A Project report on

Vehicle Make and Model Recognition

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY
In
COMPUTER SCIENCE AND ENGINEERING

Submitted by

Vanimina Vikramraj	(160102226)
Gontena Bhagyasri	(160102258)
Peddireddy Radha Krishna	(160102278)
Seshagiri Saimanoj	(160102284)

Under the Guidance of
Smt. K. ARUNA KUMARI
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SRKR ENGINEERING COLLEGE (A)

Chinna Amiram, Bhimavaram, West Godavari Dist., A.P.

[2019 – 2020]

SRKR ENGINEERING COLLEGE (A)

BONAFIDE CERTIFICATE

This is to certify that the project work entitled **"VEHICLE MAKE AND MODEL RECOGNITION"** is the bonafide work of "**VANIMINA VIKRAMRAJ, GONTENA BHAGYASRI, PEDDIREDDY RADHAKRISHNA, SESHAGIRI SAIMANOJ"** who carried out the project work under my supervision.

HEAD OF THE DEPARTMENT

Dr. MSVS Bhadri Raju

SUPERVISOR

Smt. K. Aruna Kumari

Assistant Professor

SELF DECLARATION

We hereby declare that the project work entitled “VEHICLE MAKE AND MODEL RECOGNITION ” is a genuine work carried out by us in B.Tech., (Computer Science and Engineering) at SRKR Engineering College(A), Bhimavaram and has not been submitted either in part or full for the award of any other degree or diploma in another institute or university.

1. V.Vikramraj 160102226

2. G.BhagyaSri 160102258

3. P.Radha Krishna 160102278

4. S.Saimanoj 160102284

TABLE OF CONTENTS

ABSTRACT

Detection and recognition of automobiles is an essential role in the territory of traffic regulation and management. Commonly, to handle this task, broad databases and area specific features are used to better suit the data. In our undertaking, we build test and test classifiers on a self-build small dataset for the goal of classification and identification of cars into their make and model from their images from media devices. We try different things with various degrees of transfer learning to fit the models to our domain. We report and contrast these outcomes with that of standard models, and talk about the points of interest of this methodology.

LIST OF FIGURES

1.INTRODUCTION

Vehicle is a great invention in human history. Since then it has become an integral part of modern people's life. The utilization of a colossal enormous number of vehicles mirrors the populace's portability, closeness, monetary, etc, and the examination of vehicle conduct is extremely critical for urban turn of events and government dynamic.

Vehicle Make and Model Recognition (VMMR) is an important technology for the Intelligent Traffic System (ITS) that detects vehicles from images and videos and classifies vehicles into different types or brands. With the rapidly increase of vehicles, more and more sensors and computers are employed to monitor traffic conditions. Automatic vehicle classification is a useful technique for secure access, traffic observation applications, accidents prevention and terrorist activities review etc. Usual vehicle identification methods are based on Automatic License Plate Recognition. However, ALPR requires advanced image capturing equipment to generate high-quality images and is sensitive to glare, dust and occlusion. ALPR frameworks are introduced in numerous countries for various purposes like law enforcement, electronic toll assortment, crime hindrance, traffic control, and so forth. ALPR systems distinguish a vehicle dependent on attached license plate. However, when two license plates are traded, the ALPR program can still identify both license plates but is ultimately unable to recognise the true identification. Licensing plates can be easily duplicated, occluded, and damaged. There are 3 examples in which it is almost difficult to identify the origin of the car is in Figure 1.1 Classification of vehicles could be an effective complement to the ALPR, which is capable to detecting and distinguishing vehicles with low-cost image/video capture equipment.

Figure:1.1 Examples of ambiguous, forged, or damaged license plates.

With the progress of deep learning, we are proposing a combination of vehicle detection and classification approaches based on CNN (convolutional neural networks) in this project. To detect the car in the image more effectively, an effective object recognition technique is used to identify objects in the picture.

The categorization of vehicle analysis is depicted in Figure 1.2 The first stage begins with the vehicle detection. After that the vehicle has been discovered, we can identify it by make and model (Hyundai Creta, Kia seltos, Mahindra Bolero, etc.). Self-sufficient vehicles and driver assistance, monitoring, traffic control and law implementation are just a couple of the applications that profit from automated vehicle analysis. It is a hard stretch for humans to watch, observe, and detect the ever-increasing variety of vehicles manually, particularly in urban environments. In contrast with the human user, the computer vision program can track traffic for a prolonged period of time without weariness. The related costs of computer systems are lower and may be adjusted to meet the necessary cost / performance ratio.

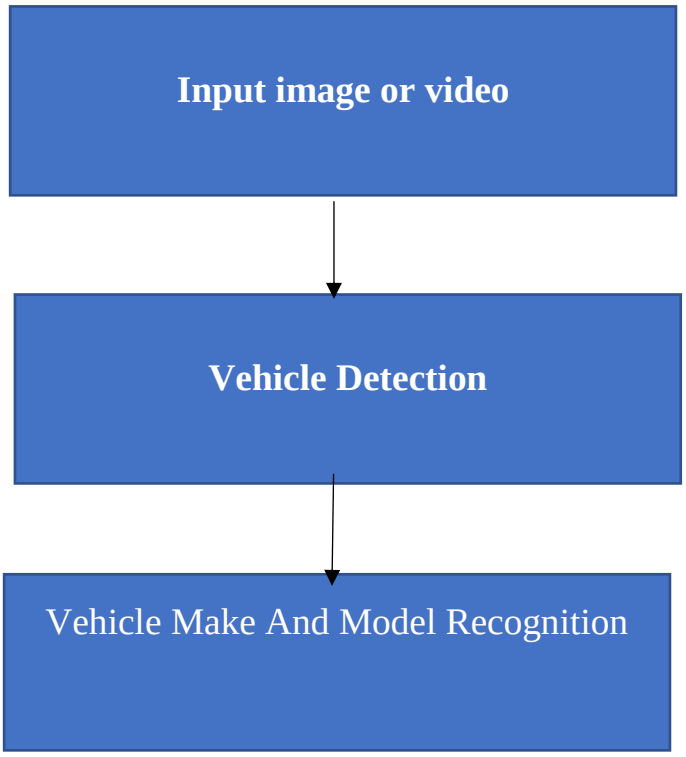


Figure:1.2

2.PROBLEM STATEMENT

The main of this project is to predict the make and model of a vehicle with the help of models made from convolution neural networks with the help of transfer learning. The project is based on a small dataset based on some popular cars in India, the goal is to try different things with various degrees of transfer learning to fit the models to our domain. We report and contrast these outcomes with that of standard models, and talk about the points of interest of this methodology for vehicle make and model recognition. Since, VMMR is very important traffic control and maintenance, police and many more fields.

3.LITERATURE SURVEY

The development of image process innovations, alongside the wide installation of reconnaissance cameras, empowers image-based vehicle identification and classification. Various ways to deal with image-based vehicle identification and distinction have been proposed throughout the most recent years.

Kazemi et al. Utilized 3 separate types of feature extractors, Wavelet transform, Curvelet transform and Fourier transform,, to distinguish and classify 5 vehicle models. They used the k-nearest neighbour for the classifier. [1] They then compared the 3 proposed approaches and find that the Curvelet transform is the one which can be used to extract better features.

Wen et al. [3] used feature pool of Haar-like for a 3232-grayscale image patch to reflect the appearance of a vehicle and then suggested AdaBoost's rapid incremental learning algorithm to increase AdaBoost's efficiency.

Chen et al. [2] Presented a program for the monitoring, registration, and identification of automobiles from side of the road CCTV. Initially, a Kalman filter followed a vehicle to allow majority-vote classification over a number of successive frames at that point prepared a support vector machine (SVM) utilizing a blend of vehicle silhouette and intensity-based pyramid histogram of focused gradient (HOG) features extracted after context subtraction, and then the foreground blobs are classified with with majority-vote.

Arrospide and Salgado [4] after performing serial analysis on individual performance of top techniques that are used for vehicle classification and described his results as the features Gabe and HOG result the best outcomes and trounce principal component analysis (PCA) and various classifiers concentrated on features like gradient and symmetry.

Wang et al. [5] published a vehicle identification algorithm based on novel deep learning with 2D deep-belief network; the 2D-DBN architecture mentioned utilizes 2^{nd} order plane rather than 1^{st} order vector as input and utilizes bilinear projection for preserve discriminatory information, So as to assess the deep architecture size that improves the achievement rate of vehicle identification. Their algorithm has done best in the datasets.

He et al. [6] has published a very new productive vehicle identification and classification approach dependent on convolutional neural network, This method results extraction of features, these features outperform those created by he traditional approaches.

Sermanet et al. [7] has reported an integrated framework to work with the deep learning for the identification, classification, and localization of objects. This specific framework presented by him

grabs very reliable outcomes for the identification and classifications tasks. The excellent object detection models up to now, which are based on deep learning incorporate R-CNN, YOLO, SSD, Faster R-CNN, Fast R-CNN, and R-FCN;

For a time, object detection and classification upheld on CNNs are extremely effective in the field of computer vision as of late. The essential work with respect to object recognition and classification upheld on deep learning has been done in 2013.

4. SOFTWARE REQUIREMENTS SPECIFICATION

4.1 Purpose of the system

The purpose of the system is mainly to replace the existing VMMR systems with a real-time VMMR system with better performance than in terms of recognition rate and processing speed.

4.2 Scope

The scope of the system is to recognize make and model of a vehicle with high accuracy.

4.3 Objectives

The main objectives of the project are

a. Gathering data and creating data set for the problem.
b. Creating a neural network and create train, test classifiers.
c. Evaluate and improve model.

4.4 Existing Systems

a. Yaqoob, H., Bhatti, S. and Khan, R.R.A., Car Make and Model Recognition using Image Processing and Machine Learning.
b. Biglari M, Soleimani A, Hassanpour H. A cascaded part-based system for fine-grained vehicle classification.
c. Nazemi A, Azimifar Z, Shafiee MJ, Wong A. Real-Time Vehicle Make and Model Recognition Using Unsupervised Feature Learning.

4.5 Proposed System

The proposed system is based small dataset of images, we intend to use CNN with transfer learning to attain a high accuracy with small dataset.

4.6 Requirements

4.6.1 Functional Requirements

4.6.1.1 The system must be able to recognize a vehicle and provide information about vehicle make, model and generation.

4.6.2 Non-functional Requirements

4.6.2.1 Hardware considerations:

- Processor : Intel Core i5 8th gen or higher
- RAM : 8GB
- Disk Space : 256GB

4.6.2.2 Software considerations:

- Programming language: Python

- IDE : Anaconda(Jupyter Notebook)
- Operating system : Windows 7 or higher (64 Bit)

4.6.2.3 User interface factor:

- The application should provide user interface to see the working model and various visualizations of the models.

4.6.2.4 Performance characteristics:

- It should quickly respond for the requests of the user of the system.
- The system should respond within 10 seconds from the user request.
- The accuracy of the prediction should be approximately greater than 80 percent.

4.6.2.5 Error handling:

- In case of any Error, the system should display a meaningful error message to the user, such that user can correct his error.

4.6.2.6 Quality issues:

- Data provided by the system is reliable as it directly interacts with the user. Quality issues refer to how reliable, available and robust should the system be.

4.6.2.7 Resource issues:

- System can operate in extreme conditions like low memory, low power condition.
- Resources of the computer should meet the hardware considerations. Anything less than this is not sufficient for proper functioning of the system.

5. SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE:

System Architecture describes the overall structure of the system and the ways in which the structure provides conceptual integrity. The below figure 5.1 shows the system architecture of the project. It includes mainly four parts as shown in the diagram.

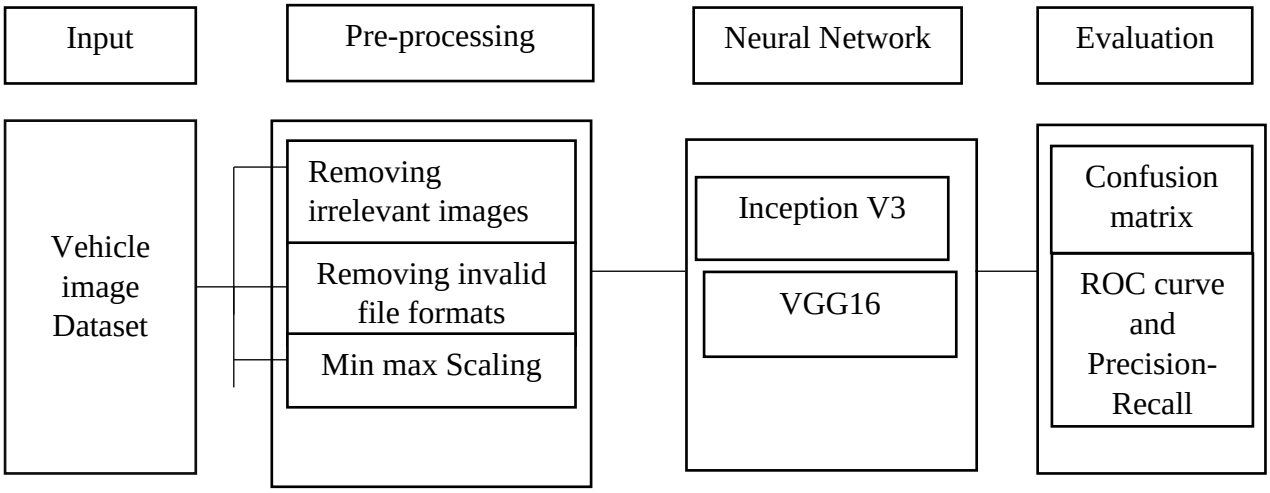


Figure:5.1 System Architecture

The figure 5.1 is the system architecture which defined the working of our model. We consider dataset containing 642 images and then loaded to the kernel. In the preprocessing stage, the input to the model is taken directly as an image and checked for any corrupted images in the dataset. The Data labelling is performed for the images in the dataset that are labelled with names and numbers. We then split the dataset into training and testing data as per our requirements. The input image is not in fixed size and the images are shifted to using the Imagedatagenerator and rescale them to get all the pixels between 1 to 255.

5.2 UML:

UML refers to Unified Modelling Language which is commonly used in the general-purpose modelling language. The main objective of UML is to define a visualized representation of how a system has been designed. It is similar to blueprints used in other fields of engineering. It is not a programming language but rather a visual language.

5.2.1 USECASE DIAGRAM:

Use case diagrams are the diagrams that are usually mentioned as behaviour diagrams that are used to describe a group of actions (use cases) that come across some system or systems could perform together with one or more external system users (actors).

Use serves as a way for:

- Capturing requirements of the system.
- Communication with the ultimate consumer.
- System testing.

We should examine the actors properly and determine what the actor actually does with the system. As it is not possible to cover all the needs of the system in a solitary use case, therefore we require multiple use cases. Altogether these aggregated use cases define all means of using the system.

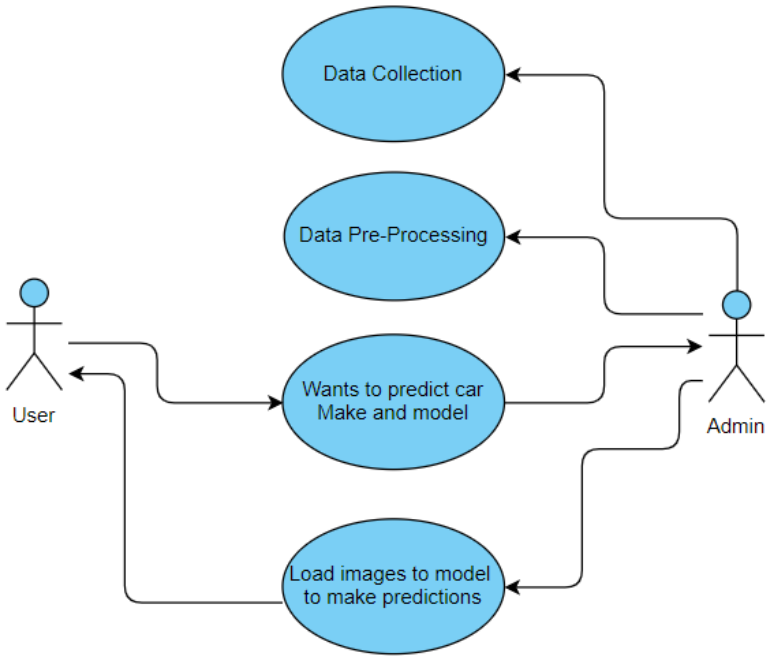


Fig: 5.2 Usecase Diagram

5.2.2 CLASS DIAGRAM:

A class diagram is a visual representation of the classes, their attributes, methods which are technically known as operations and the relations among them. It also includes the constraints over the system. This is broadly used for model the system as it would truly be mapped between the different objects.

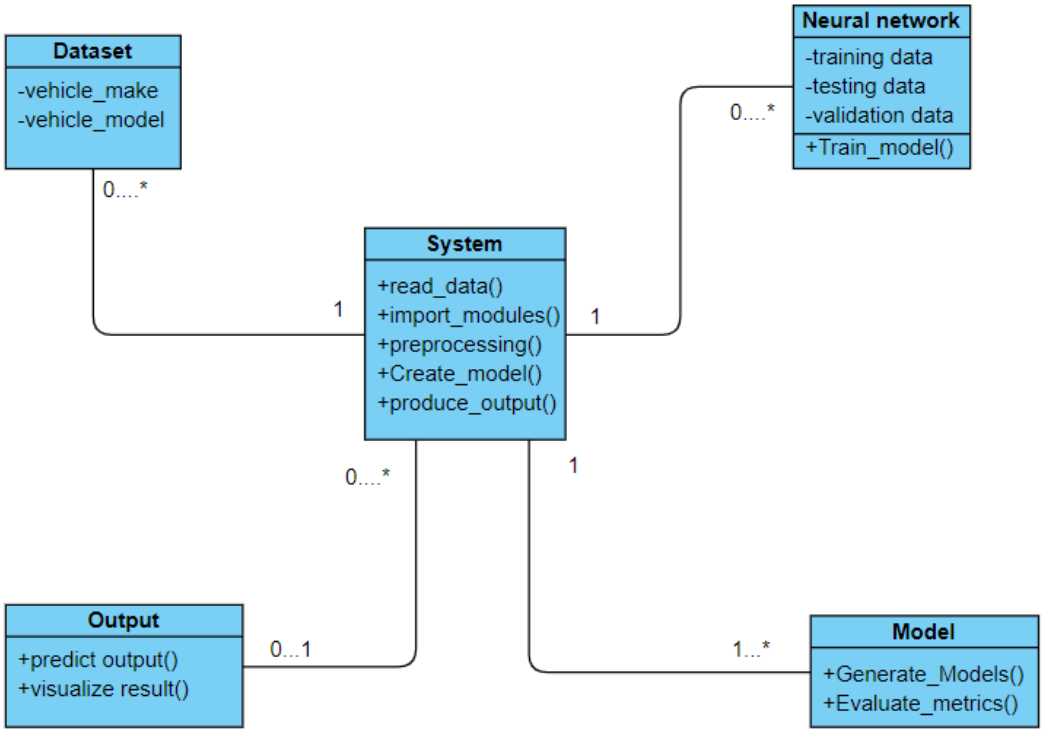


Fig: 5.3 Class Diagram

5.2.3 SEQUENCE DIAGRAM:

A sequence diagram is the interaction between the objects in a sequential order. They capture the interaction between objects in the context of collaboration. Sequence diagrams focus on time and they show the interaction order visually.

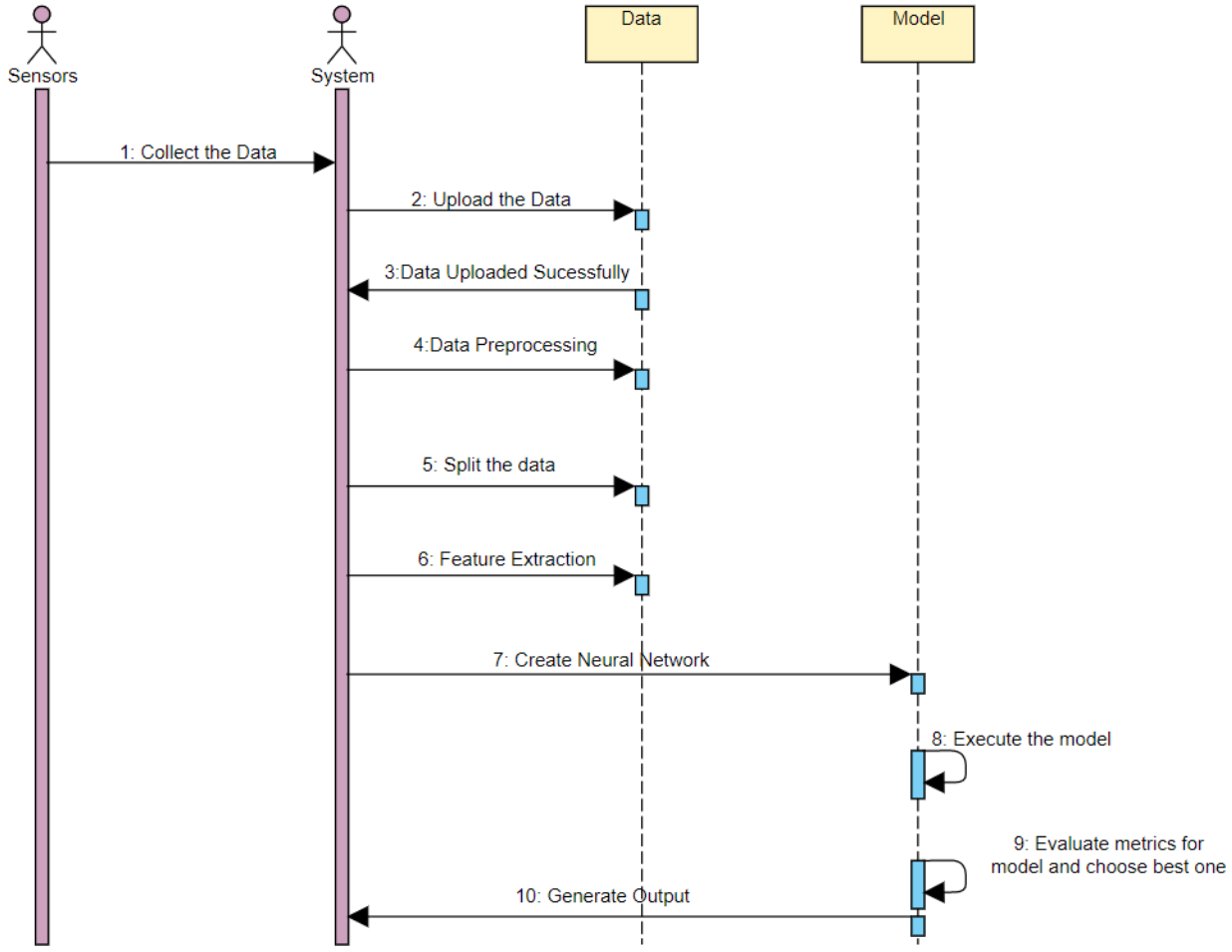


Fig: 5.4 Sequence Diagram

6. METHODOLOGY

6.1 Data set:

We chose our problem as Most Stolen cars in India.

We Referred the following sources to make a list of classes for our problem statement:

- https://m.economictimes.com/news/politics-and-nation/grand-theft-auto-car-thieves-love-for-suvs-is-giving-sleepless-nights-to-insurance-cos/articleshow/71120982.cms
- https://www.msn.com/en-in/news/other/10-most-stolen-cars-in-india/ss-BBgeX2A

We decided to make the following classes to initially train our model

1)Hyundai Creta (2015-2020, 2020- present)

2)Maruti Suzuki Vitara Brezza (2016-present)

3)Mahindra Scorpio (2006-2014, 2014-present)

4)Mahindra Bolero

5)Kia Seltos

6)Mg Hector

7)Hyundai Venue

8)Hyundai Santro

We used the following sources to get the images to create our image dataset:

- www.cardekho.com
- www.carwale.com
- www.olx.in
- www.quikr.com etc.

6.2 Methods:

6.2.1 Convolutional Neural Networks (CNN):

Convolutional Neural Networks is a class of Neural Networks that have verified terribly effective in areas like image recognition and classification. There are differing types of hidden layers in CNN. The hidden layers of a CNN generally comprise convolutional layers, pooling layers and absolutely connected layers. Here in convolution and pooling layers LeakyReLu is employed as activation perform.

In CNN language, the 3×3 matrix is named a ‘filter’ or ‘kernel’ or ‘feature detector’ and therefore the matrix formed by sliding the filter over the image and computing the scalar product is named

the ‘Convolved Feature’ or ‘Activation Map’ or the ‘Feature Map‘. It is necessary to notice that filters acts as feature detectors from the initial input image.

Let us acumen a convolution and pooling layer works to induce a transparent understanding of CNN.

Convolution:

Convolution operates on 2 signals (in 1D) or 2 pictures (in 2D) during which one are taken as sign or image, and also the alternative referred to as the kernel are taken as a filter on the input image, manufacturing an output image. Thus convolution takes 2 pictures as input and produces a 3rd as output.

Pooling: It's a sample-based discretization method. Pooling progressively reduces the dimensions of the input representation. Pooling helps to scale back the amount of required parameters and therefore the amount of computation required. It also helps control overfitting.

There are unremarkably 2 varieties of pooling referred to as max and min pooling. Max pooling is substitution the complete sample with the most prices from the chosen region and min pooling is substitution the complete sample with the minimum price from the chosen region. Figure 5.2 shows the max pool with stride 2.

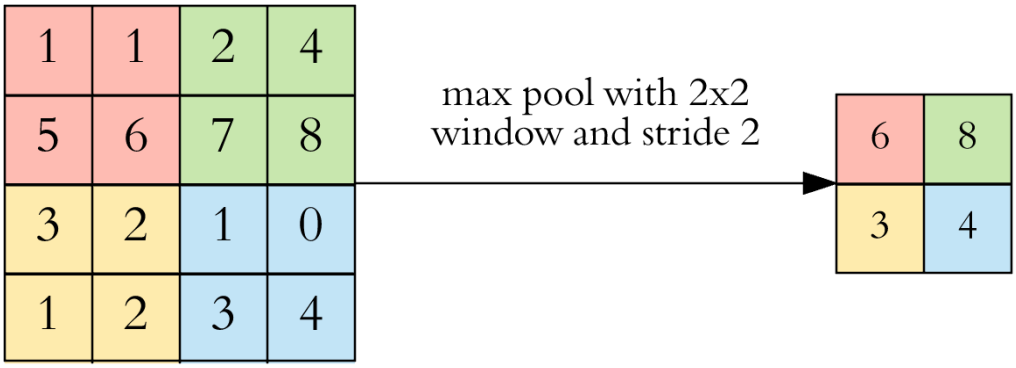


Figure:6.1

Strides:

Stride is the number of pixels shifts over the input matrix. It is defined as the number of steps the kernel or the weight matrix takes while moving across the entire image moving N pixel at a time. If the matrix moves N pixel at a time, it's referred to as stride of N.

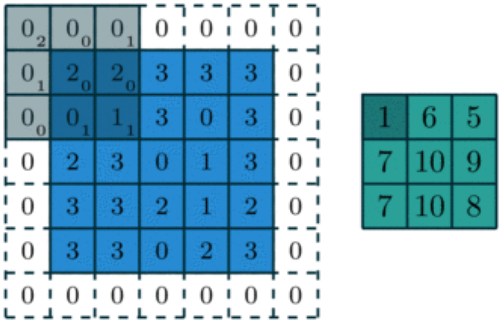


Figure:6.2

Activation Layer:

Leaky ReLu:

This is the corrected linear measure that uses activation function. It gives zero for any negative pixel value; similarly it gives the same value for any positive pixel value. It can be formulated as below:

$$f(x) = \max(0, x)$$

For a clear understanding epoch and steps per epoch should be explained briefly.

Epoch:

It is the forward and backward pass on all the training data or simply the complete iteration on all the training data. Based on the batch size and the count of the data we need to give epoch.

Steps per Epoch:

It is simply the division of count of the image dataset to the batch-size. It can be formulated as below:

$$Steps - per - epoch = (count\ of images)/batchsize$$

For example the dataset contains 10000 images and the batch size is 100 then the steps per epoch is 10000/100=100 steps.

Batch-Size:

It is simply how many training examples were taken for one forward and backward pass. They can be 10, 32, 64, 100, etc. But the increase in batch size makes the memory full and it takes more time for the process to complete.

Flattening:

This is a reasonably simple step. You flatten the pooled feature map into a sequential column of numbers (a long vector). This enables that information to become the input layer of a man-made neural network for further processing.

Fully Connected Layer: The fully connected layer may be a traditional Multi-Layer Perceptron. It uses a classifier within the output layer. The classifier is typically a softmax activation function. Fully connected means every neuron within the previous layer connects to each neuron within the next layer.

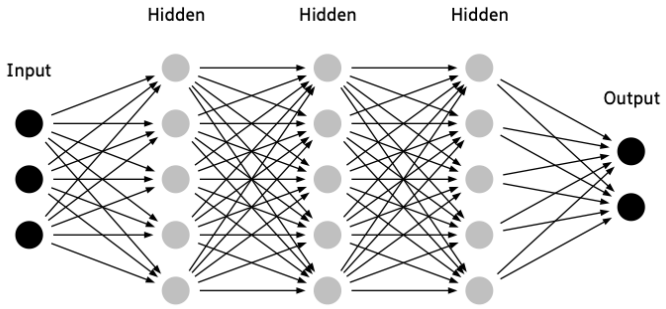


Figure:6.3

Softmax:

This brings the values between zero and one and makes them add up to one (100%). The softmax perform takes a vector of scores and squashes it to a vector of values between zero and one that add up to one.

Dropout layer:

Dropout layer in deep learning is a technique to overcome the problem of over fitting. Dropout method takes a float number between 0 and 1. This value indicates ignoring certain set of neurons while training to avoid over fitting.

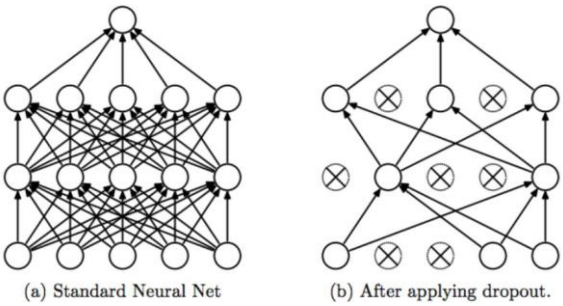


Figure:6.4

Dense layer:

The name suggests that layers are fully connected by the neurons in a network layer. Each neuron in a layer receives an input from all the neurons present in the previous layer. Thus, they're densely connected.

Adam optimizer:

Adam is an adaptive learning rate optimization algorithm that's been designed for training deep neural networks. It is used to update network weights in training data.

Loss function:

A loss function, conjointly spoken as a cost function, measures the compatibility between output predictions of the network through forward propagation and given ground truth labels. Commonly used loss function for multiclass classification is cross entropy, whereas mean square error is often applied to regression to continuous values. A type of loss function is one in all the hyper parameters and wishes to be determined consistent with the given tasks.

Algorithm 1:

Input: Cars pictures are taken as input.

Output: Predicted make and model of vehicle is obtained as output.

Steps:

1. Start
2. Initially images are loaded in to the kernel and then appended into an arrays of x and z in which they are labeled with name and number.
3. The preprocessing removes unwanted images and any corrupted data present in the dataset.
4. Visualize some random images with labels and make uniform by giving specific
5. Creating training and validation generators with data augmentation in train generator in which we split the dataset and do the shifts of the image to bring them in correct orientation and we rescale the images between 1 to 255 pixels.
6. The base sequential model is built with different layers of batch size 128 and epochs of 50 respectively. The First conv layer is built with filters size 32, kernel size 5 X 5, padding same and input size (150,150,3). Second Conv layer is built with filters 196, Third Conv layer is built with filters 256, The Forth Conv layer is built with filters 512 and then we Flatten the matrix to a single array.
7. The Fully connected layer 1 is built with dense 1024 and Fully connected layer 2 is built with dense 512. At Last Output Layer is built with activation function softmax to classify the flower images to which class they belong to.
8. Compile the model with loss catergorical_crossentropy and the loss is reduced with adam optimizer and we evaluate the metrics with accuracy
9. Train the base model with all the defined
10. Predicting the flower from the trained model

11. Evaluating the Model Performances with plot graphs

12. Stop

In this Algorithm which is built from scratch we first load the images and append then in into arrays to label and number them. In the second step we remove the unwanted and corrupted images. Next we visualize some images randomly. In the next step we split training and validation set and do data augmentation of all shifts using Imagedatagenerator and then we rescale the images between 1 to 255. Next the layers of the sequential model are created. The filters for the layers are 32,196,256,512 accordingly. We have used Leaky_RELU as activation to normalize the values. The max pool is in layers to get the max values of the convoluted images. In the next step the layers are flattened using flatten() function. The fully connected layers are created with dense 1024 and 512 and at last the output layer is built by using the activation function softmax for classification. Next we compile the model with optimizer adam, loss function categorical_crossentropy and metrics of accuracy. In the eight step we train the model with all the required parameters using model.fit and from the last step we predict the flower from the train model.

Considering our 150 X 150 input image, if we add a padding of 1, 3 X 3 filter, then a simple equation can help us figure out the shape of the output: n+2p-f+1. Thus, the output shape will be: 150+2(1)-3+1 = 150. Therefore, the output will be a 150 x 150 matrix, just like the input image. When no padding is applied, this is called a “valid convolution”. Otherwise, it is termed “same convolution”.

Consider a one dimensional convolution, if input F and kernel G are both functions of one dimensional data like time series data then their convolution is given by this equation 1. It looks like fancy math but it has meaning the equation represents the percentage of area of the filter G that overlaps the input F at a time tau over all time T. Since tau less than zero is meaningless and tau greater than T represents the value of a function in the future which we don’t know we can apply tiger bounds to the integral.

This corresponds to a single entry in the 1d convolutional tensor that is the t f – entry to compute the complete convolve tensor we need to iterate T over all possible values in practice we may have multi-dimensional input tensors that require multi-dimensional kernel tensor. In this case we consider image input I and a kernel H whose convolution is defined as shown. The First equation performs convolutions by sliding the image over the kernel and the second equation does it the other way around, it slides the kernel over the image, since the usually less possible values for x and y in the kernel than in the kernel than in the image we use the second for the convolution. The result will be a scalar value we repeat the process for every point xy for which conv exists on the

image these values are stored in a convolve matrix represented by I*H this output constitutes a part of a feature map.

6.2.2 VGG16:

The Architecture:

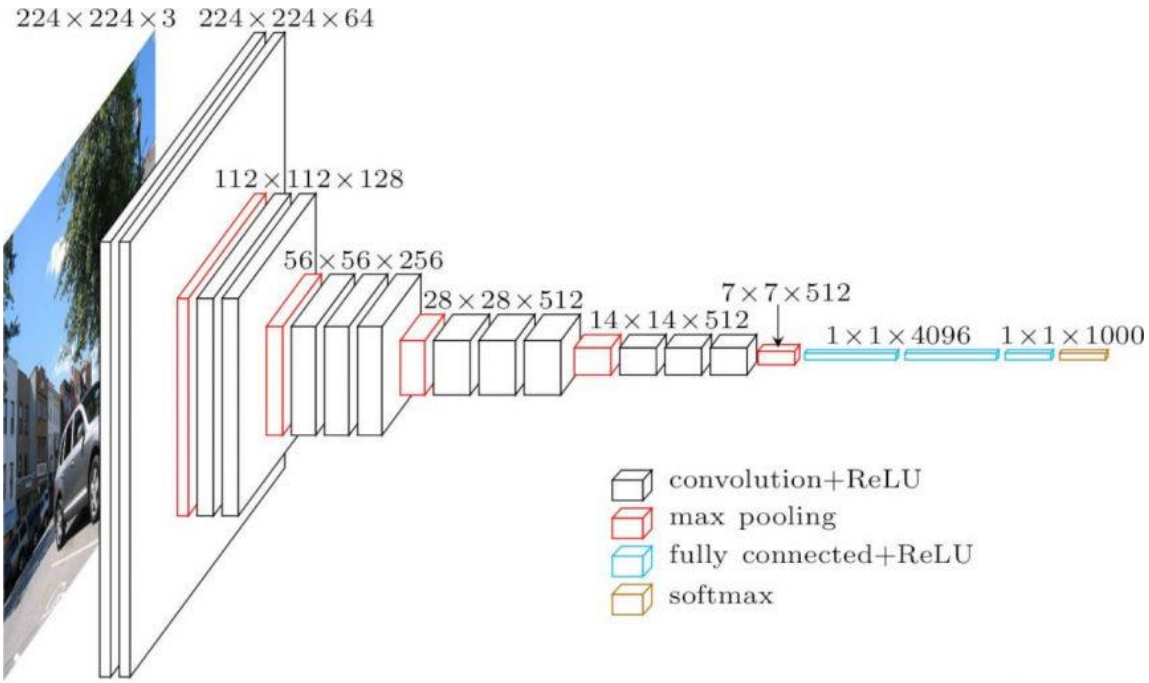


Figure:6.5

VGG16 is a convolutional neural network model planned by K. Simonyan and A. Zisserman from the University of Oxford within the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". The model achieves 92.7% top-5 check accuracy in ImageNet, that could be a dataset of over fourteen million pictures happiness to one thousand categories. It absolutely was one among the notable model submitted to ILSVRC-2014. It makes the advance over AlexNet by replacement massive kernel-sized filters (11 and five within the 1st and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one when another.

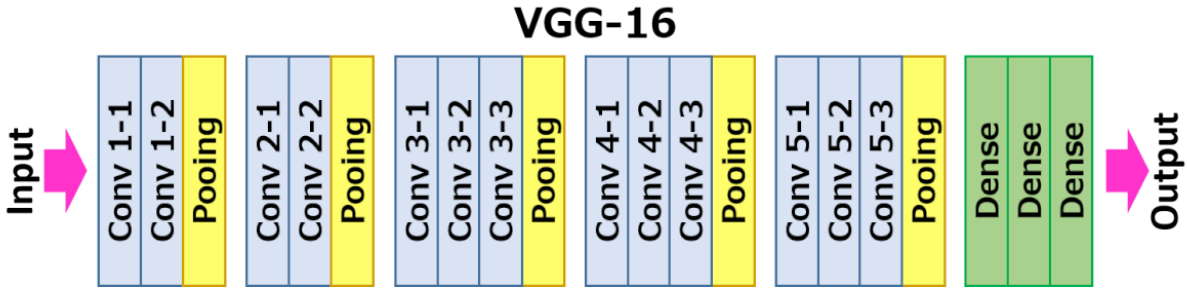


Figure:6.6

The input to cov1 layer is of fastened size 224 x 224 RGB image. The image is passed through a stack of convolutional layers, wherever the filters were used with a awfully little receptive field:

3×3. In one among the configurations, it additionally utilizes 1×1 convolution filters, which might be seen as a linear transformation of the input channels. The convolution stride is fastened to one pixel; the spatial artifact of conv. layer input is specified the spatial resolution is preserved once convolution, i.e. the artifact is 1-pixel for 3×3 conv. layers. Spatial pooling is disbursed by 5 max-pooling layers, that follow a number of the conv. layer . Max-pooling is performed over a 2×2 picture element window, with stride two.Three Fully-Connected (FC) layers follow a stack of convolutional layers the primary 2 have 4096 channels every, the third performs a thousand-way ILSVRC classification and therefore contains 1000 channels (one for every class). The ultimate layer is that the soft-max layer. The configuration of the totally connected layers is that the same all networks.

6.2.3 InceptionV3:

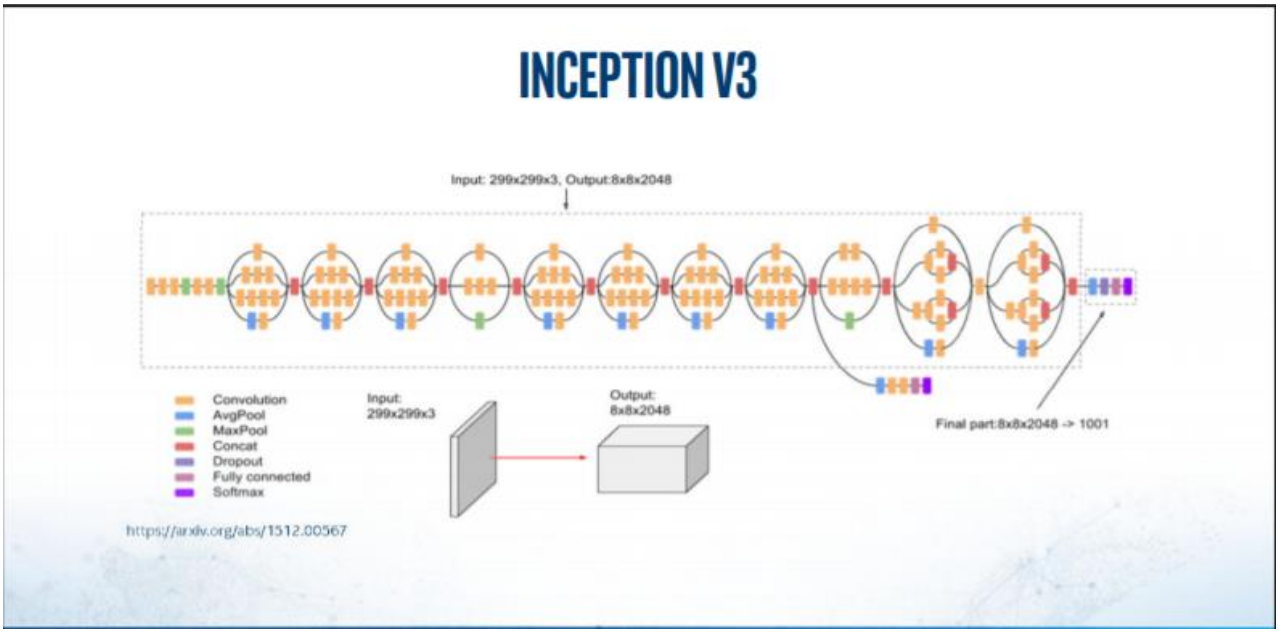


Figure:6.7

Inception-V3 is one among the convolutional neural network which is trained on 1,000,000 different pictures. The network increases the width of every layer instead of the depth. It is also same as Network-In-Network during which a 1x1 convolution is placed as a bottleneck between 3x3 convolutions to scale back computational complexity. The Inception network builts on the ideas of the Network-in-Network idea with a more complex hidden block consisting of 1x1, 3x3, a 5x5, and max-pooling layer all composed into one neural network block. The Inception network focuses on the width of every layer.

The popular versions square measure as follows:

- Inception v1.
- Inception v3 and Inception v2
- Inception v4 and Inception-ResNet.

Looking at figure 6.7, we see this concept of stacking many various layers together clearly illustrated. This is often in contrast with a standard CNN which might line these different operations up sequentially such the output of the 1x1 convolution is that the input to the Inception module copies the first input and feeds this same input to every of the various operations.

Algorithm 2:

Input: Cars pictures are taken as input.

Output: Predicted make and model of vehicle is obtained as output.

Steps:

1. Load the Sequential Model
2. Add Inception module and its weights
3. Adding Fully connected layer 1
4. Adding Fully connected layer 2
5. Adding Output Layer
6. Compile the model with optimizer adam to reduce the loss.
7. Finally Training with model with the defined epochs and batch size
8. Stop

In this algorithm we first define the already built sequential model and then we add the base inception v3 model in which the weights are loaded automatically with the predefined input size. In the next step we freeze the initial layers, setting last layers as trainable and unfreezing layer layers. Next we add the flatten layer with fully connected layer of dense 1024 and activation function LeakyReLU of alpha value 0.02. The dropout layer is added with rate 0.1. In the next step the fully connected layer 2 is added with dense 512 and activation function LeakuReLU of alpha= 0.02. The output layer is added with numclasses and activation “softmax” for classification and then we compile the model with loss= 'categorical_crossentropy', optimizer = adam to reduce the loss occurred in the model with metrics= accuracy. In the last step we train the model we predefined transfer learning model inception v3 with our requirements.

7. TESTING AND RESULT ANALYSIS

7.1 Result Analysis:

7.1.1 Visualizing some random images from Dataset:

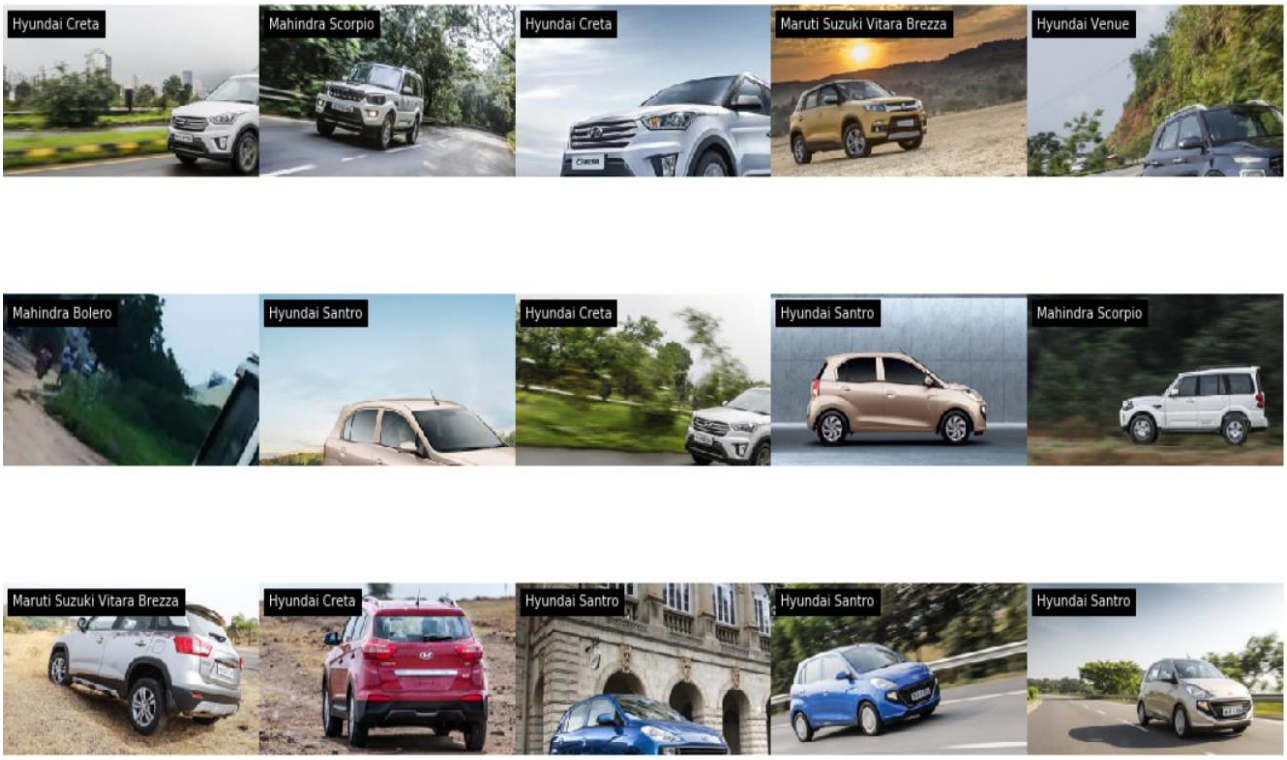
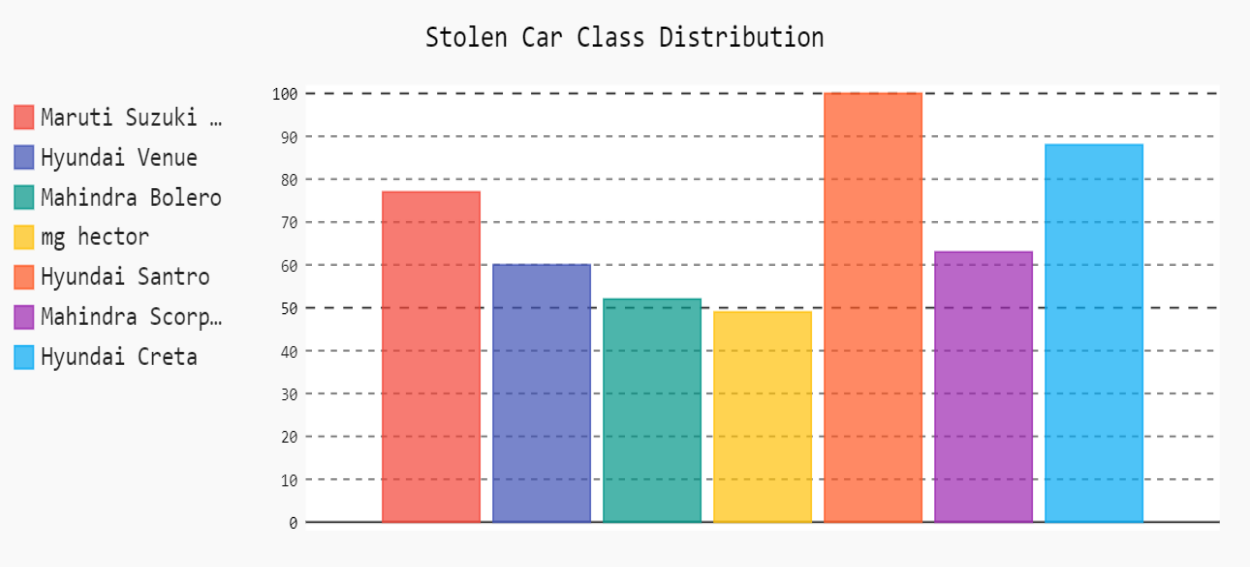


Figure:7.1

7.1.2 Visualizing distribution of Images in dataset before pre-processing:



After Visualizing the distribution, we found some minority classes in the dataset. Since we cannot work on undistributed data since it raises fitting problem.

Figure:7.2

7.1.3 Visualizing distribution of Images in dataset after pre-processing:

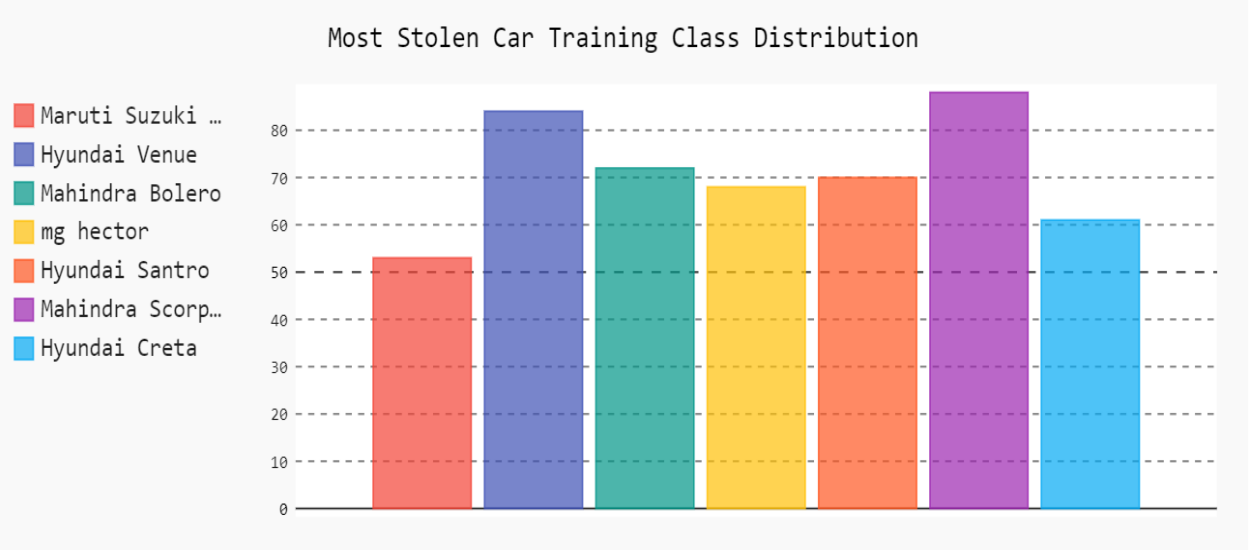


Figure:7.3

By using the augmentation techniques, we have oversampled minority classes in training set. We have not done these steps in validation or test in order not to create any bias on the data.

7.1.4 Visualizing some random images from dataset after pre-processing:

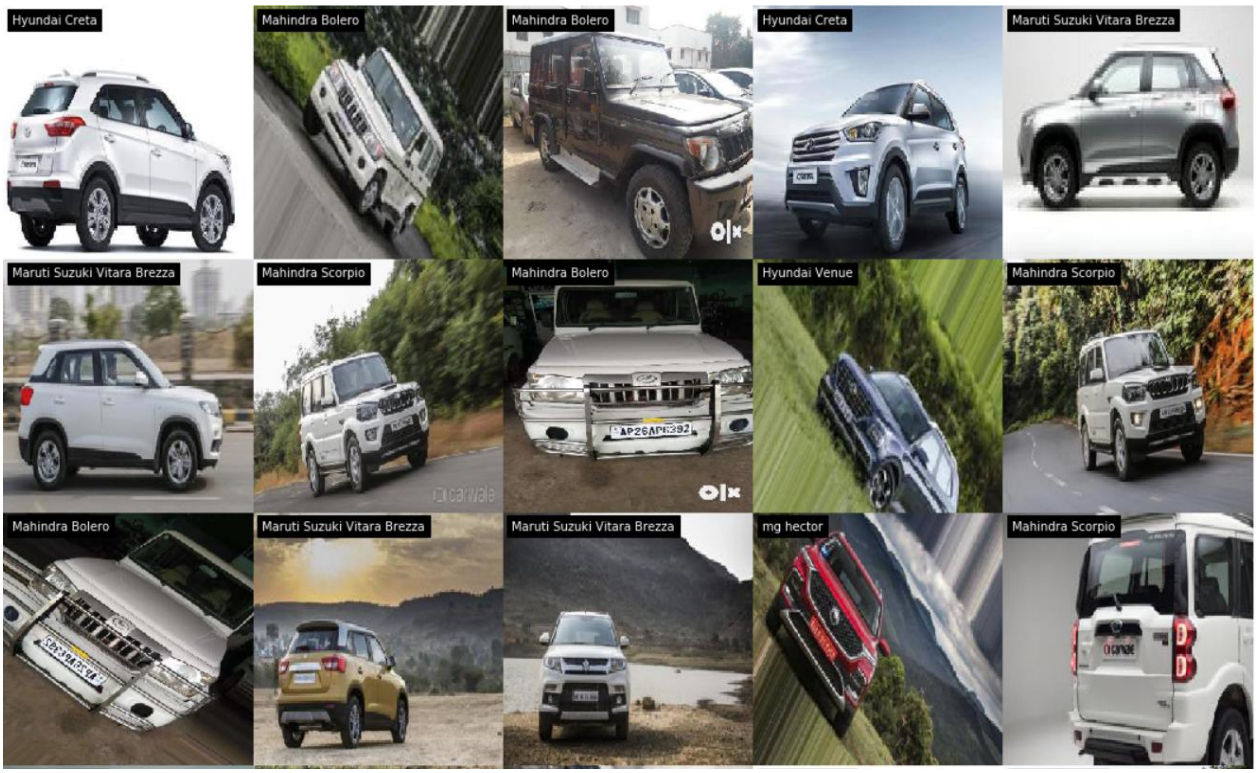


Figure:7.4

7.1.5 Testing:

Classification metrics:

Recall: Recall is the fraction of positives that were correctly identified.

Recall = TP/(TP+FN)

Precision: Accuracy of positive predictions.

Precision = TP/(TP + FP)

F1 Score: The F_1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0.

F1 Score = 2*(Recall * Precision) / (Recall + Precision)

Support: The support is the number of samples of the true response that lie in that class.

7.1.5.1 VGG16 Model:

We have obtained an accuracy of 88% with VGG16 model with a size of 312mb

Classification report:

	precision	recall	f1-score	support
Hyundai Creta	0.73	0.58	0.65	19
Hyundai Santro	0.68	0.95	0.79	20
Hyundai Venue	0.54	0.58	0.56	12
Mahindra Bolero	0.73	1.00	0.85	11
Mahindra Scorpio	0.88	0.54	0.67	13
Maruti Suzuki Vitara Brezza	0.83	0.59	0.69	17
mg hector	0.75	0.82	0.78	11
accuracy			0.72	103
macro avg	0.73	0.72	0.71	103
weighted avg	0.74	0.72	0.71	103

7.1.5.1.1 Confusion matrix:

The figure 7.5 below represents the confusion matrix for VGG16 model, which is a description of the study of prediction results on the classification problem. The amount of correct and incorrect predictions is summarized with count values and divided down by each class. We can observe that we have very good predictions with every classes but have small setbacks with Maruti Brezza often misclassified for Hyundai Santro and Hyundai Venue for Hyundai creta.

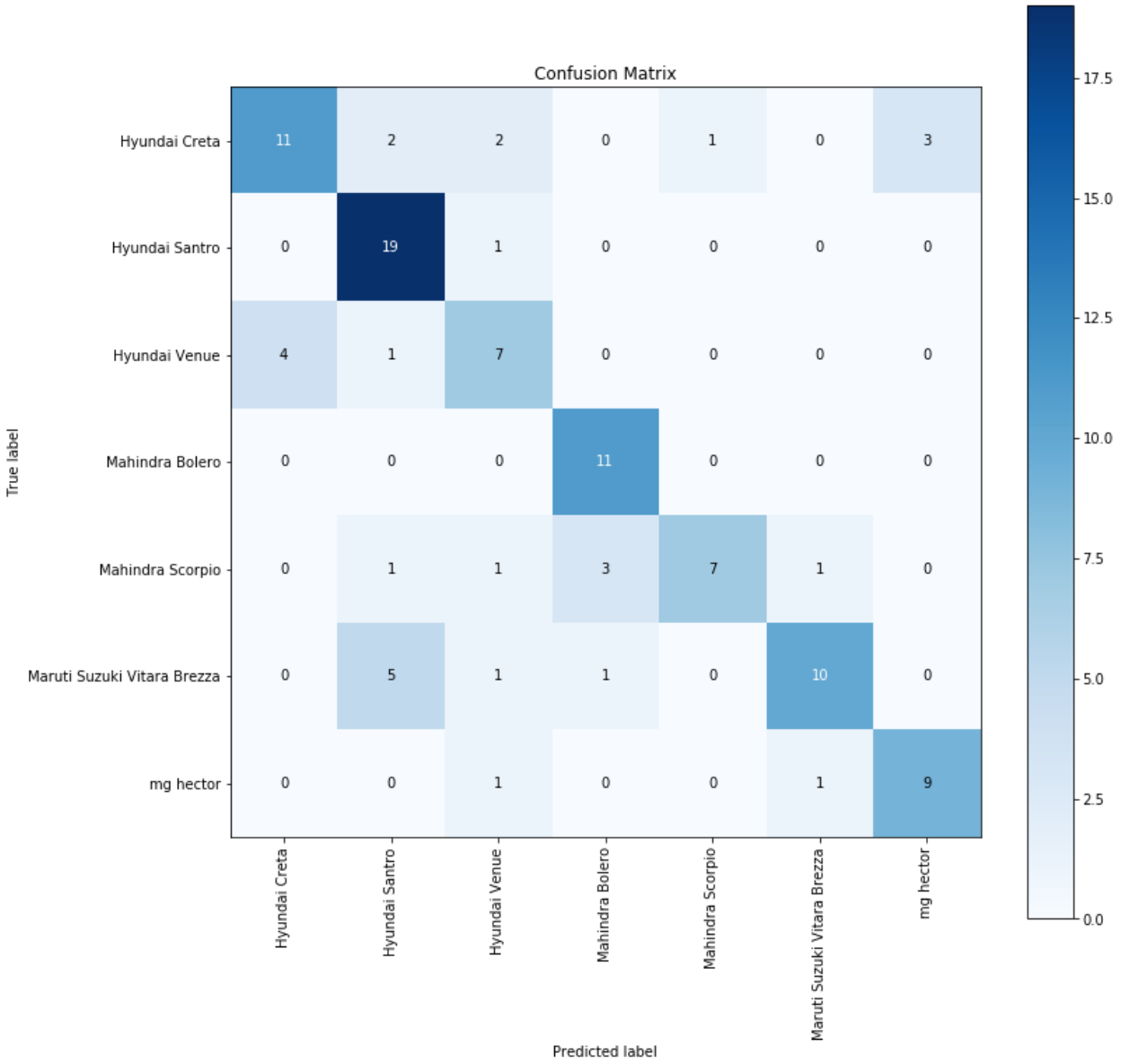


Figure:7.5

7.1.5.1.2 Precision-Recall curve:

A precision-recall curve is a plot of the precision (y-axis) and the recall (x-axis) for different thresholds. The curves with greater area under them imply a good classifier. Since we find a class somewhat near the threshold there is room for improvement for those classes. A model with high precision and recall is desired.

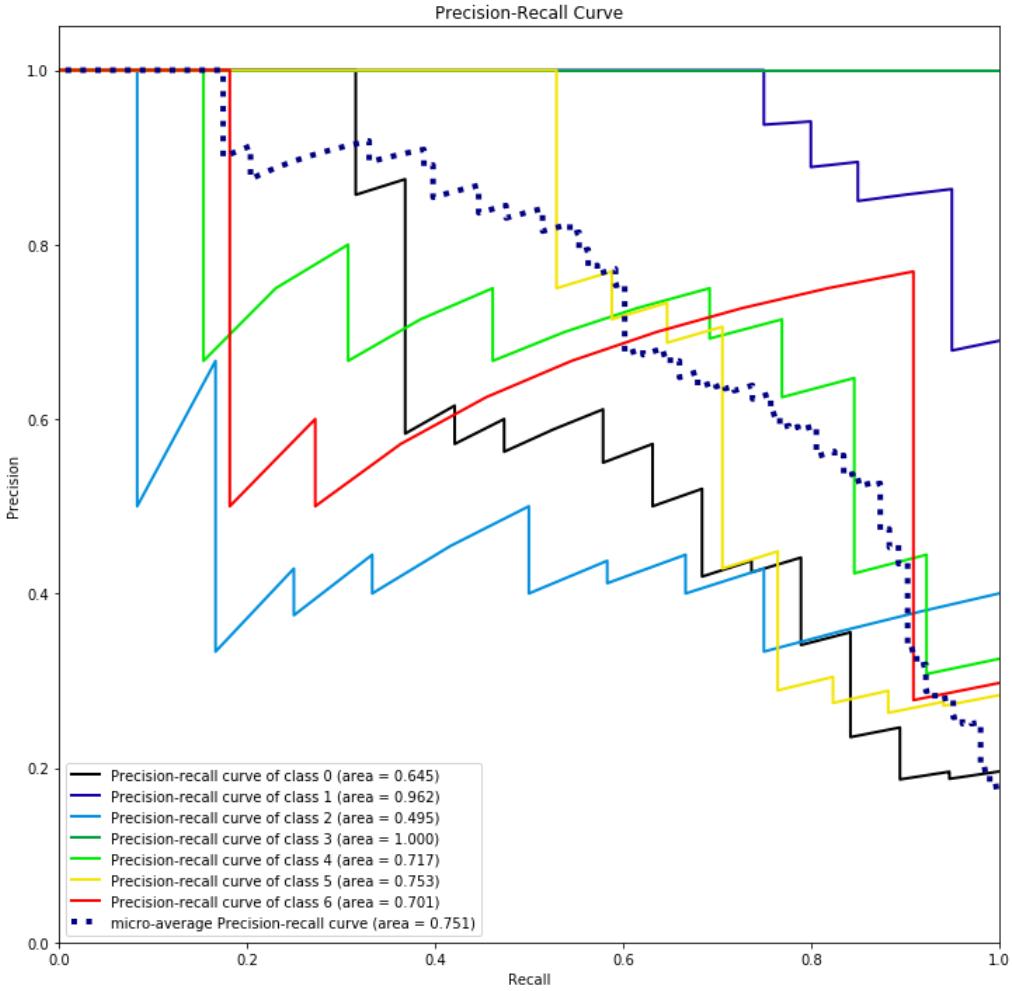


Figure:7.6

7.1.5.1.3 ROC curve:

The below figure 7.7 is a plot of the false positive rate vs. the true positive rate on x and y axis for various candidate threshold values between 0.0 and 1.0. In other words, it plots the false alarm rate vs. the hit rate.

True Positive Rate = TPs / (TPs + FNs)

False Positive Rate = FPs / (FPs + TNs)

The points to be noted are

- A model that randomly guesses the label will result in the black line and our desired model is the one that has a curve above this black line.
- A ROC that is away from the black line is the best one.

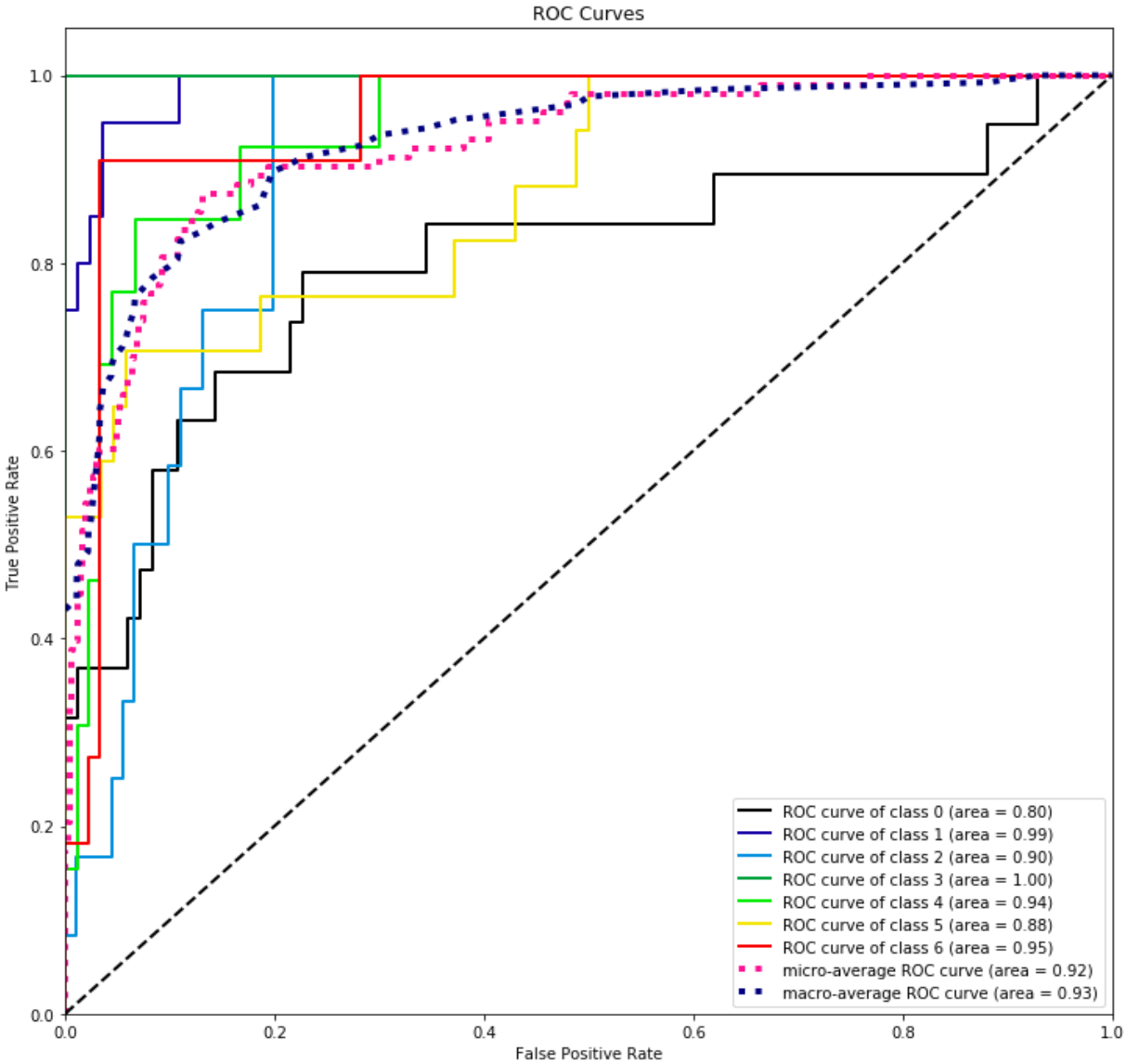


Figure:7.7

7.1.5.2 InceptionV3 Model:

We have obtained an accuracy of 82% with InceptionV3 model with a size of 118mb

Classification report:

	precision	recall	f1-score	support
Hyundai Creta	0.85	0.58	0.69	19
Hyundai Santro	0.95	0.95	0.95	20
Hyundai Venue	0.25	0.08	0.12	12
Mahindra Bolero	1.00	0.09	0.17	11
Mahindra Scorpio	0.50	0.77	0.61	13
Maruti Suzuki Vitara Brezza	0.35	0.88	0.50	17
mg hector	1.00	0.18	0.31	11
accuracy			0.57	103
macro avg	0.70	0.51	0.48	103
weighted avg	0.70	0.57	0.54	103

7.1.5.2.1 Confusion matrix:

The figure 7.8 below represents the confusion matrix for VGG16 model, which is a description of the study of prediction results on the classification problem. The amount of correct and incorrect predictions is summarized with count values and divided down by each class. We can observe the model has more setbacks than the VGG16 model as this model has misclassified all mg hector class as Maruti Brezza and most Bolero class are misclassified as Scorpio.

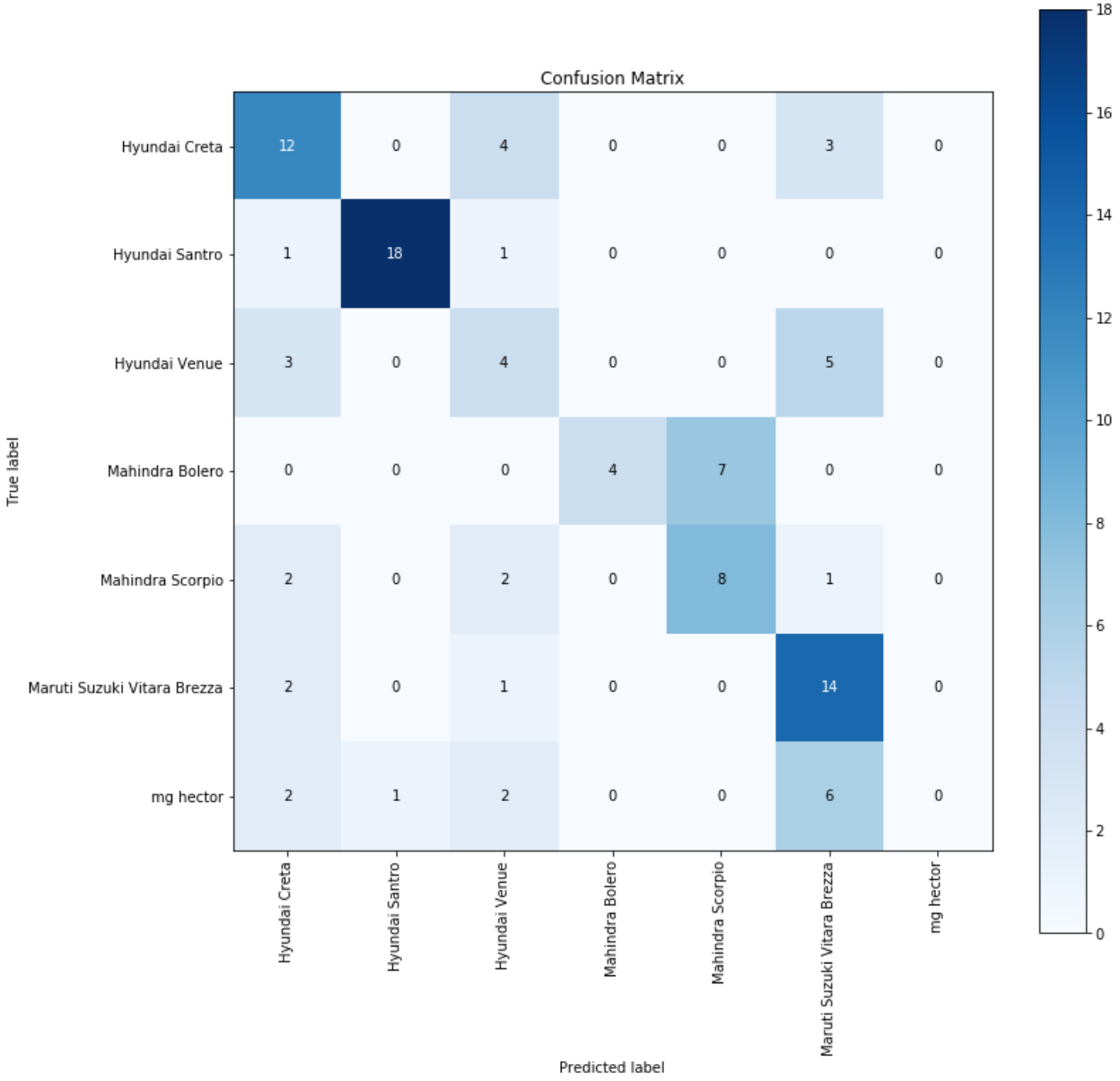


Figure:7.8

7.1.5.2.2 Precision-Recall curve:

A precision-recall curve is a plot of the precision (y-axis) and the recall (x-axis) for different thresholds. The curves with greater area under them imply a good classifier. Since we find 2 two classes some what near the threshold there is room for improvement for those classes. A model with high precision and recall is desired.

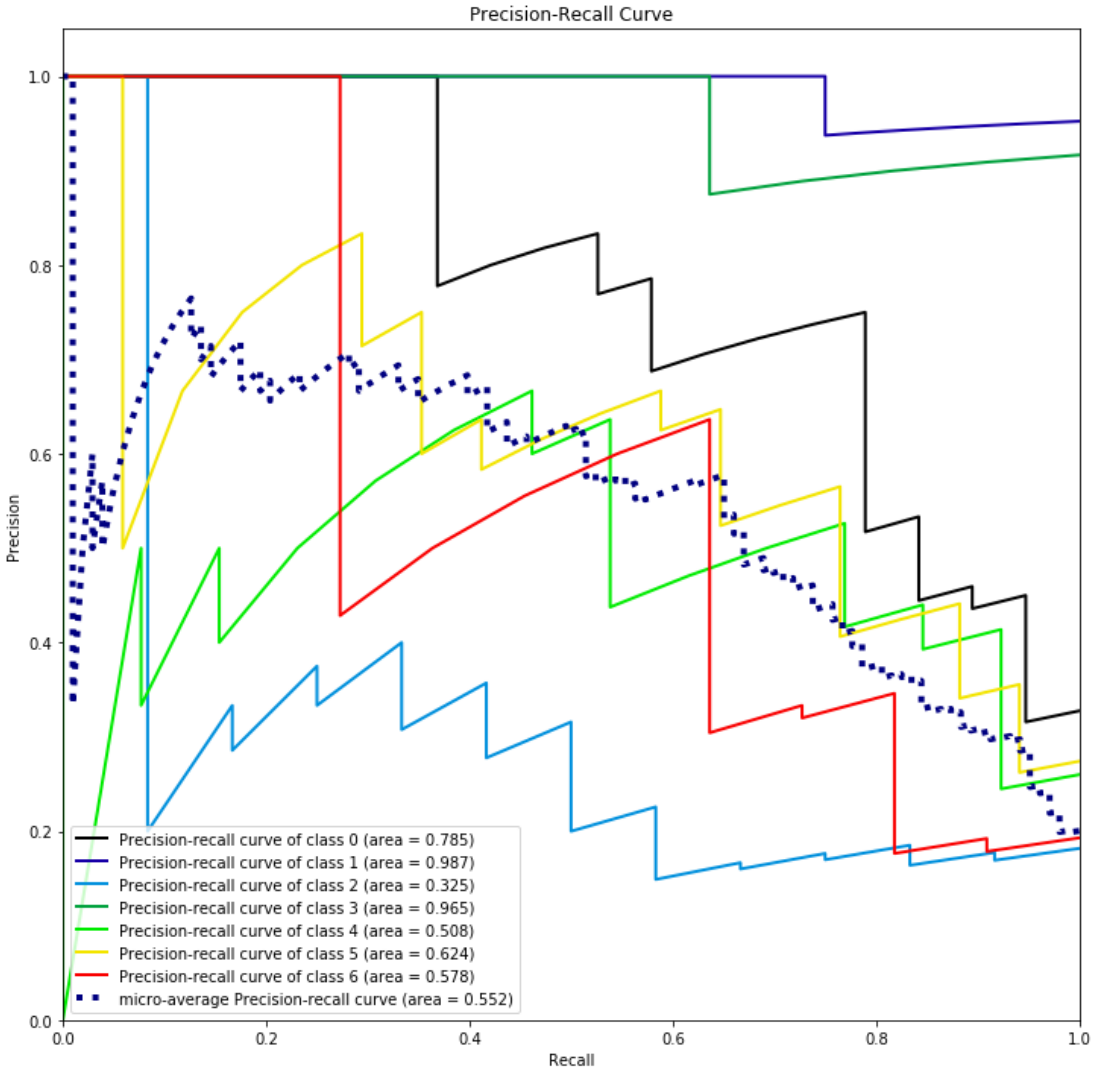


Figure:7.9

7.1.5.2.3 ROC curve:

The below figure 7.10 is a plot of the false positive rate vs. the true positive rate on x and y axis for various candidate threshold values between 0.0 and 1.0. In other words, it plots the false alarm rate vs. the hit rate.

True Positive Rate = TPs / (TPs + FNs)

False Positive Rate = FPs / (FPs + TNs)

The points to be noted are

- A model that randomly guesses the label will result in the black line and our desired model is the one that has a curve above this black line.
- A ROC that is away from the black line is the best one.

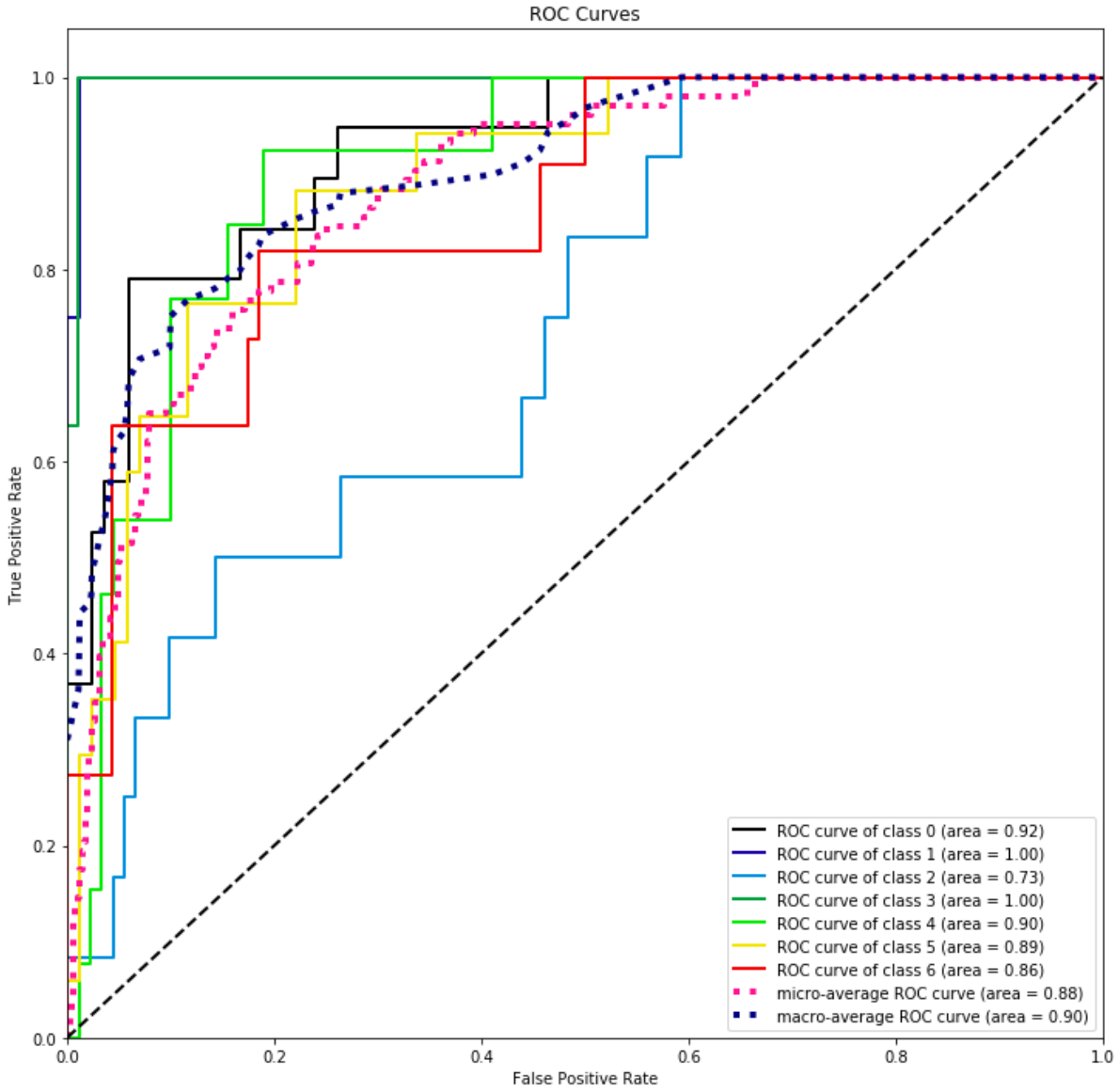


Figure:7.10

8.CONCLUSION

Vehicle make and model recognition will play a crucial role in the surveillance field. After comparing the two models we had the following results. The VGG16 was the largest in size with size of 312mb and highest accurate in term of metrics of the two with top accuracy of 88%. The InceptionV3 has a size of 118mb with mediocre performance in terms of metrics with top accuracy of 82%.

So, based on our requirements the choice of topology and framework will differ.

The requirements include Time to train, Size of model, Inference speed, Acceptable accuracy. In addition to answering a few questions about image recognition with small datasets, the work completed still has some room to progress.

9.REFERENCES

[1] Rahati, S., Moravejian, R., Kazemi, E.M. and Kazemi, F.M., 2008, April. Vehicle recognition using contourlet transform and SVM. In Fifth International Conference on Information Technology: New Generations (itng 2008) (pp. 894-898). IEEE.

[2] Chen, Z., Ellis, T. and Velastin, S.A., 2012, September. Vehicle detection, tracking and classification in urban traffic. In 2012 15th International IEEE Conference on Intelligent Transportation Systems (pp. 951-956). IEEE.

[3] Wen, X., Shao, L., Fang, W. and Xue, Y., 2014. Efficient feature selection and classification for vehicle detection. IEEE Transactions on Circuits and Systems for Video Technology, 25(3), pp.508-517.

[4] Arróspide, J. and Salgado, L., 2014. A study of feature combination for vehicle detection based on image processing. The Scientific World Journal, 2014.

[5] Wang, H., Cai, Y. and Chen, L., 2014. A vehicle detection algorithm based on deep belief network. The scientific world journal, 2014.

[6] He, D., Lang, C., Feng, S., Du, X. and Zhang, C., 2015, August. Vehicle detection and classification based on convolutional neural network. In Proceedings of the 7th International Conference on Internet Multimedia Computing and Service (pp. 1-5).

[7] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R. and LeCun, Y., 2013. Overfeat: Integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229.

Appendix Sample Code:

Importing Packages:

```
#Import some packages to use
import cv2
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import glob
import shutil
#To see our directory
import os
import random
import gc   #Gabage collector for cleaning deleted data from memory


stolen_cars_path = "unzipped_folder/most_stolen_cars"

display_images(stolen_cars_path)
```

Splitting Data

```
import math

import re

import sys

#Train and Test Set Variables

train_val_test_ratio = (.7,.1,.2) # 70/10/20 Data Split

test_folder = 'Dataset/test/'

train_folder = 'Dataset/train/'

val_folder = 'Dataset/val/'
```

```
file_names = os.listdir('unzipped_folder/most_stolen_cars')
#Remove Existing Folders if they exist
for folder in [test_folder, train_folder, val_folder]:
    if os.path.exists(folder) and os.path.isdir(folder):
        shutil.rmtree(folder)
#Remake Category Folders in both Train and Test Folders
for category in file_names:
    os.makedirs(test_folder + category)
    os.makedirs(train_folder + category)
    os.makedirs(val_folder + category)
#Split Data by Train Ratio and copy files to correct directory
for idx, category in enumerate(file_names):
    file_list = os.listdir(stolen_cars_path + '/' + category)
    train_ratio = math.floor(len(file_list) * train_val_test_ratio[0])
    val_ratio = math.floor(len(file_list) * train_val_test_ratio[1])
    train_list = file_list[:train_ratio]
    val_list = file_list[train_ratio:train_ratio + val_ratio]
    test_list = file_list[train_ratio + val_ratio:]
    for i, file in enumerate(train_list):
        shutil.copy(stolen_cars_path + '/' + category + '/' + file, train_folder + '/' + category + '/' +
file)
    sys.stdout.write('Moving %s train images to category folder %s' % (len(train_list), category))
    sys.stdout.write('\n')
    for i, file in enumerate(val_list):
```

```
        shutil.copy(stolen_cars_path + '/' + category + '/' + file, val_folder + '/' + category + '/' + file)
    sys.stdout.write('Moving %s validation images to category folder %s' % (len(val_list), category))
    sys.stdout.write('\n')
    for i, file in enumerate(test_list):
        shutil.copy(stolen_cars_path + '/' + category + '/' + file, test_folder + '/' + category + '/' + file)
    sys.stdout.write('Moving %s test images to category folder %s' % (len(test_list), category))
    sys.stdout.write('\n')
print("Done.")
```

Building Inception Model

```
# Initialize InceptionV3 with transfer learning
base_model = applications.InceptionV3(weights='imagenet',
                                      include_top=False,
                                      input_shape=(WIDTH, HEIGHT,3))
# add a global spatial average pooling layer
x = base_model.output
x = GlobalAveragePooling2D()(x)
# and a dense layer
x = Dense(1024, activation='relu')(x)
predictions = Dense(len(train_flow.class_indices), activation='softmax')(x)

# this is the model we will train
model = Model(inputs=base_model.input, outputs=predictions)
```

```
# first: train only the top layers (which were randomly initialized)

# i.e. freeze all convolutional InceptionV3 layers

for layer in base_model.layers:

    layer.trainable = False

# compile the model (should be done *after* setting layers to non-trainable)

model.compile(optimizer=optimizers.Adam(lr=0.001),

            metrics=['accuracy', 'top_k_categorical_accuracy'], loss='categorical_crossentropy')

model.summary()
```

Training Model

```
import math

top_layers_file_path="top_layers.iv3.hdf5"

checkpoint = ModelCheckpoint(top_layers_file_path, monitor='loss', verbose=1,
save_best_only=True, mode='min')

tb = TensorBoard(log_dir='./logs', batch_size=val_flow.batch_size, write_graph=True,
update_freq='batch')

early = EarlyStopping(monitor="loss", mode="min", patience=5)

csv_logger = CSVLogger('./logs/iv3-log.csv', append=True)

history = model.fit_generator(train_flow,

                              epochs=6,

                              verbose=1,

                              validation_data=val_flow,

                              validation_steps=math.ceil(val_flow.samples/val_flow.batch_size),

                              steps_per_epoch=math.ceil(train_flow.samples/train_flow.batch_size),

                              callbacks=[checkpoint, early, tb, csv_logger])
```