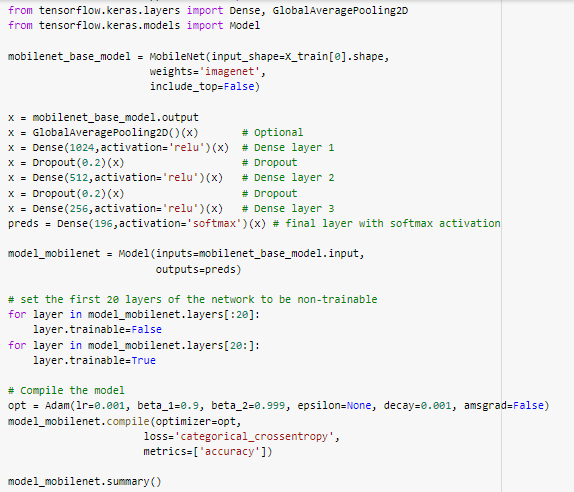
## ResNET50 model was built using following code

Base ResNET50 model with 3 dense layers was built. Optimizer used was ‘sgd’.



## Mobilenet model was built using following code

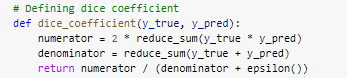
MobileNet model with 3 dense layers was built. Optimizer used was ‘adam’.



## 

## Model Building for predicting bounding box:

Dice coefficient is used to assess the performance predict bounding box. Following is the code to define dice coefficient:



Two models were used for prediction of the bounding box. Following are the models:

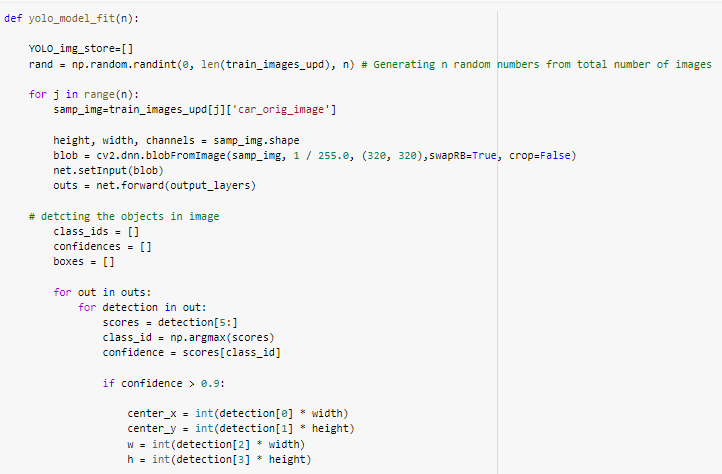
### Mobilenet

Following model was built using mobilnet base model with performance parameter as dice coefficient:



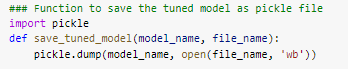
### Yolo

Yolo model was used to predict the bounding box of car models. Following is the code for the model

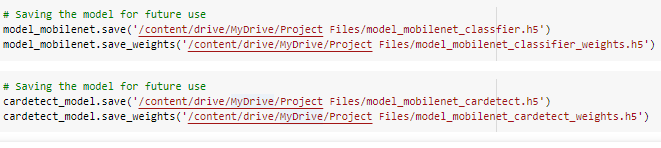


## Saving the Model

All the required models were saved to pickle for future use:



With the above function, both mobilenet and yolo for car detection and bounding box prediction were saved for future use.



# Model Evaluation

Car Model Classification

Models performance has been assessed using accuracy of prediction of test images.

Test and train accuracies are as tabulated below:

| **Sl.No.** | **Model** | **Train Accuracy** | **Test Accuracy** | **Test Loss** |
| --- | --- | --- | --- | --- |
| 1 | Basic CNN model | 97% | 3% | 8.6 |
| 2 | CNN model with image augmentation | 1% | 1% | 5.2 |
| 3 | vgg19 | 99% | 15% | 12.6 |
| 4 | Resnet | 99% | 21% | 6.4 |
| 5 | Mobilnet | 99.8% | 52.3% | 3.4 |

From results tabulated above, following can be observed:

1. Transfer learning models perform better than basic CNN model
2. Mobilnet transfer learning algorithm with adam as optimizer has best test accuracy and least loss.
3. **Hence the model with MobileNet algorithm is taken as the final model.**

## Object Detection

Model performance has been assessed using dice coefficient.

MobileNet performance:

MobileNet has given good accuracies in both train set and test set. In the train set, the model has given 89.31% dice\_coefficient and in the test set 88.02% dice\_coefficient. Since the results are very close in train and test set, the model has consistently performed well and can be saved for future use.

Using the MobileNet, we predict the masked area of the car.

Yolo performance:

Dice coefficient of around 90% was achieved using the Mobilnet model. It detects the car by bounding box

# Comparison with benchmark

Taking benchmark as the base model, transfer learning improved the test accuracy considerably. Finally the MobileNet is the base model, and has an improvement of almost 50% in terms of accuracy.

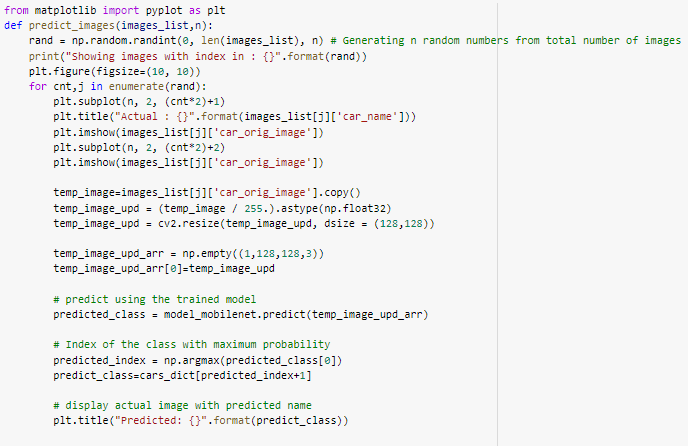
# Visualisations of Outputs

Accuracy and Loss plots for various models while training are as follows:

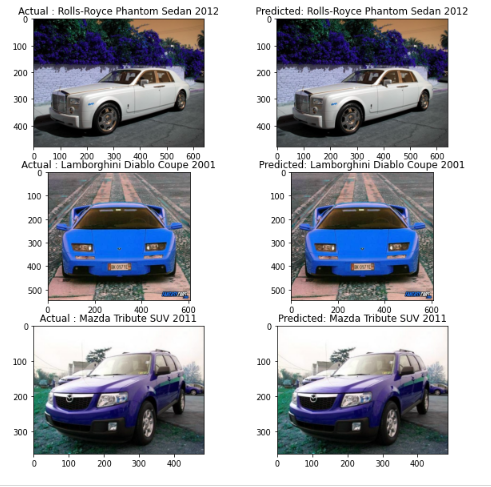
| S.No. | Model | Accuracy Plots | Loss plots |
| --- | --- | --- | --- |
| 1 | CNN |  |  |
| 2 | CNN with augmented images |  |  |
| 3 | vgg19 |  |  |
| 4 | Resnet |  |  |
| 5 | Mobilnet |  |  |

## Car model prediction with mobilnet

Following code was used to display the predicted car models:

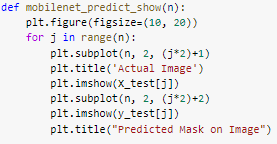


## MobileNet Car Model Prediction



## Bounding box detection using Mobilnet model

Following code was used to display n number of random images and predicted bounding box:

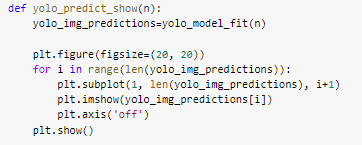


Images with predicted bounding box are displayed as below:



## Bounding box detection using yolo model

Following code was used to display n number of random images and predicted bounding box:



Images with predicted bounding box are displayed as below:

****

# Implications

Various use cases as given below can be used in real life for car detection:

* image recognition of a car model from image or video (live or recorded) is essential like vehicle tracking by security departments
* Traffic surveillance
* Vehicle identification by insurance firms
* Automating the car recognition for auto websites
* Shopping option of car model based on image
* Vehicle security etc.

Recommendations for real world applications:

The model needs to learn continuously with the relevant dataset. Confidence level is based on the model performance, gradually it needs to be improved until the business expectations are met.

# Limitations

The model built has following limitation:

1. accuracy achieved (51%) may not be sufficient for real world applications and need to be improved further.
2. Model is trained on 196 classes of car models. It can be extended to further models for real world applications.
3. The train images classes are not equally distributed. Few car models have only a few images and vice versa.

Following enhancements can be done to improve the model:

1. Keep training the model with real world dataset
2. Fine tune the model with hyper parameters

Following improvements can be done for model training:

1. Calculate and understand the system requirement based on the model complexity and set up the infrastructure accordingly
2. Build a model training pipeline which will be an automated process to train the model continuously

# Conclusion

This project provided the opportunity to use various computer vision algorithms to detect the car model and predict the bounding box of the car.

Following algorithms were used in the project

1. Convolutional Neural Network
2. CNN with image augmentation
3. Transfer learning using VGG19, ResNet and Mobilnet
4. Predicting bounding box using MobileNet and Yolo3

Handling a large dataset was one of the challenges where the RAM required was quite high. The models and image dataset were optimized to avoid RAM issues.

Finally, the MobilNet algorithm was able to detect the car model with good accuracy. Yolo and MobilNet were used to predict the bounding box.

Next time, the following actions can be included

* Model life cycle management and monitoring the model training and model prediction metrics
* Build a data pipeline to cleanse the input real time feed to continuously train the model

# 