

Vehicle Model Identification using Neural Network Approaches

By

UTTAM MOJUMDER

11110005

TOQI TAHAMID SARKER

12201007

GULNAHAR MAHBUB MONIKA

12201089

NURUL AMIN RATUL

13201033



Inspiring Excellence

Department of Computer Science & Engineering
BRAC UNIVERSITY

Supervised by: MOIN MOSTAKIM

Thesis report submitted to the BRAC University in accordance with the requirements of the degree of BACHELOR IN COMPUTER SCIENCE AND ENGINEERING in the Dept. of Engineering & Computer Science.

Submitted On:
14TH DECEMBER, 2016

ABSTRACT

Our world is going through a constant phase of growth and advancement where the manufacture of vehicles has increased exponentially. Vehicles of different brands with different features and models are vastly available in all over the world. Along with the fact that vehicles are now a basic need, every individual is now able to afford a vehicle of their choice and status. Consequently, misdemeanors such as theft, accidents, damage done, relating to automobiles has also increased over the years. Identifying a specific model vehicle among these several brands of vehicles can be considered difficult. Our main goal is to find the details of a specific model of a transport from several unknown automobile's datasets. Our system will help to identify a vehicle and its model using still pictures of any brand of car. We hope that in future we can extend it to a more advanced identifying system which can be used by the government to reduce all forms of transgressions towards vehicles.

ACKNOWLEDGEMENT

Before starting to write this paper, we would like to express my utmost gratitude to Almighty Allah who gave us the zeal, determination, strength and intelligence to complete our thesis. We want to thank our parents for their support and our respected faculties and dear classmates for their constant support and motivation. Most importantly, we would like to thank our supervisor MoinMostakim Sir for his consistent supervision, guidance and unflinching encouragement in accomplishing our work and help us to execute our idea with great success.

Author's Declaration

We, hereby declare that this thesis is based on the results found by ourselves. Materials of work found by other researcher are mentioned by reference. This thesis, neither in whole or in part, has been previously submitted for any degree.

SIGNATURE OF THE AUTHORS:

.....

UTTAM MOJUMDER

11110005

.....

TOQI TAHAMID SARKER

12201007

.....

GULNAHAR MAHBUB MONIKA

12201089

.....

NURUL AMIN RATUL

13201033

SIGNATURE OF THE SUPERVISOR:

.....

MOIN MOSTAKIM

LECTURER

DEPT. OF COMPUTER SCIENCE, BRAC UNIVERSITY

Table of Contents

List of Figure	ix
List of Table	xi
Chapter 1 Introduction	1
1.1. Motivation	1
1.2. Objectives	1
Chapter 2 Literature Review	2
2.1. Overview	2
2.2. Previous work	2
2.3. System implementation.....	3
2.4. Different CNN models.....	5
2.4.1. VGG-16.....	5
2.4.2. VGG-19.....	5
2.4.3. Inception	7
2.4.4. Deep Residual Network.....	8
Chapter 3 Work and Analysis.....	11
3.1. Dataset extraction and Preprocessing.....	11
3.2. Data Augmentation	12
3.2.1. Training Process and Image augmentation	13
3.2.2. Testing Process and Image augmentation.....	14
3.3. Fine-tune pre-trained Networks	14
3.4. Prediction of CNN models	15
3.4.1. Using ConvNet model VGG-16.....	15
3.4.2. Using ConvNet model VGG-19.....	19
3.4.3. Using ConvNet model Inception V3.....	23
3.4.4. Using ConvNet model Residual Network.....	28
3.4.5. Final prediction.....	34
3.4.6. Hardware Implementation and Toolkits	37
3.4.7. Limitations.....	38

Chapter 4 Conclusion	40
4.1. Conclusion	40
4.2. Future Work	40
BIBLIOGRAPHY	42

List of Figure

Figure 2.1 Process of System implementation	4
Figure 2.2 Logo of different brand of cars	4
Figure 2.3 VGG-16	5
Figure 2.4 VGG-19	6
Figure 2.5 Difference between the VGG-16 and VGG-19	6
Figure 2.6 GoogleNet model architecture [12]	7
Figure 2.7 Inception-v3 complete architecture [12]	8
Figure 2.8 Comparison between left : VGG-19 model, mid : a plain network with 34 parameter layers and right : a residual network with 34 parameter layers. [13]	9
Figure 2.9 Residual learning: a building block [13]	10
Figure 3.1 Bounding box	11
Figure 3.2 Work and Analysis	12
Figure 3.3 Data Augmentation.....	14
Figure 3.4 Layer visualization of convolutional layer of VGG16	16
Figure 3.5 Layer visualization of maxpooling layer of VGG16	16
Figure 3.6 VGG-16 model accuracy graph.....	17
Figure 3.7 VGG-16 model loss function graph	17
Figure 3.8 Testing output of VGG-16 with accuracy of different models of car .	18
Figure 3.9 Layer visualization of convolutional layer of VGG19	20
Figure 3.10 Layer visualization of maxpooling layer of VGG19.....	20
Figure 3.11 VGG-19 model accuracy graph	21
Figure 3.12 VGG-19 model loss function graph.....	21
Figure 3.13 Testing output of VGG-19 with accuracy of different models of car	22
Figure 3.14 Layer visualization of convolutional layer of Inception V3	24
Figure 3.15 Layer visualization of maxpooling layer of Inception V3.....	24
Figure 3.16 Inception V3 model accuracy graph.....	25
Figure 3.17 Inception V3 model loss function graph	26
Figure 3.18 Testing output of Inception V3 with accuracy of different models of car	27
Figure 3.19 Layer visualization of convolutional layer of ResNet50	29
Figure 3.20 Layer visualization of maxpooling layer of ResNet50.....	29
Figure 3.21 ResNet50 model accuracy graph.....	30
Figure 3.22 ResNet50 model loss function graph	31
Figure 3.23 Testing output of ResNet50 with accuracy of different models of car	32
Figure 3.24 Train accuracy of the four Models.....	35
Figure 3.25 Train loss of the four models	35
Figure 3.26 Test accuracy of the four models	35

Figure 3.27 Test loss of the four models.....	36
Figure 3.28 NumPy.....	37
Figure 3.29 Matplotlib	38
Figure 3.30 Theano and Keras	38
Figure 3.31 Python	38

List of Table

Table 2.1 Dataset Information	4
Table 3.1 Prediction after testing phase of VGG-16	19
Table 3.2 Prediction after testing phase of VGG-19	23
Table 3.3 Prediction after testing phase of Inception V3.....	28
Table 3.4 Prediction after testing phase of ResNet50.....	33
Table 3.5 Table of predicted output	34
Table 3.6 Bar chart of final prediction	36

Chapter 1

Introduction

1.1. Motivation

In machine learning, Convolutional Neural Network (CNN) provide a wide application in image and video recognition, recommender systems and natural language processing. CNN or ConvNet consists of multi-layered based neural network where there are huge number of connections between neurons [1,9,8]. CNN is an adaptive, intelligent self-learning model in which the network is trained using a large data set with respective assigned or random weights, this allows the system to learn and develop its classification head and then later in testing phase make assumptions of images consisting of cars, humans, animals, etc. appropriately. So this is a highly advanced artificial intelligence system which can make human like assumptions in digital world.

1.2. Objectives

In our project we have used the Stanford car dataset model [2], our aim is to compare different CNN models (such as VGG-16, VGG-19, Residual Neural Network, Inception-V3) and their respective output using this dataset [7,14,15]. We compared the different CNN models to see which gave better predictions (i.e. output results) and has the lowest error percentage (i.e. loss function). After this, we can now get a clear vision as to which would be the best approach for vehicle model identification.

Chapter 2

Literature Review

2.1. Overview

Initially our approach was made using Residual Network (ResNet50) model alongside [12,14] which we have applied our desired design over on Inception's GoogleNet, Inception V3, VGG-16 and VGG-19. GoogleNet Inception model is one which deals with increased width and depth of the neural networks. Multiple convolutional layers compile at the pooling layer and then gives out the final output at the fully connected layer. The VGG-16 and the VGG-19 models work in a very similar manner. We use very small 3×3 filters in both convolutional layers, where the VGG-19 is one step deeper than the VGG-16. ResNet50 is a deep learning model in which adding more layers in the convolutional layer concludes to better output result, this is possible because in ResNet50 something new is always learned in every layer while keeping all the information from the previous layers stored and hence ResNet50 is able to handle vanishing gradient problems in deep layers.

2.2. Previous work

To achieve a clear vision of Convolutional Neural Network we had to do research on previous work related to this topic, some of the most important papers that we used as references are discussed below,

The paper we had read they have used an effective pre-training strategy to improve basic Convolutional Neural Network (CNN) where the computational cost of training stages in the layers of CNN are reduced and the classification performance is enhanced [1]. This paper gave us an initial understanding of how the CNN and its layers work. Another paper, which discussed how the CNN might not suffice to provide proper classification for all forms of Fine Grained Image Classification (FGIC) task as the position, view-point, etc. can affect the result. They proposed a novel data augmentation approach by identifying inherent and easily annotated hyper-classes in the fine-grained data and accumulating a large amount of similar images labeled by hyper-classes. The augmented data of hyper-class can generalize feature learning by integrating multi-task learning into a deep CNN model [4]. This shows the changes that can

be made to a basic CNN model to reach a desired output result. Further for more knowledge we researched another thesis paper, where they discuss three popular CNN structures for their project, they are: CaffeNet, GoogleNet, and VGGNet. Here, they have used these CNN models and further improved the classification performance by using novel pre-training and data enhancement strategy[15]. We learn how different models of CNN work and how making addition to the layers of the neural network can change a model's output and how it can be manipulated. We learned from another thesis paper that how they have used co-segmentation and alignment, which they combined in a discriminative mixture on fine grained recognition that uses no part annotations at training time[3]. Here we see the use of bounding box method and co-segmentation method.

For our project we have used the Stanford car dataset model which consists of 16,185 images of 196 classes of different set of cars, most of the research papers have used this dataset to conduct their approach.

2.3. System implementation

The system we have used was implemented using Stanford car dataset [2] which contains 16,185 images of 196 classes of cars. The data was already split into 8144 training images and 8041 testing images, where each class was split roughly in a 50-50 split. We gathered this dataset for our system's purpose where we made necessary changes such as cropping the raw image using the bounding box parameter that was given, then we divided each car brand and their respective car models into different folders (the labels for both training and testing images were given). After that the implementation of algorithm is done, here we have used Theano [17] as the machine learning library and Keras[5] as the wrapper library. Theano has a vast library which can be easily accessed using Keras's methods, this is done for simplicity purpose and time shortage. The algorithm then pre-processes data where image augmentation is done to reach a generalized form. We then run our different models of Convolutional Neural Network (CNN), here each of the different models used (Inception V3, Residual Network, VGG-16, and VGG-19) follow the basic CNN format (i.e. ConvNets followed by Max-pooling layers and then finally Fully Connected Layers) with different parameters and functionalities. We train our augmented data's in our models and obtain different predicted outputs, we then compare our results and reach conclusion. A diagram of our workflow is given below:



Figure 2.1 Process of System implementation

	Training Data	Testing Data
Car Brands	45	45
Car Models	196	196
No. of Images	8144	8041

Table 2.1 Dataset Information

Below is the total list of cars brand logos:



Figure 2.2 Logo of different brand of cars

2.4. Different CNN models

2.4.1. VGG-16

VGGNet is a neural network that performs very well with large data as it works with 3×3 convolution filters where the depth is 16-19 weights layers. We fix other parameters of the architecture at first, and then steadily increase the depth of the network by adding more convolutional layers, which is feasible due to the use of very small (3×3) convolution filters in all layers. As we can see in the figure below:

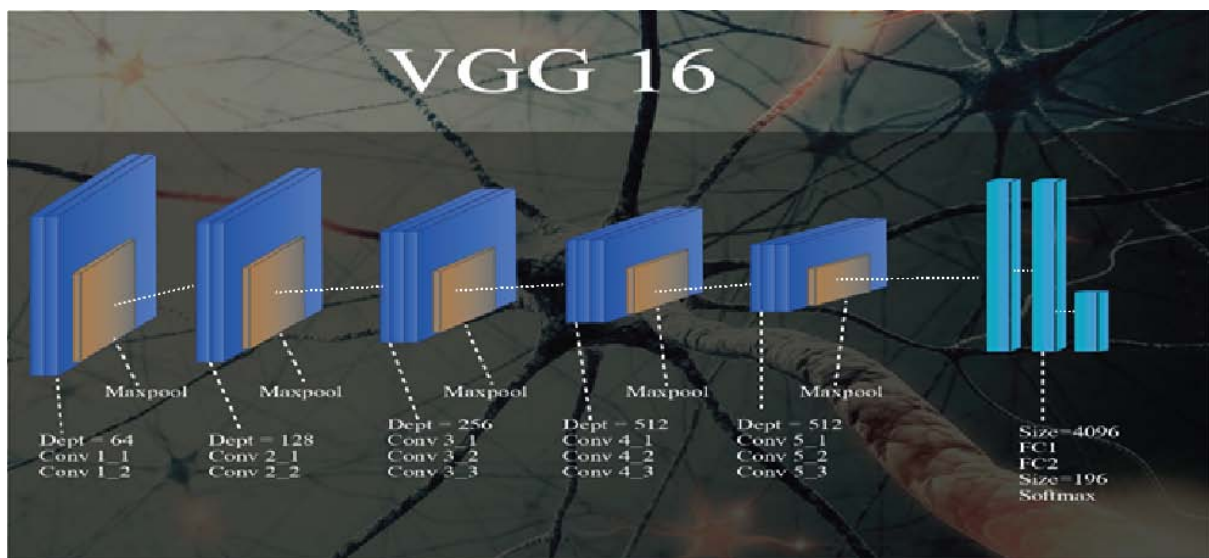


Figure 2.3 VGG-16

2.4.2. VGG-19

Similarly, VGG-19 performs in the same way as we mentioned in (2.3.3). This VGGNet deals with more deeper layers for more accurate and better outputs.

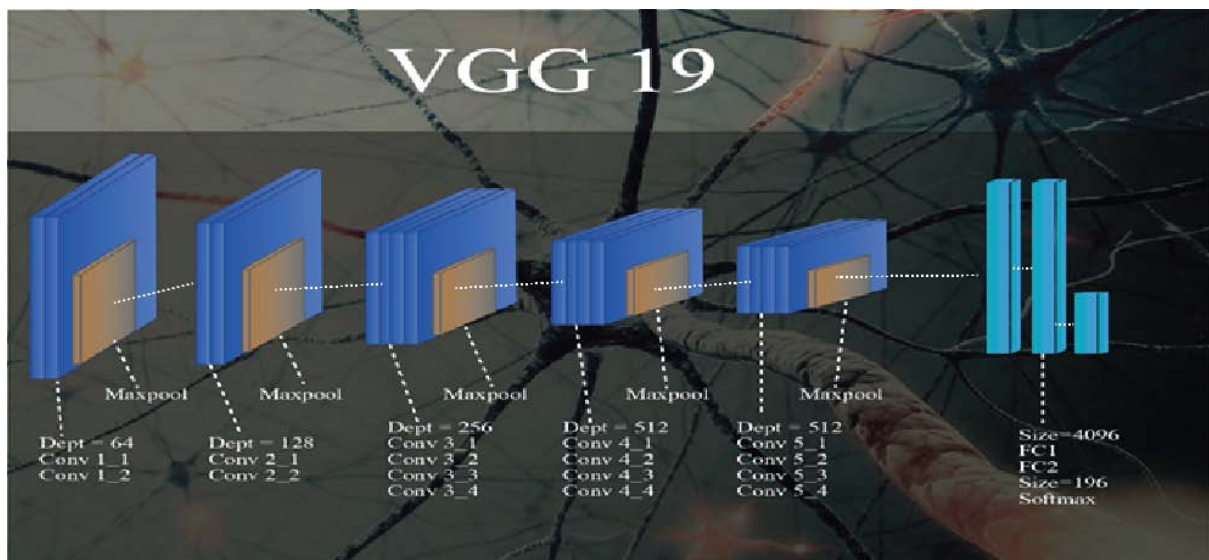


Figure 2.4 VGG-19

The difference between VGG-16 and VGG-19 is shown below with a proper diagram for better understanding:

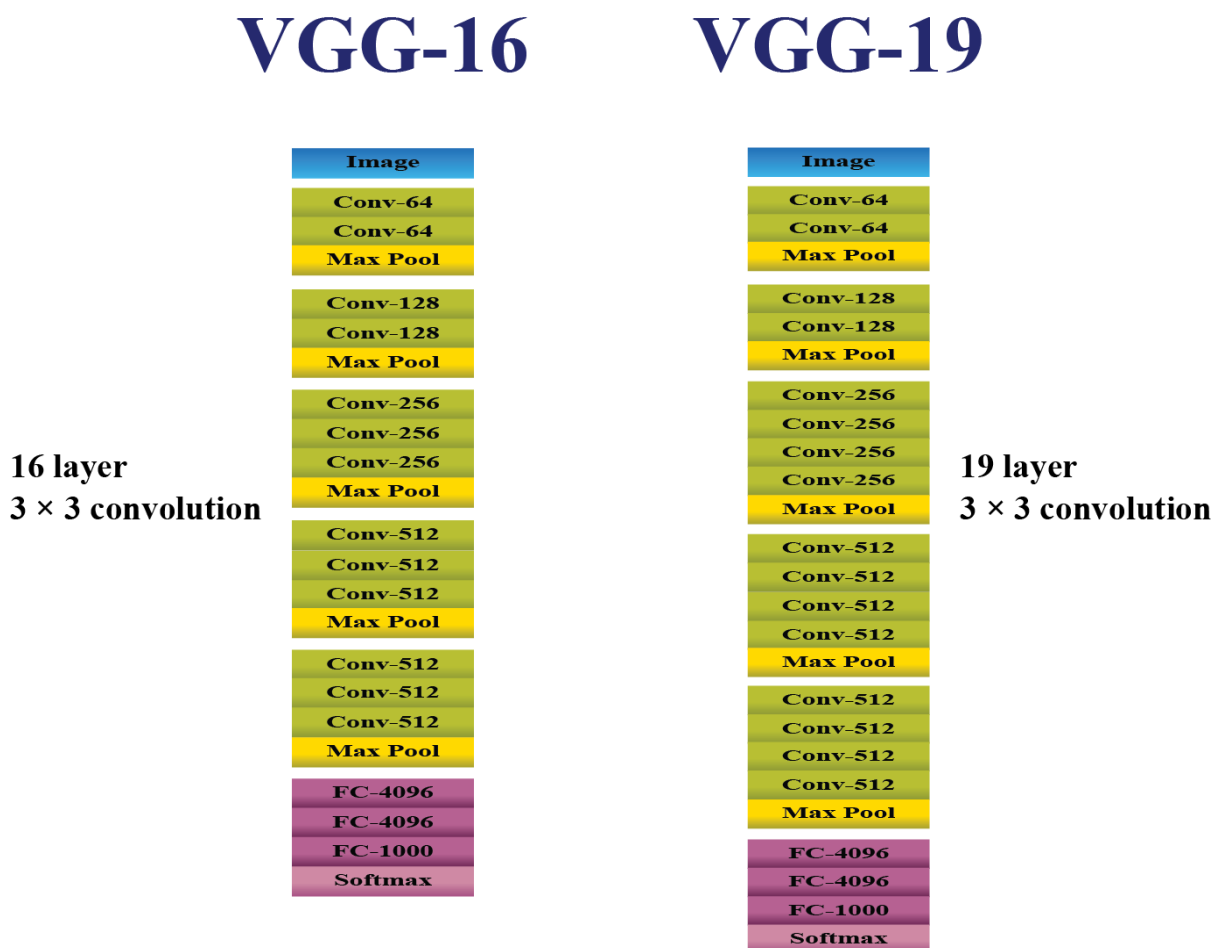


Figure 2.5 Difference between the VGG-16 and VGG-19

2.4.3. Inception

2.4.3.1. GoogleNet (Inception V1)

GoogleNet is a 22 layers deep network, the quality of which is evaluated in the context of classification and detection. Google-Net uses very few parameters and provides efficient output with less power, cost and memory use. The paper [5,12,14] we had taken reference from had the following classifiers as follows:

- An average pooling layer with 5x5 filter size and stride 3, resulting in a 4x4x512 output for the (4a), and 4x4x528 for the (4d) stage. [14]
- A 1x1 convolution with 128 filters for dimension reduction and rectified linear activation.
- A fully connected layer with 1024 units and rectified linear activation.
- A dropout layer with 70% ratio of dropped outputs. [14]

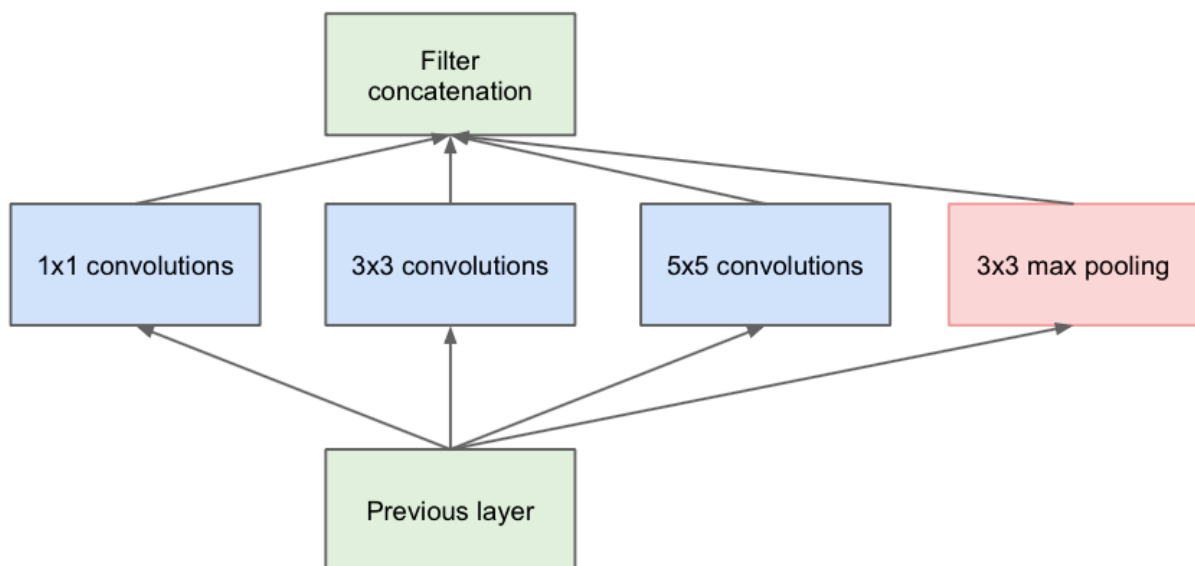


Figure 2.6 GoogleNet model architecture [12]

The module basically acts as multiple convolution filter inputs that are processed on the same input. It does also pooling at the same time. All the results are then combined. This allows the model to take advantage of multi-level feature extraction from each input. For instance, it extracts large (5x5) and local (1x1) features [12] at the same time.

2.4.3.2. Inception V3

Inception module uses the concept of adding multiple layers before pooling layer in the convolutional layer and concatenates the output of the whole process to pass it to the full connected neural network layer.

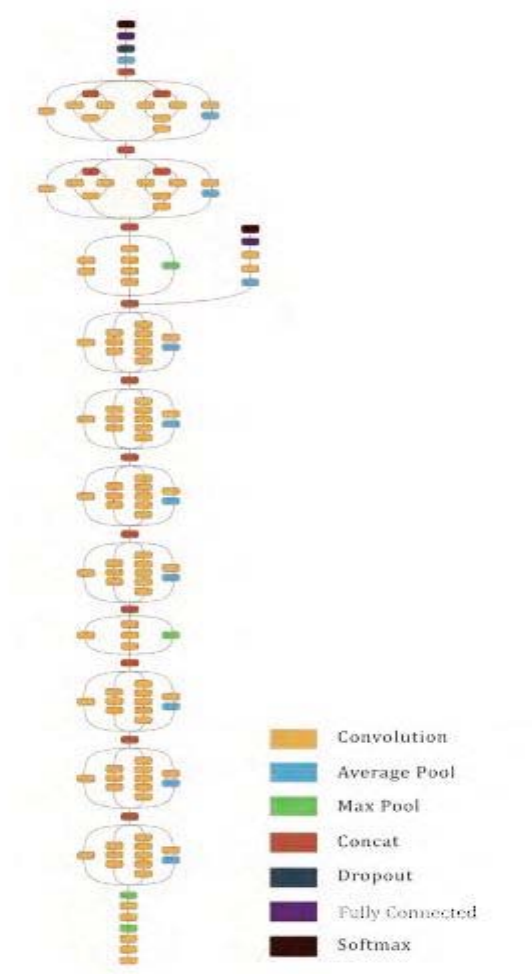


Figure 2.7 Inception-v3 complete architecture [12]

2.4.4. Deep Residual Network

Residual neural network (ResNet50) is able to provide and ease of training for networks that are subsequently deeper and hard to train. It was developed by Microsoft research which consists of 50 layers, almost 3 times deeper than a VGG network. The figure is given,

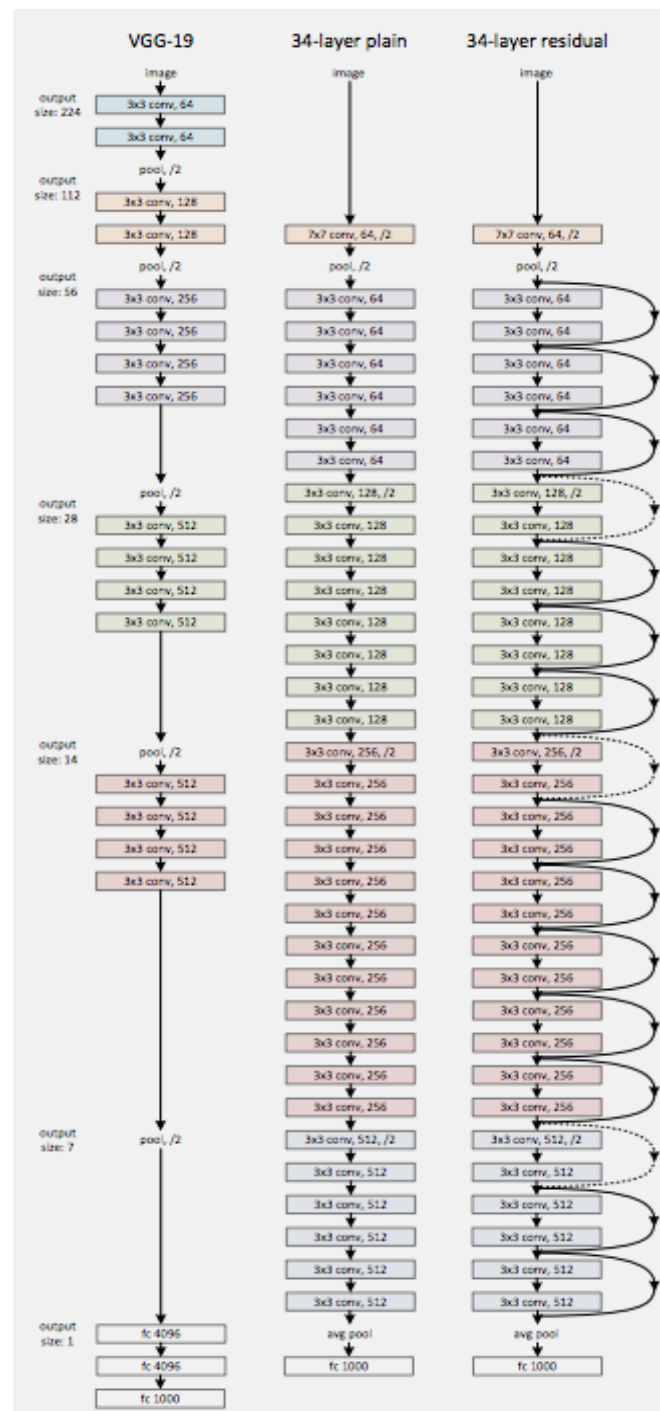


Figure 2.8 Comparison between **left:** VGG-19 model, **mid:** a plain network with 34 parameter layers and **right:** a residual network with 34 parameter layers. [13]

A block diagram representation of residual network is given below:

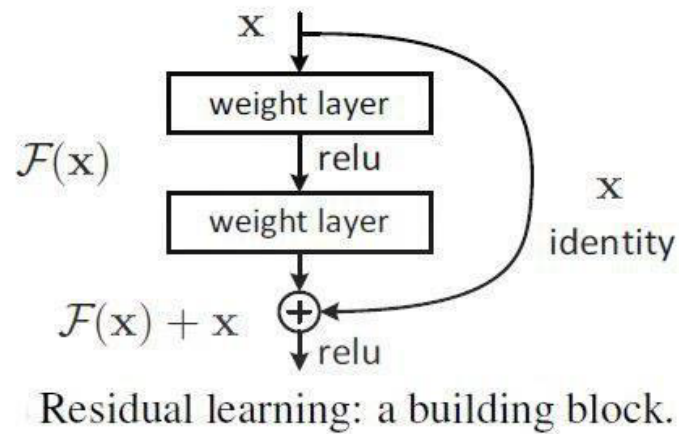


Figure 2.9 Residual learning: a building block [13]

The figure shows how a residual network proceeds from one node to another, for an input (x) which is passed onto a layer $F(x)$ gives out an output of $F(x)+x$. [13] We give an input and compute an output, before moving to the next node the residual network concatenates the new output and the previous input and then passes this concatenation to the next node as its input. This whole procedure continues throughout the whole network, where every new node/layer is aware of its predecessor.

Chapter 3

Work and Analysis

3.1. Dataset extraction and Preprocessing

The dataset used is obtained from the Stanford car dataset [2]. The dataset has been generated by supplying pictures from search engines like Google, Bing, etc. to Amazon mechanical Turk system where noise free pictures are given out as output pictures which are necessary for training in executing different convolutional models. This system has been used to get feasible output from our proposed models, normal data's could have been used but those would have had high level of discrepancies and noise margins which we have avoided for our training purposes in our convolutional models. In the whole of the dataset, 196 models of different cars have been used in which 8041 are testing set and 8144 are training set, cars such as Sedans, SUVs, Coupes, Convertibles, Pick-ups, Hatch-bags and Station Wagons.

The data's in Stanford car dataset [2] has images which have been cropped using bounding box method. The data set has fixed points of bounding box using which we have achieved our desire form of pictures. An example of bounding box is given below:

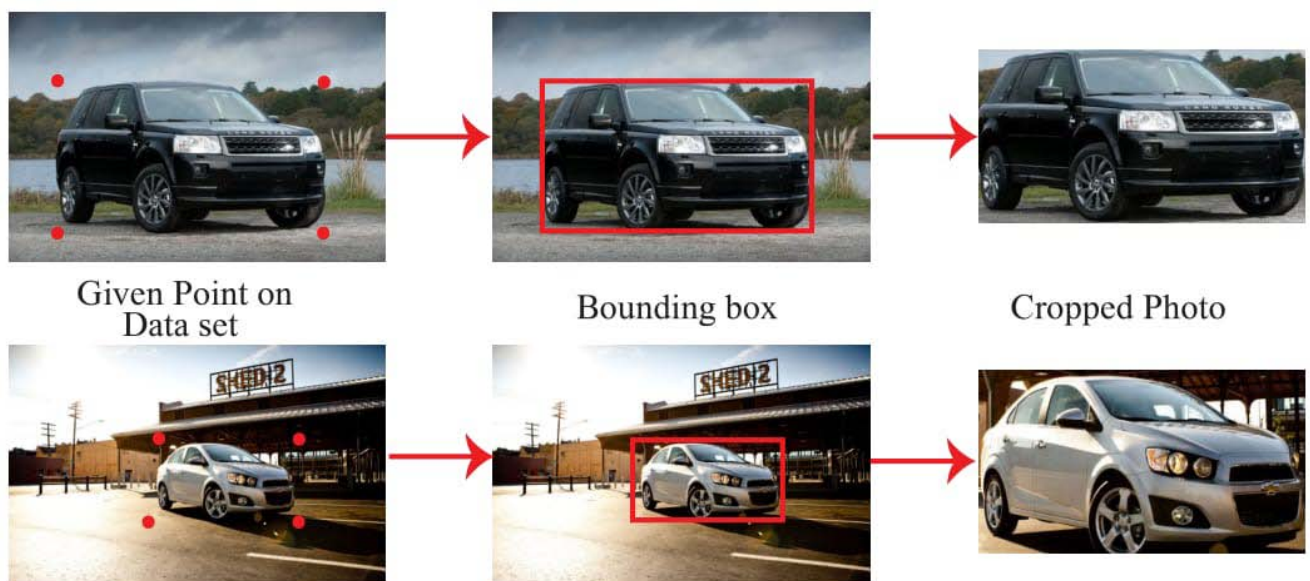


Figure 3.1 Bounding box

3.2. Data Augmentation

The models we will be testing our data on (i.e. Residual Network, Inception V3, VGG-16 and VGG-19), data augmentation for all of these models are the same. The process we are following works by making multiple data images of one single input data image.

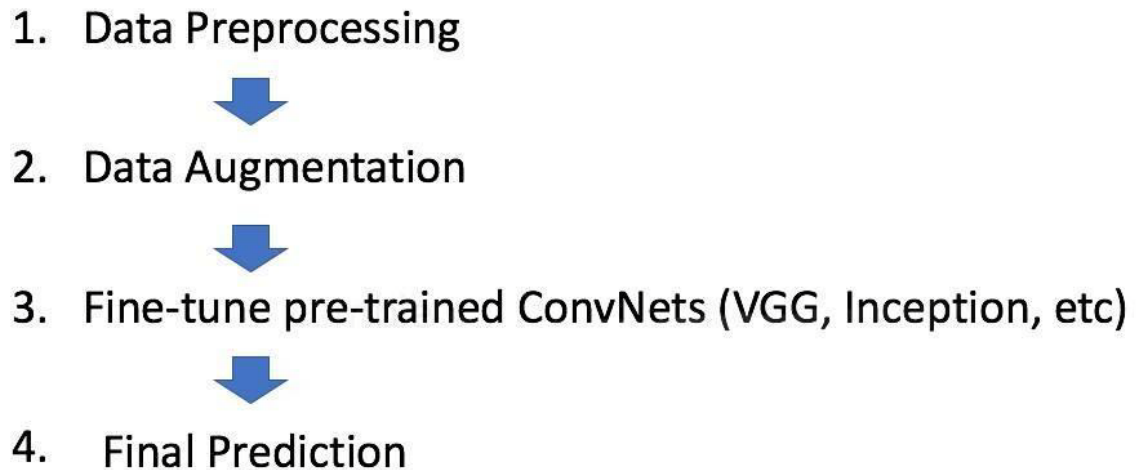


Figure 3.2 Work and Analysis

The creation of these multiple images are different for testing and training, the functions we have used for augmenting our image data are as follows:

- **Rescale:** This method multiplies the values of any data before any additional processing. We know the original images consists of 0-255 in RGB coefficients, which is very high for our models to process. Because of which we scaled our images with $1./255$ factors.
- **Rotation range:** This method randomly rotates every image in a range between values of degree (0-180).
- **Width shift range:** Here it randomly translates images in vertical ranges.
- **Height shift range:** Here it randomly translates images in horizontal ranges.
- **Horizontal flip:** It randomly flips half of the image horizontally.
- **Zoom range:** This method randomly zooms in inside images.
- **Channel shift range:** Transformation of the channel.

- **Fill mode:** This method is a filling strategy which helps to fill the newly created pixels which apparently can appear after any kind of rotation or shifting i.e width or height.

In our training process we have used all the functions mentioned above, for our testing process we have used only a few of these functions which are mentioned below:

3.2.1. Training Process and Image augmentation

In order to train our pre-trained ConvNets, we have used Stochastic Gradient Descent (SGD) [15] learning optimizer i.e. our primary learning machine of our neural network. The learning rate of our program was set at 0.001 rate, this is because we aimed to achieve a fine tuned network with a sound structural program body. After every batch of training sets, our learning rate is decreased by a very small factor. This is due to Decay application (our decay rate was 0.000001), whose only function is to decrease the learning rate and provide better training of our pre-trained dataset. Along with this cross-entropy function computes all the loss function of the network. We have also included a momentum update (our momentum update was 0.9), which leads to better convergence of the network. The way momentum update works is by applying a force on the particles (i.e. ConvNet units) which relates to the gradient of the loss function. If the loss function is high the momentum is slow and vice-versa i.e. Momentum update is proportion to the negative function of the gradient loss [Momentum]. Nesterov momentum (for our program it was set to TRUE) is another function which tends to produce a stronger convergence. This momentum system prioritizes on a “look-ahead” position i.e. Nesterov momentum does not evaluate gradient at the current position instead it focuses on the fact that our momentum will carry us to a forward position and evaluates the gradient at that position [Momentum].

- Rescale-1./255
- Rotation range=45
- Width shift range=0.25
- Height shift range=0.25
- Horizontal flip=True
- Zoom range=0.5
- Channel shift range=0.5
- Fill mode='nearest'.

3.2.2. Testing Process and Image augmentation

In testing phase, we have used our ConvNet dataset from ImageNet as input and through forward pass we have passed the data to the final output layer where after using the softmax function to calculate the loss function a prediction is given out as result.

- Rescale= $1./255$,
- Rotation range=45,
- Width shift range=0.25,
- Height shift range=0.25,
- Horizontal flip=True



Figure 3.3 Data Augmentation

3.3. Fine-tune pre-trained Networks

To satisfy our required goals and vision we have used four set of models – VGG-19, VGG-16, Inception-GoogleNet and Residual Network. In order to execute our program we have used a simple neural based library called Keras built on top of theano, to achieve fast implementation of our design. Usually a Café based neural network library can be used which gives the access of processing multiple ConvNet models but it has a complex structural model which would require long

time for us to learn hence we have used Keras, which has a very simple and compact structure with a vast library. We have used pre-trained ConvNets from ImageNet [], because using a ConvNet and training it from scratch would not suffice to give out a good output due to a small dataset. Using the mentioned ConvNet models we fine-tune our augmented dataset. The processing and augmentation for our proposed design is the same for all the ConvNet models [3,2,4]. After the proper structuring of our dataset we go through further processing inside the deep layers of Convolutional network using our different set of models, their feature and processing structure are stated as given below:

3.4. Prediction of CNN models

3.4.1. Using ConvNet model VGG-16

We have used the basic VGG-16 model to execute our program, which has all the traditional functions of a normal VGG-16 ConvNet model except the Fully-Connected layer. In the fully connected layer of our VGG-16 model we have added a Dropout function because our dataset is limited and there is a highly probability of overfitting taking place which is not feasible for our program. The way Dropout works is that, it randomly drops units from the neural network during training phase as a result co-adaptation of units are prevented [Dropout]. In our system we have set the parameter for the Dropout function in a way where half of the numbers of units are dropped at every fully connected layer except the prediction layer. The output of these fully connected layers produce 1000 class prediction, since we have used the data weights of ImageNet. But our design consists of 196 classes i.e. our input dataset consists of 196 classes and according to CNN our output should consist of 196 classes as well. But that is not happening in our fully connected layers, as a result, in our prediction layer i.e. our final fully connected layer we have removed the excess classes produced and implemented our own desired output system.

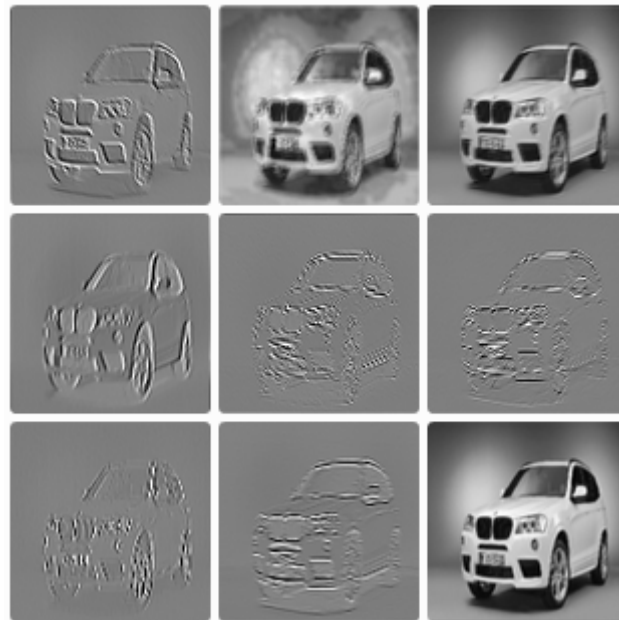


Figure 3.4 Layer visualization of convolutional layer of VGG16



Figure 3.5 Layer visualization of maxpooling layer of VGG16

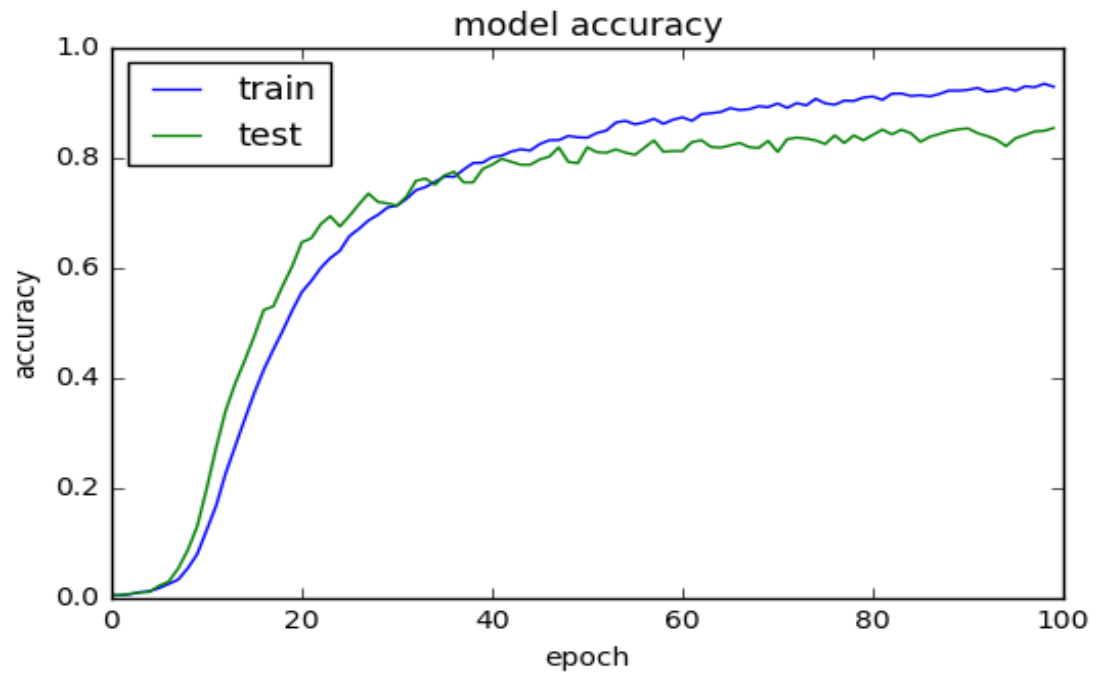


Figure 3.6 VGG-16 model accuracy graph

The training accuracy was 86.94% and the testing accuracy was 83.05%.

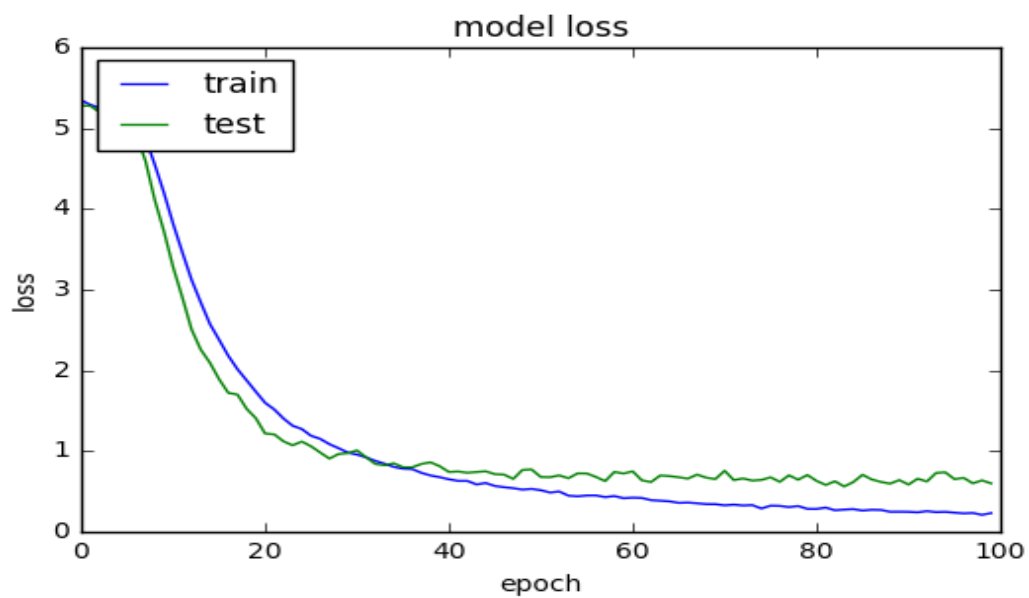


Figure 3.7 VGG-16 model loss function graph

The loss function of trained data was 45.85% and the tested data was 64.14%.

Work and Analysis

BMW M3 Coupe 2012 	BMW M3 Coupe 2012 	BMW M5 Sedan 2010 	BMW M6 Convertible 2010 	BMW 3 series Wagon 2012 	BMW ActiveHybrid 5 Sedan 2012 
Testing car 1	1st Pre: 100 %	2nd Pre: 00 %	3rd Pre: 00%	4th Pre: 00%	5th Pre: 00%
Bentley Continental Supersports Conv. Convertible 2012 	Bentley Continental Supersports Conv. Convertible 2012 	Bentley Continental GT Coupe 2012 	Bentley Continental Flying Spur Sedan 2007 	BMW 1 Series Convertible 2012 	MINI Cooper Roadster Convertible 2012 
Testing car 2	1st Pre: 100 %	2nd Pre: 00 %	3rd Pre: 00 %	4th Pre: 00 %	5th Pre: 00 %
Dodge Magnum Wagon 2008 	Dodge Magnum Wagon 2008 	Dodge Magnum Wagon 2008 	Audi TT RS Coupe 2012 	Chevrolet Cobalt SS 2010 	Ford Expedition EL SUV 2009 
Testing car 3	1st Pre: 99.99 %	2nd Pre: 0.01%	3rd Pre: 00%	4th Pre: 00%	5th Pre: 00%
Chrysler Crossfire Convertible 2008 	67.21% Ford GT Coupe 2006 	Ford Mustang Convertible 2007 	Dodge Charger SRT-8 2009 	Dodge Charger Sedan 2012 	Acura TL Type-S 2008 
Testing car 4	1st Pre: 67.21 %	2nd Pre: 17.53 %	3rd Pre: 2.45 %	4th Pre: 1.82 %	5th Pre: 1.38 %
Mercedes-Benz E-Class Sedan 2012 	Mercedes-Benz E-Class Sedan 2012 	Mercedes-Benz C-Class Sedan 2012 	BMW 3 Series Wagon 2012 	Chrysler Sebring Convertible 2010 	Ford Expedition EL SUV 2009 
Testing car 5	1st Pre: 100 %	2nd Pre: 00%	3rd Pre: 00%	4th Pre: 00 %	5th Pre: 00 %

Figure 3.8 Testing output of VGG-16 with accuracy of different models of car

The diagram shows the predicted output and their respective percentages.


Testing Car	Given best acuracy	Acuracy %	Prediction
		76.36%	NO
Audi TT Hatchback 2011	Audi TT RS Coupe 2012		
		100%	YES
BMW X3 SUV 2012	BMW X3 SUV 2012		
		100%	YES
Chevrolet Cobalt SS 2010	Chevrolet Cobalt SS 2010		
		99.94%	YES
Chevrolet Malibu Hybrid Sedan 2010	Chevrolet Malibu Hybrid Sedan 2010		
		51.04%	No
Dodge Ram Pickup 3500 Crew Cab 2010	Ford Ranger SuperCab 201110		

Table 3.1 Prediction after testing phase of VGG-16

Here, we see that 3 out of 5 predictions made were accurate.

3.4.2. Using ConvNet model VGG-19

VGG-19 model is similar to that of VGG-16 with many common functionalities. But VGG-19 leads to faster convergence compared to VGG-16, since it consists of deeper ConvNet layers. That means the loss function here is comparatively lower but the learning rate is slower since it has deeper layers.

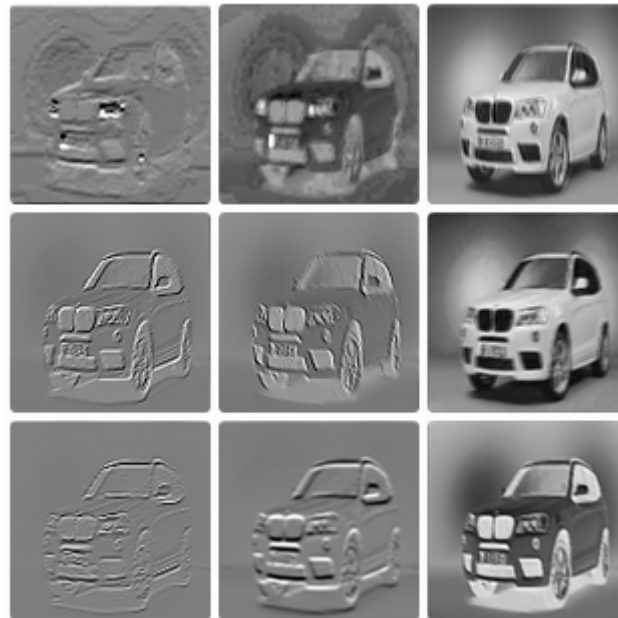


Figure 3.9 Layer visualization of convolutional layer of VGG19

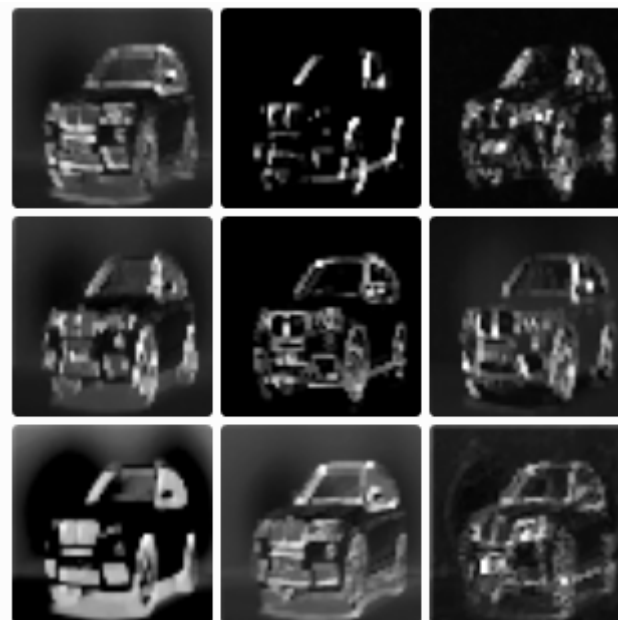


Figure 3.10 Layer visualization of maxpooling layer of VGG19

Work and Analysis

We have obtained better output result for VGG-19 model, the following results are given below:

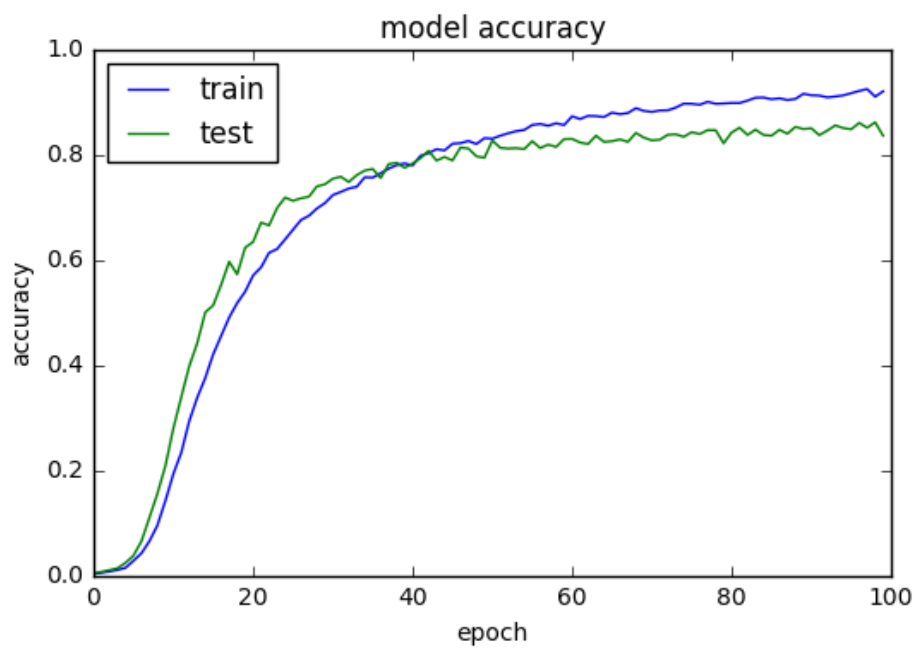


Figure 3.11 VGG-19 model accuracy graph

The training accuracy was 92.56% and the testing accuracy was 86.25%. Here we see an improvement in the testing accuracy of VGG-19 compared to that of VGG-16.

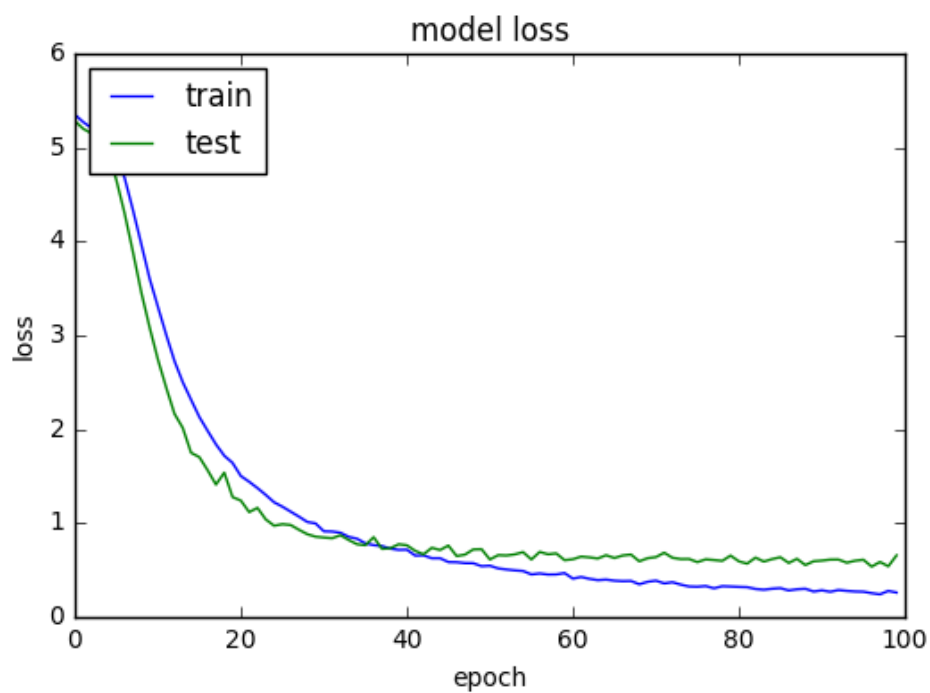


Figure 3.12 VGG-19 model loss function graph

Work and Analysis

The loss function of trained data was 24.28% and the tested data was 54.11%. We can see a decrease of loss function of the tested data compared to VGG-16.

Volvo 240 Sedan 1993	Volvo 240 Sedan 1993	Audi V8 Sedan 1994	Audi 100 Wagon 1994	Audi 100 Sedan 1994	Volkswagen Golf Hatchback 1991
					
Testing car 1	1st Pre: 99.99%	2nd Pre: 00 %	3rd Pre: 00%	4th Pre: 00%	5th Pre: 00%
Daewoo Nubira Wagon 2002	Daewoo Nubira Wagon 2002	Plymouth Neon Coupe 1999	Volkswagen Golf Hatchback 1991	Acura Integra Type R 2001	FIAT 500 Convertible 2012
					
Testing car 2	1st Pre: 97.45 %	2nd Pre:2.54 %	3rd Pre:0.01 %	4th Pre: 00%	5th Pre: 00 %
Buick Enclave SUV 2012	Buick Enclave SUV 2012	Hyundai Veracruz SUV 2012	Buick Verano Sedan 2012	Chevrolet Traverse SUV 2012	GMC Acadia SUV 2012
					
Testing car 3	1st Pre: 100 %	2nd Pre:00%	3rd Pre: 00%	4th Pre: 00%	5th Pre: 00%
Aston Martin V8 Vantage Coupe 2012	Aston Martin V8 Vantage Coupe 2012	Aston Martin V8 Vantage Convertible 2012	Aston Martin Virage Coupe 2012	BMW M3 Coupe 2012	Aston Martin Virage Convertible 2012
					
Testing car 4	1st Pre: 99.97 %	2nd Pre:0.02 %	3rd Pre: 00%	4th Pre: 00 %	5th Pre: 00 %
BMW 1 Series Convertible 2012	BMW 1 Series Convertible 2012	BMW 1 Series Coupe 2012	BMW Z4 Convertible 2012	BMW 3 Series Wagon 2012	Mercedes-Benz 300-Class Convertible 1993
					
Testing car 5	1st Pre: 99.97%	2nd Pre: 0.03%	3rd Pre: 00%	4th Pre: 00%	5th Pre: 00 %

Figure 3.13 Testing output of VGG-19 with accuracy of different models of car

The diagram shows the predicted output and their respective percentages. Here almost all the predictions made were highly successful.



Testing Car	Best Accuracy Car	Accuracy%	Prediction
 Audi TT Hatchback 2011	 Audi TT Hatchback 2011	99.99%	Yes
 BMW X3 SUV 2012	 BMW X3 SUV 2012	97.64%	YES
 Chevrolet Cobalt SS 2010	 Chevrolet Cobalt SS 2010	100%	YES
 Chevrolet Malibu Hybrid Sedan 2010	 Chevrolet Malibu Hybrid Sedan 2010	100%	YES
 Dodge Ram Pickup 3500 Crew Cab 2010	 Ford Ranger SuperCab 2011	23.41%	No

Table 3.2 Prediction after testing phase of VGG-19

We have computed down to 5 image/ car model to show the prediction accuracy and the diagram above shows that 4 out of 5 different car model gave out positive output result. This shows how sound this model and its computation i.e. how better it is training and testing set turned out to be.

3.4.3. Using ConvNet model Inception V3

In (2.4.3) the inception process has been briefly discussed. To reach our objective we had to make certain changes to the basic version of this Inception model. The input image size here was 299 x 299, whereas in other models we used 224 x 224. In this model while conducting the testing phase, if after five iteration no improvement was made the learning rate was decreased by a factor of 0.5 (initial learning rate was 0.001). Also, when the testing accuracy increases we save the improved weights, if no increase is made then there will be no improvement.

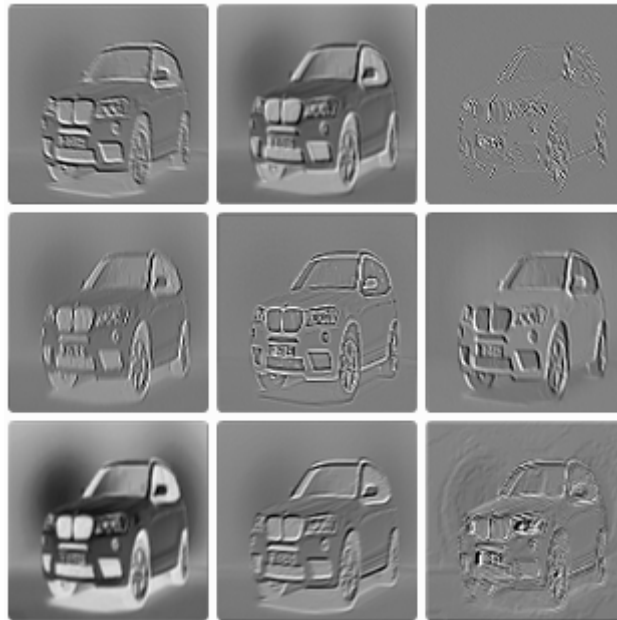


Figure 3.14 Layer visualization of convolutional layer of Inception V3



Figure 3.15 Layer visualization of maxpooling layer of Inception V3

Work and Analysis

Below are the given graphs and output results:

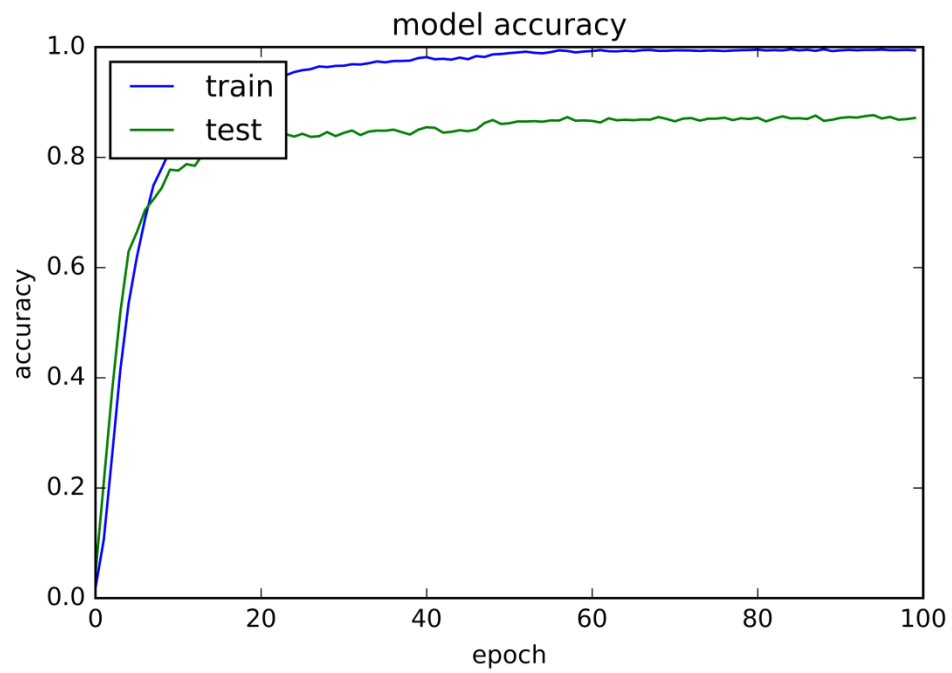


Figure 3.16 Inception V3 model accuracy graph

The training accuracy of Inception V3 was 99.61% and the testing accuracy was 87.65%, which is much better than the VGG-16 and VGG-19.

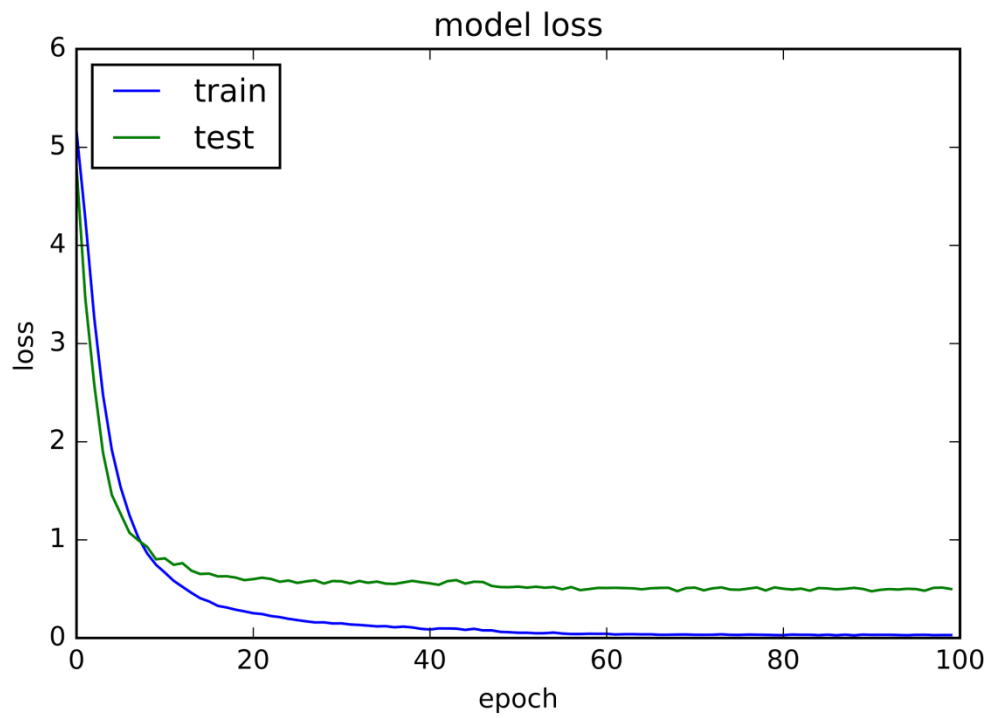


Figure 3.17 Inception V3 model loss function graph

The training loss function of Inception V3 was 2.71% and the testing loss function 50.15%, which is comparatively much lower than VGG-16 and VGG-19.

Work and Analysis

BMW M3 Coupe 2012	BMW M3 Coupe 2012	BMW M5 Sedan 2010	BMW M6 Convertible 2010	BMW 3 series Wagon 2012	BMW ActiveHybrid 5 Sedan 2012
					
Testing car 1	1st Pre: 100 %	2nd Pre: 00 %	3rd Pre: 00%	4th Pre: 00%	5th Pre: 00%
Bentley Continental Supersports Conv. Convertible 2012	Bentley Continental Supersports Conv. Convertible 2012	Bentley Continental GT Coupe 2012	Bentley Continental Flying Spur Sedan 2007	BMW 1 Series Convertible 2012	MINI Cooper Roadster Convertible 2012
					
Testing car 2	1st Pre: 100 %	2nd Pre:00 %	3rd Pre:00 %	4th Pre: 00 %	5th Pre: 00 %
Dodge Magnum Wagon 2008	Dodge Magnum Wagon 2008	Dodge Magnum Wagon 2008	Audi TT RS Coupe 2012	Chevrolet Cobalt SS 2010	Ford Expedition EL SUV 2009
					
Testing car 3	1st Pre: 99.99 %	2nd Pre:0.01%	3rd Pre: 00%	4th Pre: 00%	5th Pre: 00%
Buick Enclave SUV 2012	Buick Enclave SUV 2012	Hyundai Veracruz SUV 2012	Buick Verano Sedan 2012	Chevrolet Traverse SUV 2012	GMC Acadia SUV 2012
					
Testing car 4	1st Pre: 100 %	2nd Pre:00%	3rd Pre: 00%	4th Pre: 00%	5th Pre: 00%
Mercedes-Benz E-Class Sedan 2012	Mercedes-Benz E-Class Sedan 2012	Mercedes-Benz C-Class Sedan 2012	BMW 3 Series Wagon 2012	Chrysler Sebring Convertible 2010	Ford Expedition EL SUV 2009
					
Testing car 5	1st Pre: 100 %	2nd Pre: 00%	3rd Pre: 00%	4th Pre:00 %	5th Pre: 00%

Figure 3.18 Testing output of Inception V3 with accuracy of different models of car

The diagram shows the predicted output and their respective percentages. Here all the predictions made were highly successful.

Testing Car	Best Accuracy Car	Accuracy%	Prediction
		100%	YES
Audi TT Hatchback 2011	Audi TT Hatchback 2011		
		100%	YES
BMW X3 SUV 2012	BMW X3 SUV 2012		
		89.62%	YES
Chevrolet Cobalt SS 2010	Chevrolet Cobalt SS 2010		
		100%	YES
Chevrolet Malibu Hybrid Sedan 2010	Chevrolet Malibu Hybrid Sedan 2010		
		76.36%	YES
Dodge Ram Pickup 3500 Crew Cab 2010	Dodge Ram Pickup 3500 Crew Cab 2010		

Table 3.3 Prediction after testing phase of Inception V3

Inception gave the best loss function i.e. the lowest loss function prediction. Here, we see that 5 out of 5 predictions made are correct, which shows how Inception is better than both VGG-16 and VGG-19

3.4.4. Using ConvNet model Residual Network

The Residual Network [13] is another ConvNet model which uses the approach of passing an input to one layer and then the output produced by that same layer is passed with the input that was given to that layer to the next phase of layer. This procedure continues in all the neural layers of the residual network. As a result, a well-trained model with low loss function can be achieved.

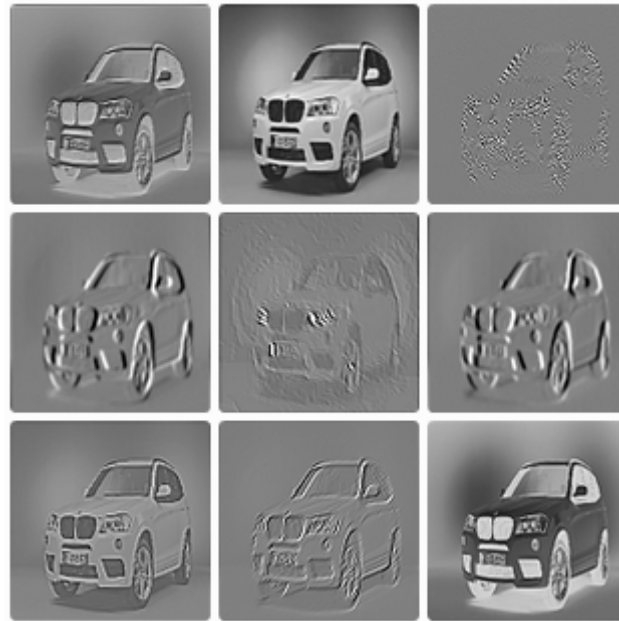


Figure 3.19 Layer visualization of convolutional layer of ResNet50



Figure 3.20 Layer visualization of maxpooling layer of ResNet50

Work and Analysis

We have compared the result obtained for this model with our previous model and the output we received using this ConvNet model is given below:

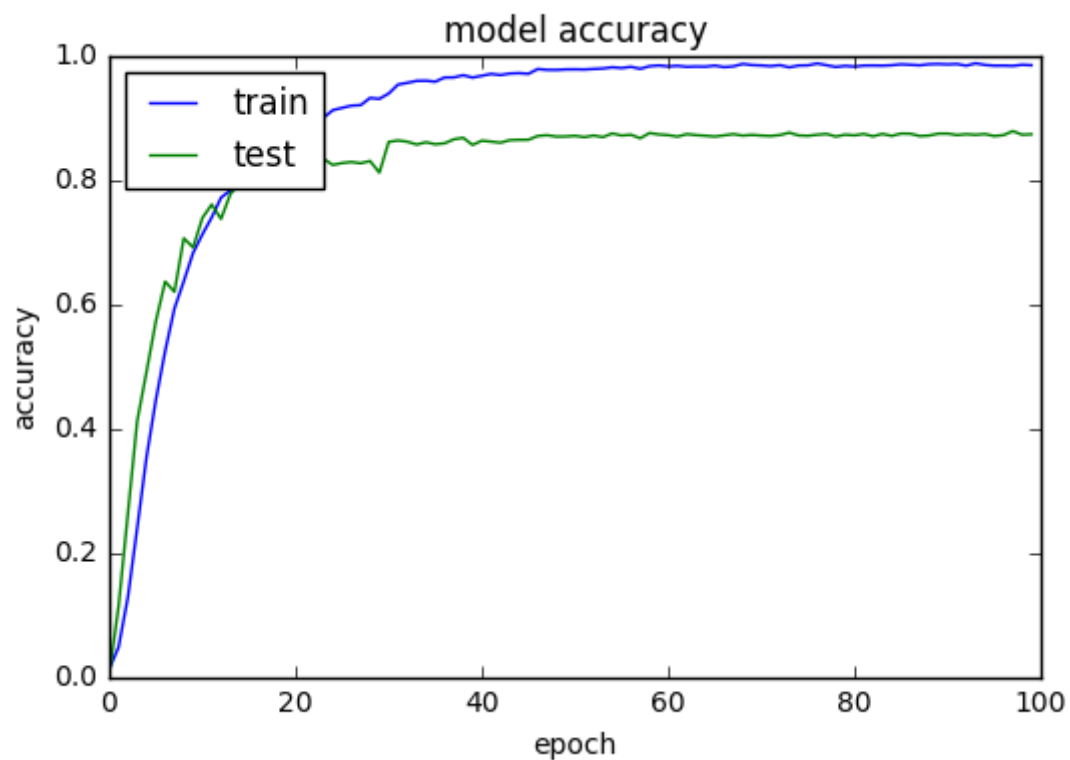


Figure 3.21 ResNet50 model accuracy graph

The testing accuracy of ResNet50 model was 87.95% and the training accuracy was 98.86%, which is the best among all the three models above (VGG-16, VGG-19 and Inception V3).

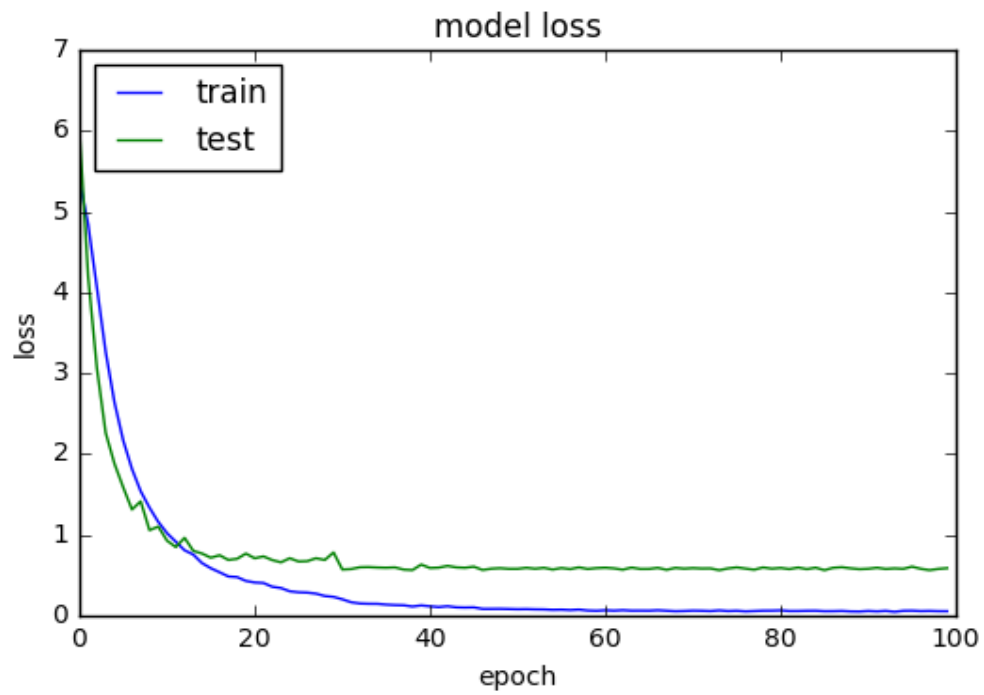


Figure 3.22 ResNet50 model loss function graph

The testing loss function was 56.45% and the training loss function was 4.88%, which is lower than VGG-16 and VGG-19 but compared to Inception V3 it is higher.































Ferrari FF Coupe 2012	Ferrari FF Coupe 2012	Audi V8 Sedan 1994	Audi 100 Wagon 1994	Audi 100 Sedan 1994	Volkswagen Golf Hatchback 1991
					
Testing car 1	1st Pre: 100%	2nd Pre: 00 %	3rd Pre: 00%	4th Pre: 00%	5th Pre: 00%
Chevrolet Silverado 1500 Regular Cab 2012	Chevrolet Silverado 1500 Regular Cab 2012	Chevrolet Silverado 2500 HD Regular Cab 2012	Chevrolet Silverado 1500 Hybrid Crew Cab 2012	Chevrolet Silverado 1500 Extended Cab 2012	Jeep Compass SUV 2012
					
Testing car 2	1st Pre: 100%	2nd Pre:00%	3rd Pre:00%	4th Pre: 00%	5th Pre: 00 %
Audi S4 Sedan 2007	Audi S4 Sedan 2007	Audi RS 4 Convertible 2008	Audi A5 Coupe 2012	Audi S6 Sedan 2011	Mercedes-Benz E-Class Sedan 2012
					
Testing car 3	1st Pre: 99.99%	2nd Pre:0.01%	3rd Pre: 00%	4th Pre: 00%	5th Pre: 00%
Acura Integra Type R 2001	Aston Martin V8 Vantage Coupe 2012	Plymouth Neon Coupe 1999	Acura Integra Type R 2001	Hyundai Elantra Sedan 2007	BMW 3 Series Sedan 2012
					
Testing car 4	1st Pre: 61.11%	2nd Pre:27.35 %	3rd Pre: 6.35%	4th Pre: 00 %	5th Pre: 1.10 %
Aston Martin V8 Vantage Convertible 2012	Aston Martin V8 Vantage Convertible 2012	BMW 6 Series Convertible 2007	Aston Martin Virage Convertible 2012	BMW 3 Series Wagon 2012	Mercedes-Benz 300-Class Convertible 1993
					
Testing car 5	1st Pre: 72.94%	2nd Pre:9.79%	3rd Pre: 6.22%	4th Pre: 4.58%	5th Pre: 3.54%

Figure 3.23 Testing output of ResNet50 with accuracy of different models of car

The five prediction given above shows the accuracy of the predictions and their correctness. Depending on the processing of the image, comparisons are made with the input and the trained model and an estimated output percentage is given out. Here we see from 1st car model to the last, only the 4th car model (i.e. Acura Integra Type R 2001) gave out a high percentage of 61.11% of, which is inaccurate. Below is an accuracy diagram given where it shows 4 out of 5 predictions are correct, here in this diagram we can see that same thing. All the predictions made are not correct, this is not due to improper image but due to less training of the respective car model.

Testing Car	Best Accuracy Car	Accuracy%	Prediction
 Audi TT Hatchback 2011	 Audi TT Hatchback 2011	89.62%	YES
 BMW X3 SUV 2012	 BMW X3 SUV 2012	100%	YES
 Chevrolet Cobalt SS 2010	 Chevrolet Cobalt SS 2010	100%	YES
 Chevrolet Malibu Hybrid Sedan 2010	 Chevrolet Malibu Hybrid Sedan 2010	100%	YES
 Dodge Ram Pickup 3500 Crew Cab 2010	 Dodge Ram Pickup 3500 Crew Cab 2010	100%	YES

Table 3.4 Prediction after testing phase of ResNet50

After running ResNet50 and receiving a successful accuracy percentage rate we computed down to 5 car model images to give a brief reference of the output: 5 out of 5 prediction gave positive result for Residual network model.

3.4.5. Final prediction

The final predicted output of the models after convergence are given below in a table:

Model Name	Train Accuracy	Test Accuracy	Train Loss	Test Loss
VGG-16	86.94%	83.05%	42.85%	61.14%
VGG-19	92.56%	86.25%	24.28%	54.11%
Inception V3	99.61%	87.65%	2.71%	50.15%
ResNet50	98.86%	87.95%	4.88%	56.45%

Table 3.5 Table of predicted output

The table here shows that ResNet50 gave the best test accuracy which is our final prediction. As discussed above in our paper our goal was to prove that ResNet50 is the most accomplished model among the other proposed models, which is clearly achieved as shown in this table.

The Graphical figures of the Train accuracy, Train loss, Test accuracy, Test loss and a final bar graph of predictions are given below:

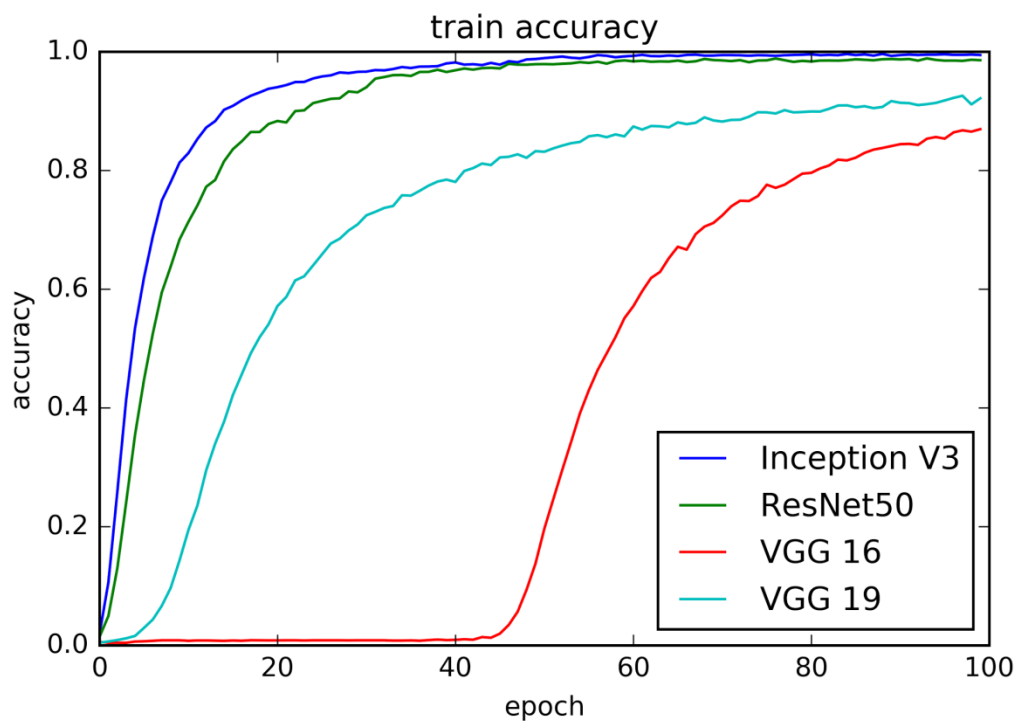


Figure 3.24 Train accuracy of the four Models

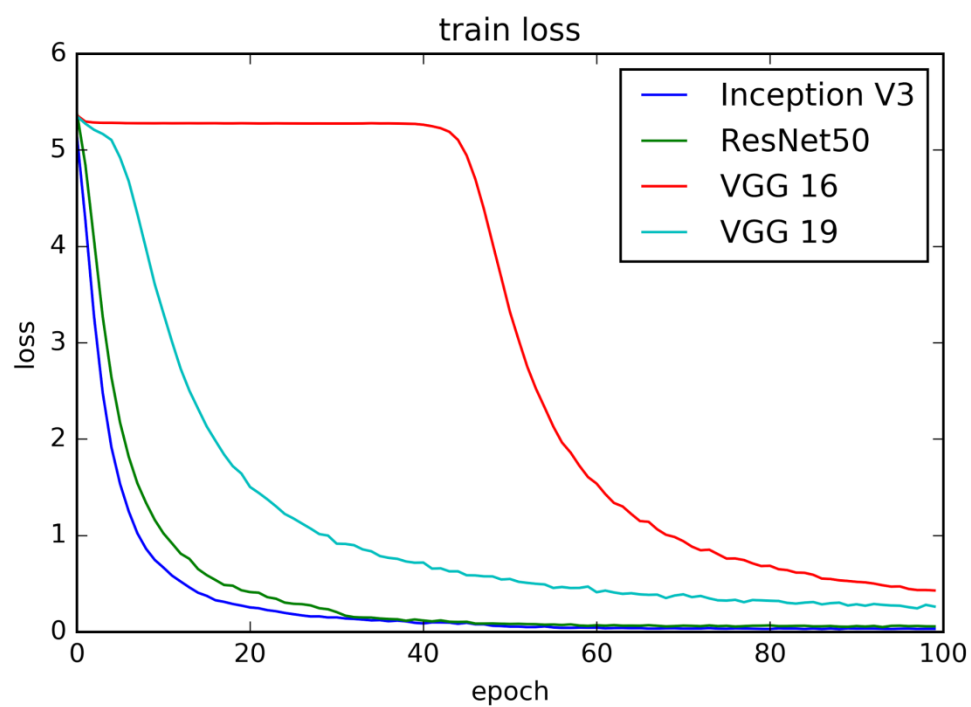


Figure 3.25 Train loss of the four models

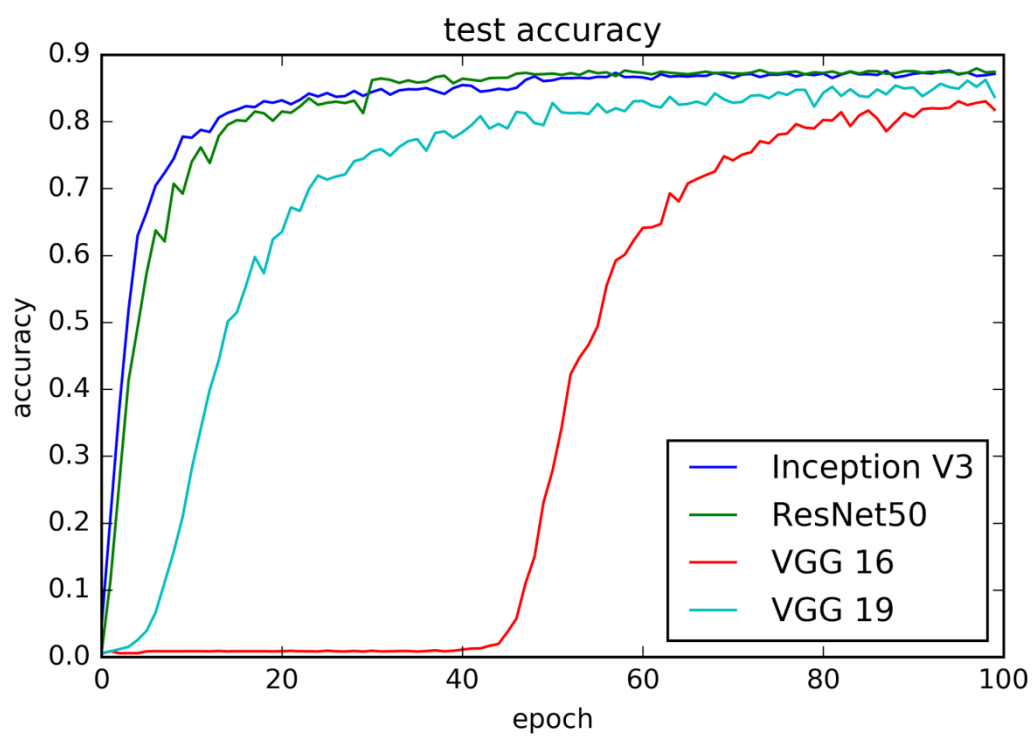


Figure 3.26 Test accuracy of the four models

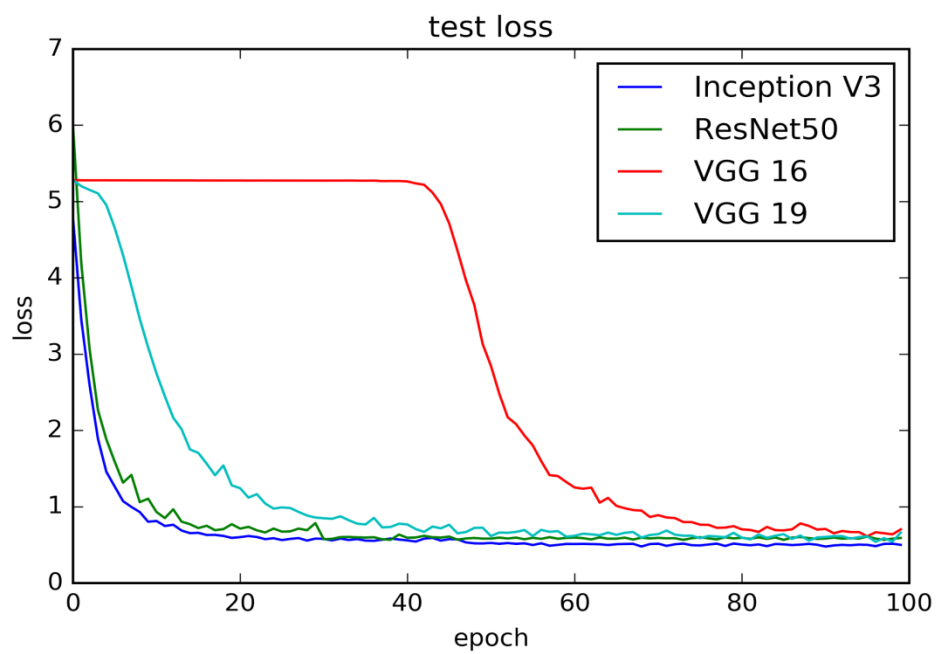


Figure 3.27 Test loss of the four models

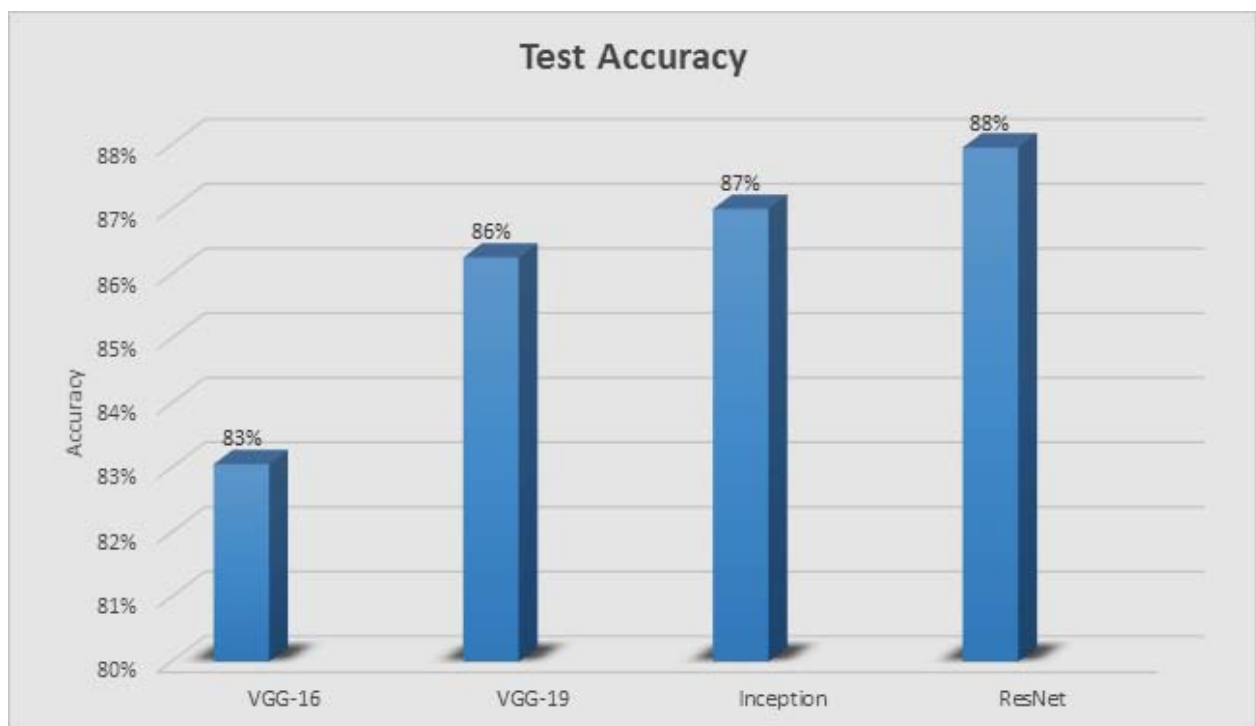


Table 3.6 Bar chart of final prediction

3.4.6. Hardware Implementation and Toolkits

The models we have used for our project are all winners of ImageNet challenge. They are all complex deep convolutional networks because of which we needed high configuration GPUs. We had to buy a Zotac GTX 1060 Graphics card and another ASUS Turbo GTX 1060 Graphics card, each having 6GB onboard memory and 16GB RAM used in two separate PC's. We have used:

- Windows 10 operating system
- Visual Studio 2015 Community Edition Update 3 (Used for its C/C++ compiler (not it's IDE) and SDK)
- Anaconda (64-bit) w. Python 2.7 (A Python distro that gives us NumPy, and other scientific libraries)
- CUDA 8.0.44 (64-bit) (Used for its GPU math libraries, card driver, and CUDA compiler)
- MinGW-w64 (5.4.0) (Used for its Unix-like compiler and build tools (g++/gcc, make...) for Windows)
- Theano 0.8.2 (Used to evaluate mathematical expressions on multi-dimensional arrays)
- Keras 1.1.0 (Used for deep learning on top of Theano)
- OpenBLAS 0.2.14 (Used for its CPU-optimized implementation of many linear algebra operations)
- cuDNN v5.1 (August 10, 2016) for CUDA 8.0 (Used to run vastly faster convolution neural networks)



Figure 3.28NumPy

Numpy was used to load the dataset as numpy array in the memory.



Figure 3.29 Matplotlib

To plot the graph of testing and training accuracy and loss of the models, matplotlib was used.



Figure 3.30 Theano and Keras

Theano was used at the backend as the machine learning library with Keras as the wrapper library.



Figure 3.31 Python

The whole algorithm and our system was implemented using Python 2.7

3.4.7. Limitations

In conducting our project, we had to face many difficulties, some of which we were able to overcome or find an alternative to reach completion of our work. One of the major limitations of our project was obtaining proper resources I.e. research materials relating to our thesis. Since it is a new concept so it was quite challenging in finding the proper resources. Subsequently we needed a proper dataset for our convolutional neural network which was difficult to obtain. Training and testing of a neural network requires a supercomputer or a high

Work and Analysis

configuration computer which was not available to us. So we had to buy two Graphics card for running our algorithms of each of the CNN models. CNN is a large network which perform even better when it is being trained with even greater dataset and more time to compute, this is something that we could not do due to lack to time and materials. We were also unable to run Inception-V1(GoogleNet) [14] and AlexNet,as it does not have pre-trained weights of ImageNet which was trained using Theano.

Chapter 4

Conclusion

4.1. Conclusion

While conducting our whole research we came to understand how each model performs and are different from one another. We started with VGG-16 which gave a low output percentage compared to its other version i.e. VGG-19 which was better. Then we tested our Inception V3 model which gave both better testing prediction and lower loss function in comparisons to the VGG-16 and VGG-19. However, the best model as we had assumed was Residual Network (ResNet50) which gave the best prediction for testing data, but had a higher loss function than that of Inception only. Although, ResNet50 leads to faster convergence and better predicted result compared to the other models. This is not the final deduction as there are many different changes that could be made to these model and there are other models which could be implemented as well. Overcoming the limitations and with proper resources more sound output result can be achieved. Nonetheless, the results obtained were very much satisfying and a proper deduction can be made as to which model is better, in our case it was Residual Neural Network. Convolutional Neural Network is a vast field where a lot of advancement can be made with further research and analysis.

4.2. Future Work

Our future goal is to contribute to our ever developing society by providing an advanced model of car recognition classification system. For that we would need train our system with more data and over a good period of time. We want our project to be used by the government (i.e. using our system in all forms of video cameras), so that we can have a protected and secured society. In, future our aim is to work with other object recognitions (such as Animals, Humans, aircrafts, ships, etc.). From our model comparisons, we can choose a clear model for a proper classification system. Furthermore, we plan to make a search image search engine where we will use image as inputs (the prototype of this reversed search engine has been already implemented by us). We plan and expect to make further contributions to our research in near future.

Conclusion

BIBLIOGRAPHY

- [1] Y. Huang, R. Wu, Y. Sun, W. Wang, and X. Ding, "Vehicle Logo Recognition System Based on Convolutional Neural Networks With a Pretraining Strategy," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 1951–1960, 2015.
- [2] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3D Object Representations for Fine-Grained Categorization," *2013 IEEE International Conference on Computer Vision Workshops*, 2013.
- [3] J. Krause, H. Jin, J. Yang, and L. Fei-Fei, "Fine-grained recognition without part annotations," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [4] S. Xie, T. Yang, X. Wang, and Y. Lin, "Hyper-class augmented and regularized deep learning for fine-grained image classification," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [5] F. Chollet, "keras," Github, [Online]. Available: <https://github.com/fchollet/keras>.
- [6] J. Bian, "Quiver," [Online]. Available: <https://github.com/jakebian/quiver>
- [7] S. Agarwal, A. Awan, and D. Roth, "Learning to detect objects in images via a sparse, part-based representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 11, pp. 1475–1490, 2004.
- [8] T.-Y. Lin, A. Roychowdhury, and S. Maji, "Bilinear CNN Models for Fine-Grained Visual Recognition," *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015.
- [9] P. Sermanet and Y. Lecun, "Traffic sign recognition with multi-scale Convolutional Networks," *The 2011 International Joint Conference on Neural Networks*, 2011.
- [10] J. Krause, T. Gebru, J. Deng, L.-J. Li, and L. Fei-Fei, "Learning Features and Parts

BIBLIOGRAPHY

- for Fine-Grained Recognition," 2014 22nd International Conference on Pattern Recognition, 2014.
- [11] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv, vol. abs/1409.1556, p. preprint arXiv:1409.1556, 2014.
- [12] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," arXiv, p. preprint arXiv:1409.1556, 2014.
- [13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
- [14] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.
- [15] L. Bottou, "Large-Scale Machine Learning with Stochastic Gradient Descent," Proceedings of COMPSTAT'2010, pp. 177–186, 2010. L. Bottou, "Large-Scale Machine Learning with Stochastic Gradient Descent," Proceedings of COMPSTAT'2010, pp. 177–186, 2010.
- [16] F. Zhang, X. Xu, and Y. Qiao, "Deep classification of vehicle makers and models: The effectiveness of pre-training and data enhancement," 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO), 2015.
- [17] T., "Theano/Theano," GitHub, 2016. [Online]. Available: <https://github.com/Theano/Theano>. [Accessed: 13-Dec-2016].