"Dog Breed Classifier Using CNN and Image Processing"

Project report submitted in fulfilment of the requirement for the degree of Bachelor of Technology
In

Computer Science and Engineering and Information Technology

By

Aditya Thakur (161274)

Under the supervision of
(Dr. Suman Saha)

To

Department of Computer Science & Engineering and Information TechnologyJaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report entitled **"Dog Breed Classifier UsingCNN and Image Processing"** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from July 2019 to May 2020 under the supervision of **Dr. Suman Saha** (Assistant Professor, Senior Grade, Computer Science &Engineering Department).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

[illegible]

Aditya Thakur (161274)

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

[illegible]

Dr. Suman Saha
Assistant Professor, Senior Grade
Computer Science and Engineering and Information Technology

Dated: 28-05-2020

ACKNOWLEDGEMENT

I owe my profound gratitude to my project supervisor **Dr. Suman Saha**, who took keen interest and guided me all along in my project work titled - **“Dog Breed ClassifierUsing CNN and Image Processing** till the completion of my project by providing all the necessaryinformation for developing the project. The project development helped me in research and I got to know a lot of new things in my domain. I am really thankful to him.

TABLE OF CONTENTS

ABSTRACT

Pattern recognition (PR) is discovered out as a human recognition approach which can be completed through computer technologies. We need to first input useful records of figuring out the object into the machine. For this cause, we should summary the popularity item and set up its mathematical version to give a reason behind it and replace the recognition item for what the machine can way. The description of this item is the sample. Simply talking, the pattern recognition is to discover the category to which the object belongs, eg. the face in face recognition. Our assignment is based totally on PR it simply is to turn out to be privy to the dog's breed. In our assignment, based totally mostly on 20,000+ snap shots of 100 and twenty breeds of dogs. In our project, based on 20,000+ images of 120 breeds of dogs. We also make some improvements on the optimization methods to increase our identification accuracy. At the end we will see how much accuracy we can get.Our awesome accuracy can be as plenty as 85%.

This project makes use of computer vision and machine learning strategies to identify dog breeds from pictures. First, we perceive dog facial key elements for each image by the usage of a convolutional neural network. These key elements are then used to extract abilities via descriptors and color histograms. We then compare a selection of type algorithms, which use the ones competencies to assume the breed of the dog tested within the picture.

CHAPTER -01

DOG-BREED-CLASSIFIER BY CNN & IMAGE-PROCESSING

INTRODUCTION

IMAGE-PROCESSING

Ithas beenconsideredasanadvancedtechniquethatisrequiredfortheconversionofanimage into its digital form or either in its analog form. It is also used for the purpose where we require only few highlighted areas in an image and thus only the important areas should beextracted.Firstly,theimageisbasicallyprocessedinitsrawformthentheresultantor themodifiedimageisgivenasanoutputtothatrawimage.Inmachinelearningalgorithms we use this image processing as part of a project which can be further used for works such as model trainingetc.

Various steps in image processing are as:

1. Image collection by some appropriatetool.
2. Second step is to perform some operations in order to find some crucial patterns among theimage.
3. Last stage is where output is an image resultant of alloperations.

IMAGE

Digital form of a picture is known as image. Image in raster form is known as bitmap. Examples of image format jpeg, png etc.

IMAGE(types)

1. Binary-Image: Every pixel in the image is eligible for binary classification of colors i.e black(1)as well as white(0) and these colors are denoted by 1 and 0respectively.

2. Gray Scale Images: Here each pixel rather than containing color information these contain all the information as intensity oflight.

3. ColoredImages:Coloredinformationisstoredasintensityoflight.Basically,itcontains three color red blue andgreen.

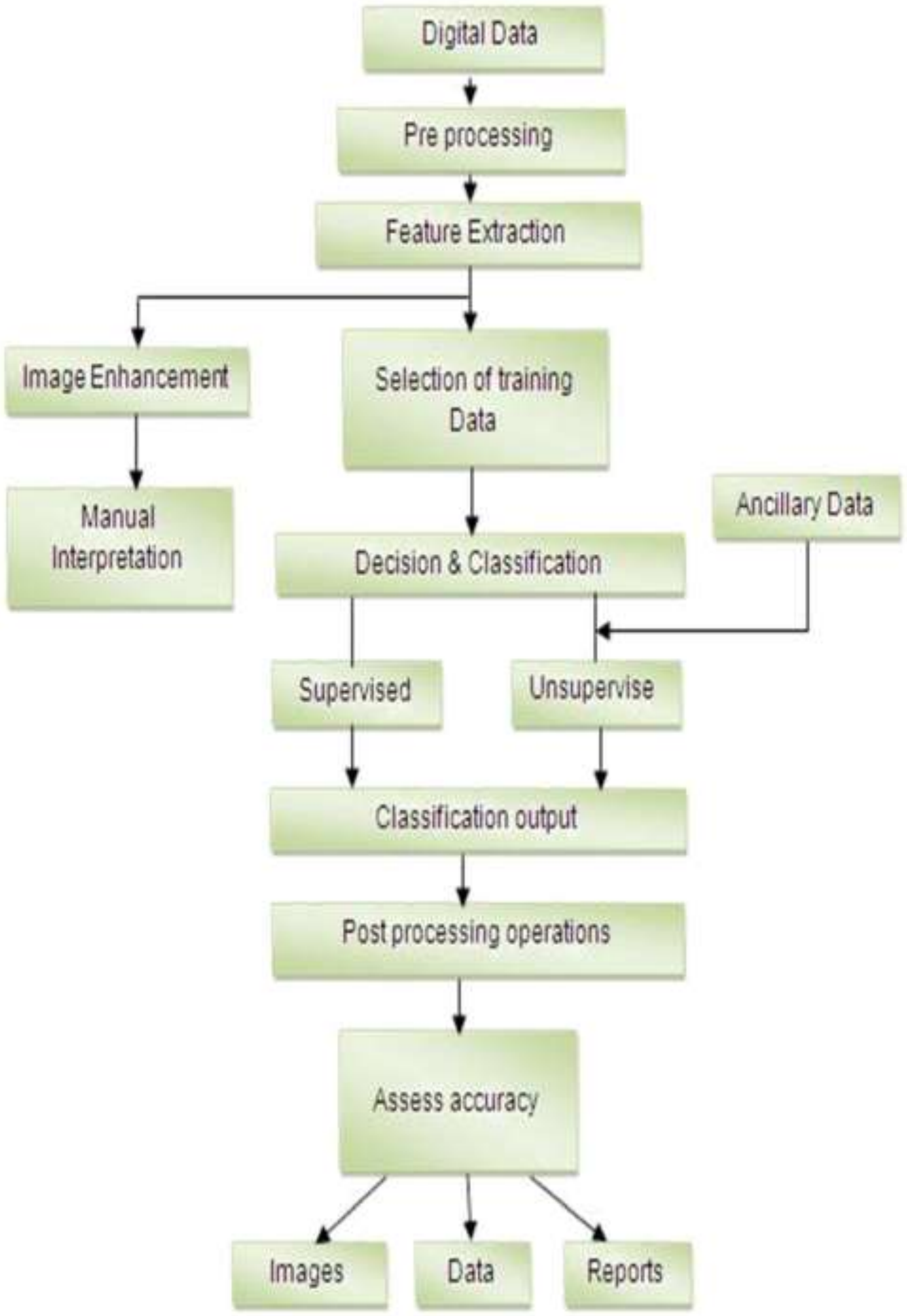


Fig. 1: Flow Chart Showing Diffrent Phases in Digital Image Processing

Convolution Neural Network:

It is derived from the neural structure that a human possesses with similar procedure of functioning.Thebasicfunctioningandtheanalogyofbotharesameaswearetalkingon the basis of some specifiedconstraints.

A signal known as output signal is basically generated by neurons which are collectively known as human neural network and that signal is transferred to the various parts of the body. The brain basically consists of neural network and is responsible for various computations. At the end of nerve cell branch like structure are basically responsible for all the required computations. These branch-like structures have sensors which detect the signals which are converted into electrical signals for the transmission purposes and then this signal traverse through axon to the nucleus and message to be returned isalso received through viceversa.

CNN has just similar neurons but there are weights and biases too. An input is given to these weights and biases in order to calculate the weighted sum which is eventually passed through the activation function in order to receive the final judgement.

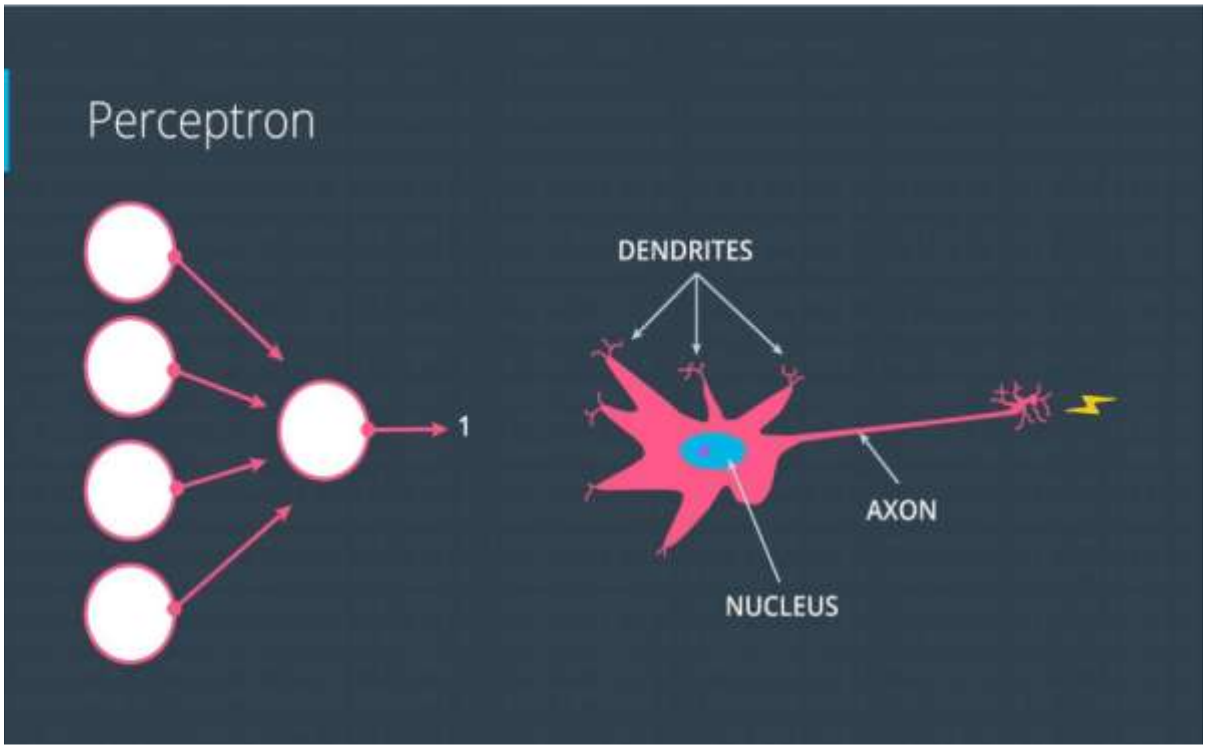


Figure 1.1 Diagram showing neuron and perceptron

Activation Function in CNN:

As per our knowledge and some research the network of neurons comprises of fairly vastcount and which neuron needs to be activated at any instant is basically decidedbytheactivationfunctionwith thecalculationofweightsandbias using addition method.Bringing nonlinearity to our model is basic task of the activation function because as if our model was linear then we would have faced so many difficulties in the process of training the data to our model in the case of dataset consisting of complexboundaries.

Types of activation functions are listed as below:

1. Sigmoid-Function: based upon the function 1/1+f(a) where f(a)=exp(-a) where the total value is b/w zero&one.
2. Hyperbolic-Function(tanh): based upon methodi.e 1-f(a)/1+f(a) where f(a)=exp(- 2a) and the total value is b/w -one andone.
3. Rectified-Linear-Units: here basic methodology is maximum(0,a) which means the total value is 0 for value <0 and a for value>=0

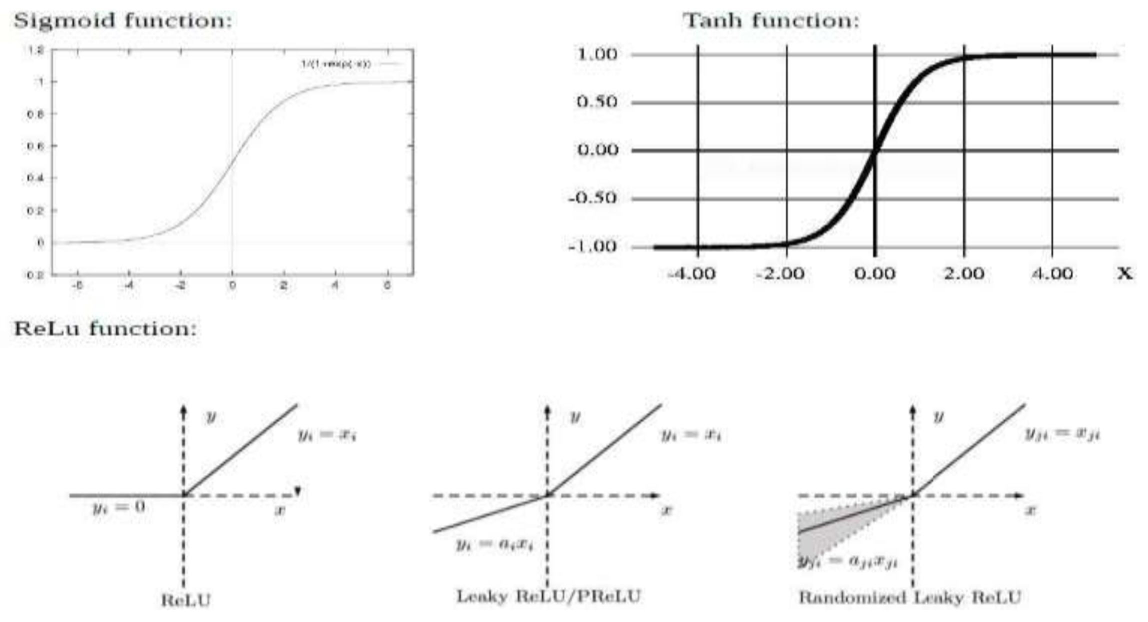


Figure 1.2 showing different activationfunction

1.1 ProblemStatement:

With the help of Image-Processing&Deep-Learning we need creation of a model which can take picturesi.e 120(approx)dog's breed as input and tells us about the breed of that dog. Use of transfer learning to recognize those breeds.

1.2 Objective:

- To predict the breed of dogs using deeplearning.
- Aiming for the high testing data accuracy.
- Using over 1 lakh of images of dogs to train our machine in order to make it capable enough to recognize 120 breeds ofdogs.
- For the purpose of pattern matching and making changes CNN isused.
- To predict the breed irrespective of its age whether it's a child or an adultdog.
- Forfilteringpurposesusingofpython,kerasandsomeoftheimageprocessingtools.

1.3 Methodology:

First of all, the images we have are converted in the form of black and white also known asgrayscaleformat.Theseimagesarestoredintheformofdatasetwhichispartitioned into training data set, testing and validation data set. Main work of training data set is to makecapableenoughtofindthepatternsamongtheimagesandalsoitispassedthrough thedeepneuralnetworks.Whetheritshouldbesentforwardorbackwarditisdecidedby calculationinerrorandadjustmentsofthe weights.Inordertoknowthedog's type, we feedrequired picture. Breed revealing's role i.e mainly done by feedforward which leads to calculation of probabilities of breed of the dog and thusoutput.

CHAPTER -02

LITERATURE

REVIEW

MACHINE LEARNING:

It is considered as a branch inArtificial-Intelligence which employs raw information observation and variousMachine-Learning-Algorithmsforgeneration ofmodels,trainthemwiththehelpof training data and test them with help of testing data in learning of patterns among the images and prediction of the output with decent accuracy. The test scores in the testing process clearly determines the accuracy of themodel.

TYPES OF MACHINE LEARNING:

- **Supervised-machine-learning:**

  It is kind oflearningthereiscreationoffunction(fx)capable of exceptinginputsbasically &givesusoutput.Themachineistrainedwiththeinputdatawhoseoutputisknown and in between this process a function is created who is able to give us output when the input data isgiven.

  Almost 7 out of 10 machine learning algorithms are of this machine-learning-algorithms.

  Types of supervised-learning given as below:

  - Logistic-Regression
  - Linear-Regression
  - Naïve-Bays
  - Decision-Tree
  - K-Means
  - Random-Forest
  - KNN.

- **Unsupervised-machine-learning:**

Actually, it isa typethat's mainlyin usageifthere'snoclassificationamongtheinput data and no labelling as well. These algorithms clearly use the highlighted areas as for pattern matching in order to reveal the unidentified data used for the training purposes. These algorithms are not well known because of their less accuracy and that is the reason why all the ML algorithms, 7 out of 10 are supervised algorithms. But as per their input data conditions their efficiency is remarkable and the only disadvantage, they have is their complexstructure.

Types of unsupervised learning are as below:

1. Hierarchical-clustering
2. Kmean-clustering
3. Expectation-minimization

- **SEMISUPERVISED:**

Thistechniqueisbaseduponbothof thecombinedfeaturesinfirstand the previous one'sfeatures i.e this will contain both identified and unidentified input data for thetrainingpurposesandthereisalsocreationoffunctionwhereithastofindthepatterns between the images in case of unidentified data. For its appropriate working it prioritizes the identified input data but for the most of the work it uses unidentified input data. Classification, regression and recognition are considered as it's sub methods. Some of the basic techniques that employs these algorithmsgiven as:

- Facial recognizing
- Vocal recognizing

- **Reinforcement-learning-algorithm:**

  In this typethespecific system is exposed to its external environment in order to learn about various patterns and training purposes. Here nothing is told to the modelabout anyinputdataandmachinehastolearnonitsownaboutalltheconditionsithasbeen putin.Themethodithasthatonlythatmachinehastodosomanyhitandtrialsbased upon the previous learning it has gained from past experiences. This machine is motivatedonthebasisofrewardsit'sbeenoffereddueto itscorrectjudgementsithas been made and for every wrong judgement it's been given some negative points. Henceduetoagreaternumberoftriesandinordertogainmorerewardsitsaccuracy of better prediction increases significantly. This is process which is necessary for agentinordertogainthefeedbackforthebetterunderstandingofnewfeaturesofthe system.

Some of the unsupervised learning algorithms under this category are as listed below:

- SARSA(system action reward stateaction)
- DQN(deep qnetwork)
- DDPG(deep deterministic policy gradient) and Qlearning

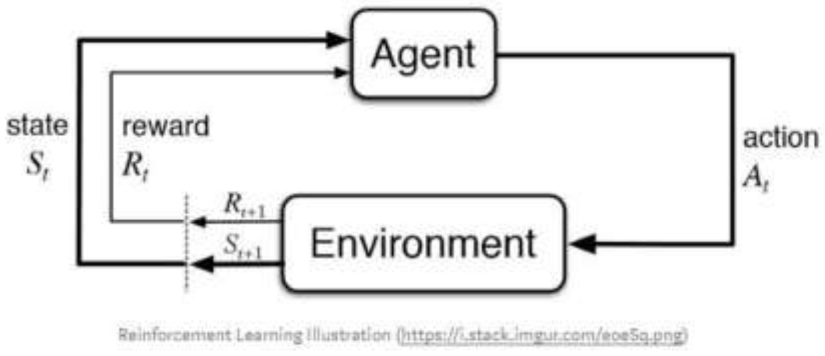

Agent
state
S_t
reward
R_t
action
A_t
R_{t+1}
S_{t+1}
Environment
Reinforcement Learning Illustration (https://i.stack.imgur.com/eoeSq.png)

Figure 2.1 showing interaction of agent with environment

How a machine perceives an image:

As it is very clear from the fact that machines and humans are very different in structure and creation, so is their perceiving ability i.e how to perceive an image in front of them. Well it is not a big deal to humans in everyday life because we see more than amachine and there is lot of image capturing by our brain and we don’t need to notice about the detailsofeveryimagebecauseourmindworkswayfasterandbeforeweevennoticeour mind has all the details about the image it sees. Same is not the case with machines as they don’t possess the human brains which can make their work easier so we need to train these machines with the images of the various objects we want them to recognize. Inordertomakethemachineslearnthenewdatasetofimagesalltheimagesneedtobe convertedinthedigitalformfirstandalso,theycanbeusedforfurtherimageprocessing.

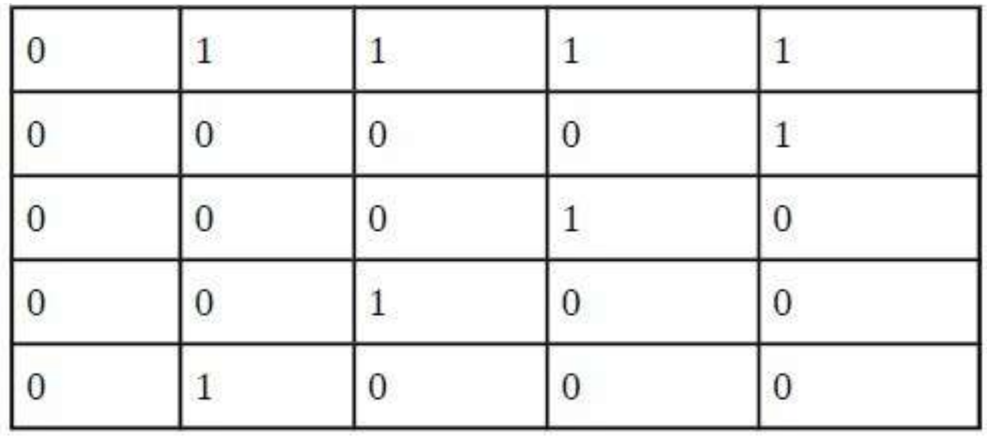
Animageismadeupofpixelswhicharebasicallycollectionofintensityandcolorvarying spatial patterns. For different images there are different order of arrangement for these pixels. Changing one or more-pixel arrangement can change the entire image forsure.

Let us take an example of an image with a number written on it. This image will contain thepixelswiththeintensityrangefrom1to256where1representthewhitestshadeand the other remaining shades will be dark when they are compared to the whitestshade.

After that it is the work of the neural network to identify patterns in the image which are not detectable by normal eye, so that it can recognize an image when it sees it again. These pixel values are generally considered as the features which are necessary for the machine learning algorithms for the final judgement of the output. The only thing we do here is that we provide different weights and bias so that we can alter our output.

To make simple to understand for us how a machine perceives an image we usually say that an image is made up of matrix that can be of any rows and columns and this matrix each one has values which will tell us about the intensity of any particular pixel in that image. These values are generally in the form of 0 and 1.

Example of an image representation:



0	1	1	1	1
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	1	0	0	0

Introduction to a Convolution Neural Network:

AswehavediscussedearliertheonlyworkofCNNispatternrecognitioninanimageand now,wearefocusingonitsstructure.Ithasthreelayersbasicallywhichareasfollowing:

- Convolutionlayer
- Poolinglayer
- Outputlayer

Convolution layer:

Inordertorecognizeanimagewhichismadeupofmatrixcontainingdifferentpixelvalues two-dsmall tablewhere everyvalue consistingof numericalvalue. In most of the cases this matrix is considered of 3*3 or 4*4. This weighted window is placed over the image matrix and multiplied with the values of that portion of the image matrix. These multiplied valuesare added in order to get the resultant value. After that the image is moved to the right or in downwards direction so that resultant value of the window can be calculated. This calculated value is used for the purpose of imagerecognition.

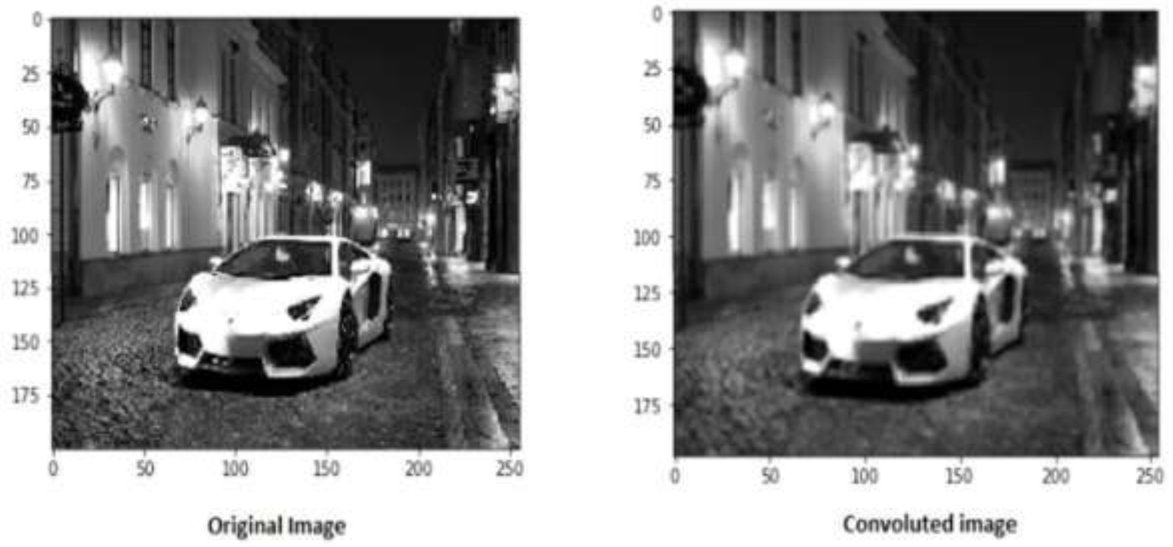
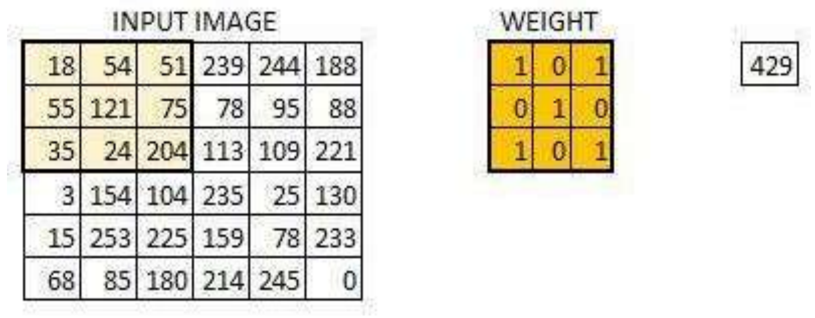


Figure 2.2 showing original and convoluted image



INPUT IMAGE

18	54	51	239	244	188
55	121	75	78	95	88
35	24	204	113	109	221
3	154	104	235	25	130
15	253	225	159	78	233
68	85	180	214	245	0

WEIGHT

1	0	1
0	1	0
1	0	1

429

Figure 2.3 showing input image and mask

To analyze depth effect in the input image we use multiple filters. Taking depth as a parameter singularfiltering surfacehas been proved quite insufficient for the desired result. 3D matrix is the resultant of manyone after other surfaces and it also take cares of depth. Final output is the resultant of sum of all the values of weighted matrix also known as convoluted image.

Pooling layer:

It is used when we want the machine to learn the larger inputs like images with greater size. If all the parameters are large which are required to learn by machine then these parametersmustbereducedforthepurposeofimagerecognitionandthatiswhypooling layers insertion is required between them. Doing so will also reduce our worry for the depth of the image as depth dimensions are also not altered when pooling is done. The mostly used pooling is known as max pooling in which the images appears as if no techniques were implied except for the dimensions which appears a little bitaltered.

There are three parameters which decide the size of output image:

- **Filters:**

  Width is the only factor which determines the number of the filters are required andthevolumeisalsodependentuponthewidth,sothevolumeisalsodecided by the number of filters. Width is the only factor which remains unaffected after pooling.

- **Stride’s role:**

  Thevalue which decides a filtering surface should be shifted left or downwards is decided particularly by the stride. Let us assume its value is one then the filter can be shifted in the listed direction by one unit which eventually leads to its larger size and that is why a larger stride is used to reduce the resultant image size.

- **Padding:**

  In this case if the value is zero then the size of the image doesn’t change otherwise the resultant image change.

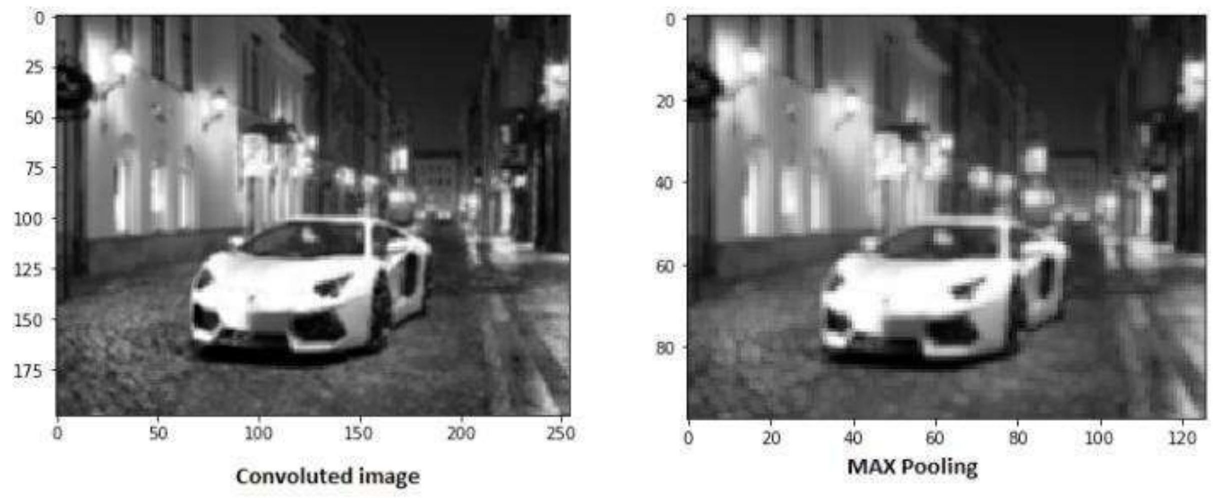


Figure 2.4 showing convoluted and Max Pooled image

Output layer:

Many features are already being extracted from the upper two layers and resultant image is little bit compressed. After that all the layers are applied to it together to get theresultantoutput.Apredictionmapin3Disrequiredtotellusthattowhichclassthat image belongsto.

Here faults can occur in detecting the image so that's why the inaccuracy in detecting the image is measured by the loss function whose formula is written below:

$$Size=((w-f+2p)/s) + 1$$

Where,

Size: size of the image s:

no. of strides used w:

volume of the input f:

size of the filter

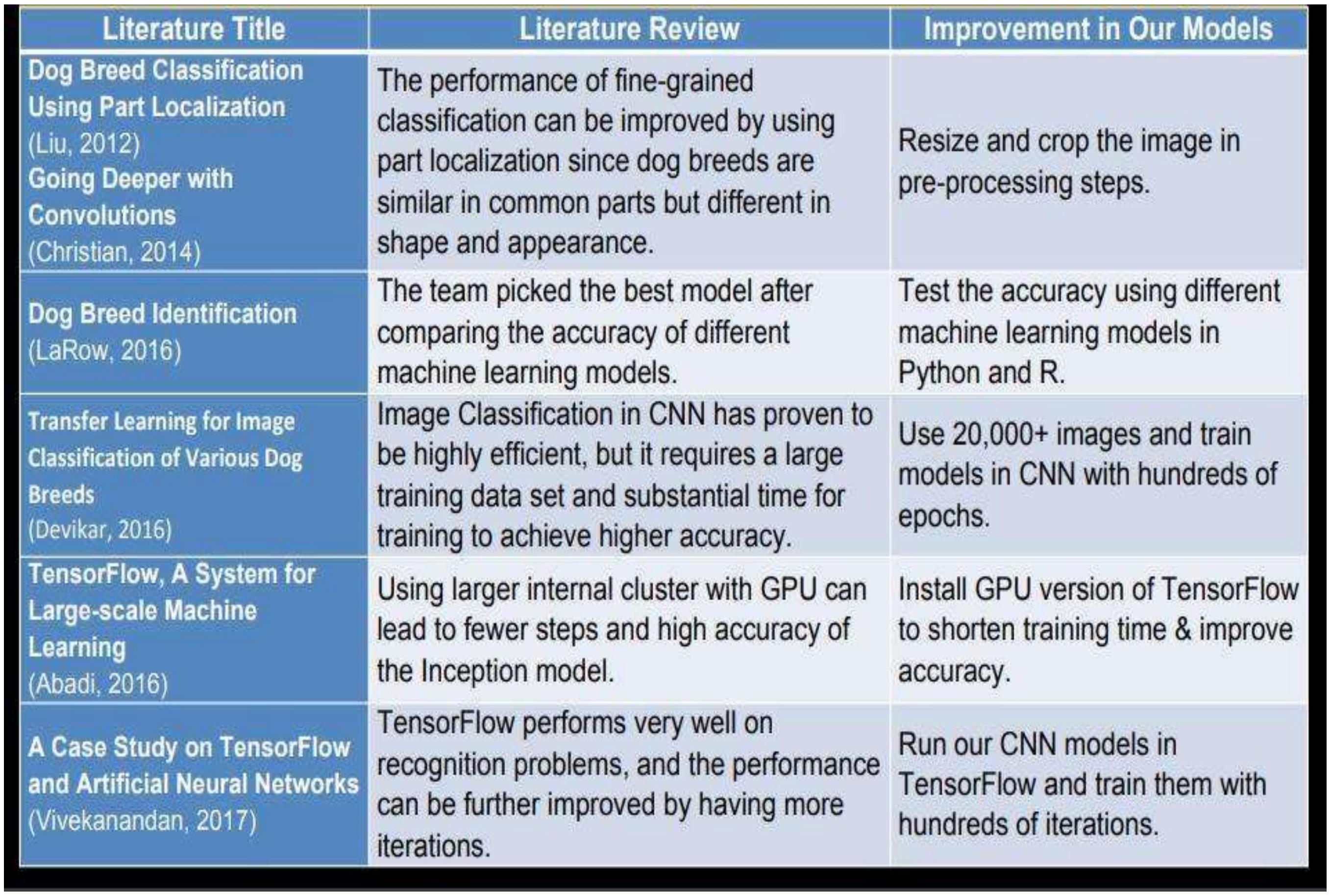
p: applied padding number

CHAPTER -03

System Development

This phase consists of following steps:

1. Designing a prototype of thesoftware.
2. Test phases required to improve thatsoftware.
3. Conversion of the prototype into realmodel.
4. Implementation of thealgorithms



Literature Title	Literature Review	Improvement in Our Models
Dog Breed Classification Using Part Localization (Liu, 2012) **Going Deeper with Convolutions** (Christian, 2014)	The performance of fine-grained classification can be improved by using part localization since dog breeds are similar in common parts but different in shape and appearance.	Resize and crop the image in pre-processing steps.
Dog Breed Identification (LaRow, 2016)	The team picked the best model after comparing the accuracy of different machine learning models.	Test the accuracy using different machine learning models in Python and R.
Transfer Learning for Image Classification of Various Dog Breeds (Devikar, 2016)	Image Classification in CNN has proven to be highly efficient, but it requires a large training data set and substantial time for training to achieve higher accuracy.	Use 20,000+ images and train models in CNN with hundreds of epochs.
TensorFlow, A System for Large-scale Machine Learning (Abadi, 2016)	Using larger internal cluster with GPU can lead to fewer steps and high accuracy of the Inception model.	Install GPU version of TensorFlow to shorten training time & improve accuracy.
A Case Study on TensorFlow and Artificial Neural Networks (Vivekanandan, 2017)	TensorFlow performs very well on recognition problems, and the performance can be further improved by having more iterations.	Run our CNN models in TensorFlow and train them with hundreds of iterations.

Figure 3.1 showing different things we have to go through for project development

Involvement of Mathematics:

The role of the mathematics is defined as below:

- Clearlythebaseofthemodeltrainingismathematicaldata,soweneedtocareful when we select thealgorithm.
- As for the sake of accurate predictions correct parameters and features should bechosen.
- Overfitting and underfitting of data should be checkedproperly.
- Uncertainty can be reason for the involvement ofmathematics.

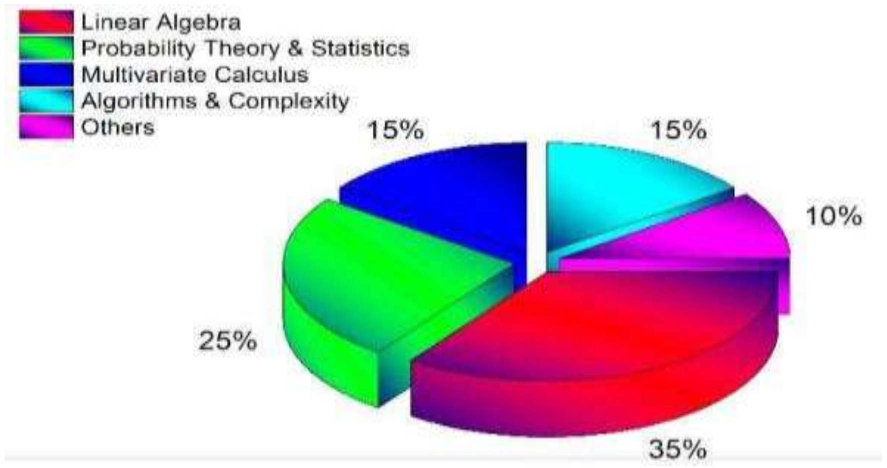


Figure 3.2 Math’s involvement

Functions(loss) involved for calculation ofmistakes:

- **Mean-Square-Error:**

A part of logistic regression that utilizes a method by calculating the mean of square of all error values obtained through an algorithm which tells us how much gap is b/w real and predicted observations.

$$\sum_{i=1}^{n} \frac{\left(w^T x(i) - y(i)\right)^2}{n}$$

Here w(t) = associated weight, x =expected observation and y =real observation.

- **Euclidean-Distance:**

It has been used to calculate the distance between the two possible outcomes in an algorithm. The inputs are the features whose values are put up in the formula of Euclidean distance. Euclidean space is also known as Pythagorean metric which is a capable metric in space.

$$dist((x, y), (a, b)) = \sqrt{(x - a)^2 + (y - b)^2}$$

Inthisformulaxandyarefeaturesofpoint1whereaandbarethefeatures of the point two.

- **ManhattanDistance:**

  Itisalsousedforthecalculationofdistancebetweentwooutcomesinalgorithms of machine learning with a slight difference from the Euclidean distance. This algorithmisusedforthequickassessmentofthedistancebetweentwooutcomes given as per thedata.

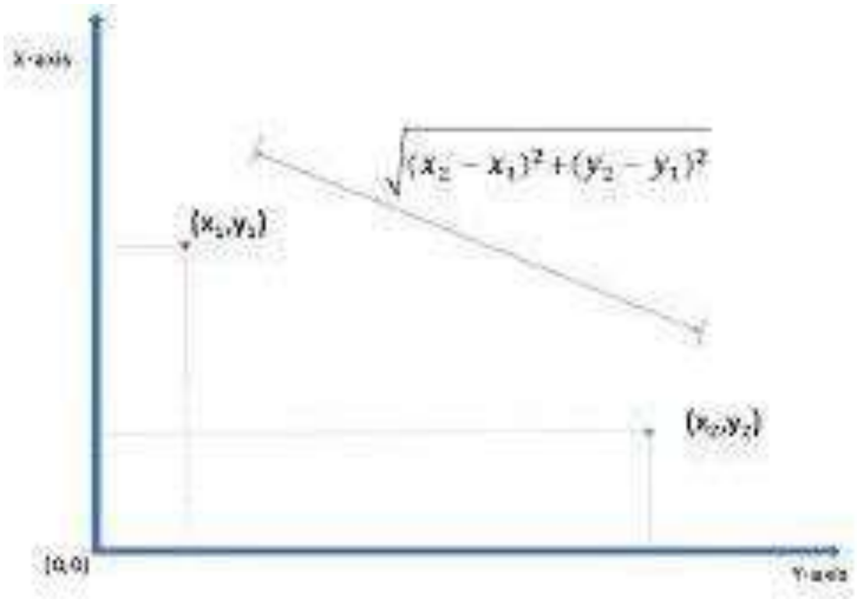


Figure 3.3 Formula for the distance

Steps involved in deep learning:

- For the purpose of training, validation and testing the dataset we have downloaded from the internet is brokendown.
- To find out the highlighted areas in the image we convert the image into black and whiteformat.
- Position of the data and encoding is necessary as because we have to detect a large number of breeds ofdogs.
- Here the model we have used is the inceptionmodel

Flow Chart of our deep learning model:

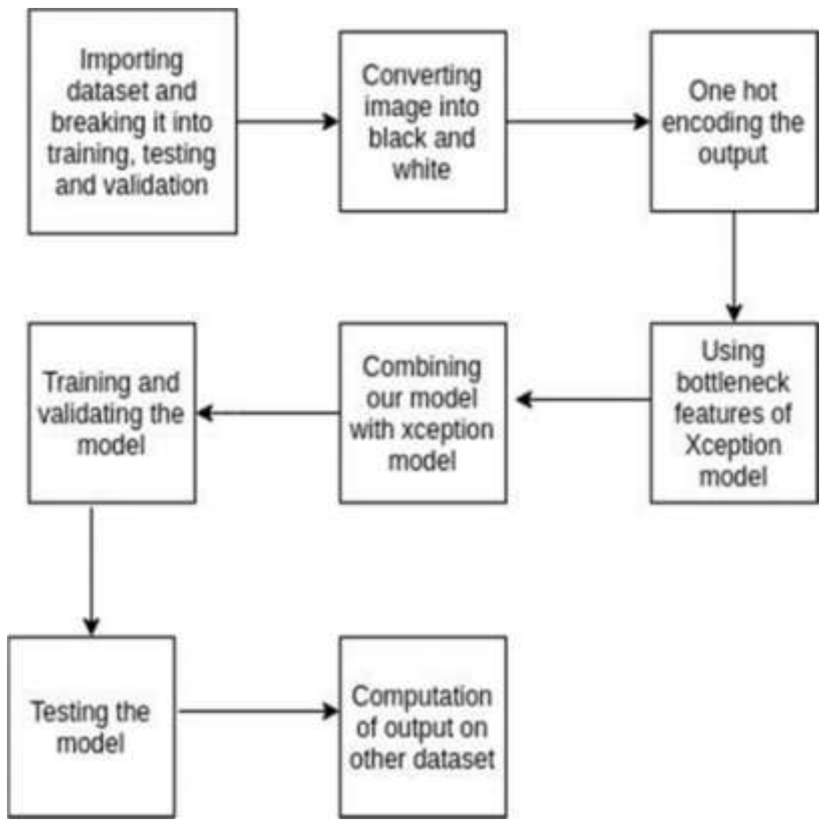

Importing dataset and breaking it into training, testing and validation
Converting image into black and white
One hot encoding the output
Using bottleneck features of Xception model
Combining our model with xception model
Training and validating the model
Testing the model
Computation of output on other dataset

Figure 3.4 Flow chart of our DL model

Algorithm:

In our algorithm we have used very advanced CNN. This design of the algorithmsuses the data and the depth knowledge of all the machines. CNN uses the advanced filters inneuralnetworkswhichareusedtorecognizesomehighlightedareasinanimageand thesewillberandomizedbytheuseofsomedistributedfunctionslikegaussian,normal and uniform, but we only need to place the filter on the image and we will get the required highlighted areas. There are also three layers as we have discussed earlier which are only dedicated for the purpose of pattern recognition in an image. Activation function play their role in neural networks aswell.

Basic elements for an algorithm:

Asithasbeendiscussedearlierfiltersarerequiredforextractionofparticularhighlighted areasbyconvolutionofthefast-movingimages.Filtersarefurtherclassifiedonthebasis how they detect pattern in an image like horizontal, vertical and in an inclined manner. So,itisquiteevidentthatthehorizontalwilldetecttheimageinahorizontalwayandso is true for rest of cases. To make it simple to understand we say that the filter is matrix which has designated weights for each empty row or column and when these weights are placed over an image these are multiplied with the corresponding image values. The resultant is passed to pooling layers to increase the depth effect in that resultant image. Even after that we have so many multiple filters for the patternrecognition.

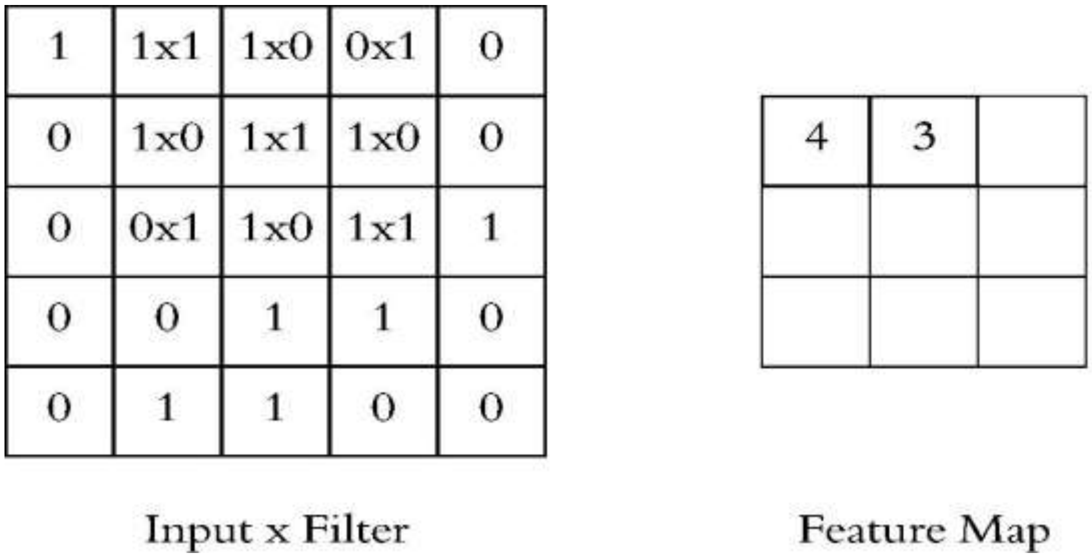


Figure 3.5 operation of mask over input image in convolution

Activation Function:

Itisusedtodetectthemostfeasibleandpossibleoutputasitstaskistoscaletheoutput orconversionaspertherequirementandasperourprojectthemachinewillbeableto detect the breed of the dog which has the highest probability as per the input dataset. We have relu activation function which can convert all the negative(-ve) valuesinto zeroes (0) and it is also used to dispose all the unnecessary data which can cause troublefortheclassifierfortheextractionofnecessarydata.Wehavedifferentvariants of the activation functions like SoftMax, sigmoid but for the most of the cases we use relu activation function. It has a great role when it comes to the classification of the breeds of the dogs. Only by just removal unwanted data from the dataset it helps a lot inincreasingtheaccuracyoftheclassifierandthetestingaswellasvalidationprevents the overfitting of the model that we want totrain.

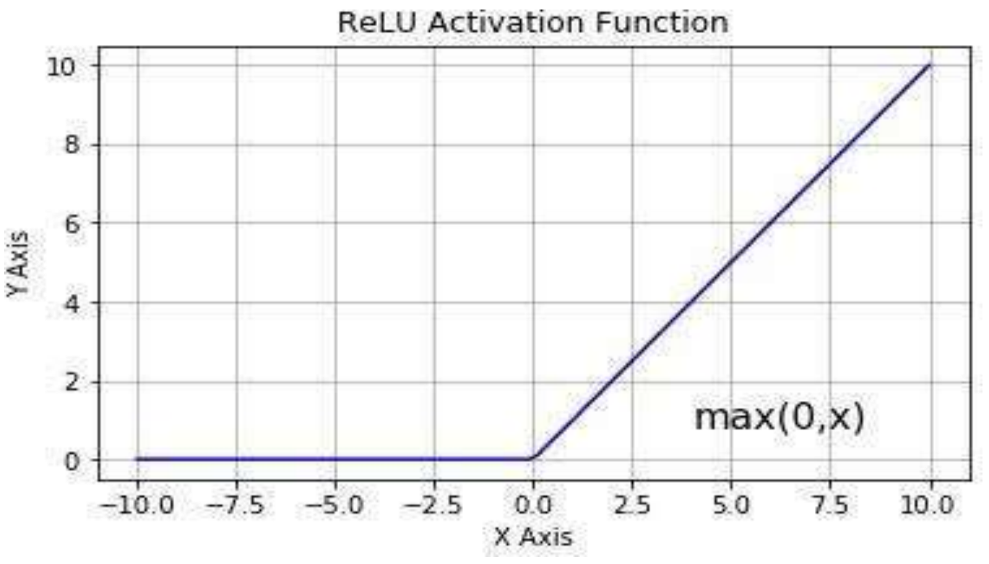


Figure 3.6 ReLU Activation Function

Convolution layers:

These layers have their own importance in the classification in the dataset and these layershelpinpreservingtheconnectionbetweentheneighboringpixelsasthevalueof one pixel is dependent upon the other whereas the case is very much different in sentimental analysis where no important information regarding the image is given. These layers are very much helpful in the case where pattern recognition is important even in the case of the complex patterns. Here inputs initialize the layer. These layers consist of strides which decides the movement of the filter in the required directionand there is also a kernel initializer which can generate filter of n*n from the public distribution function.

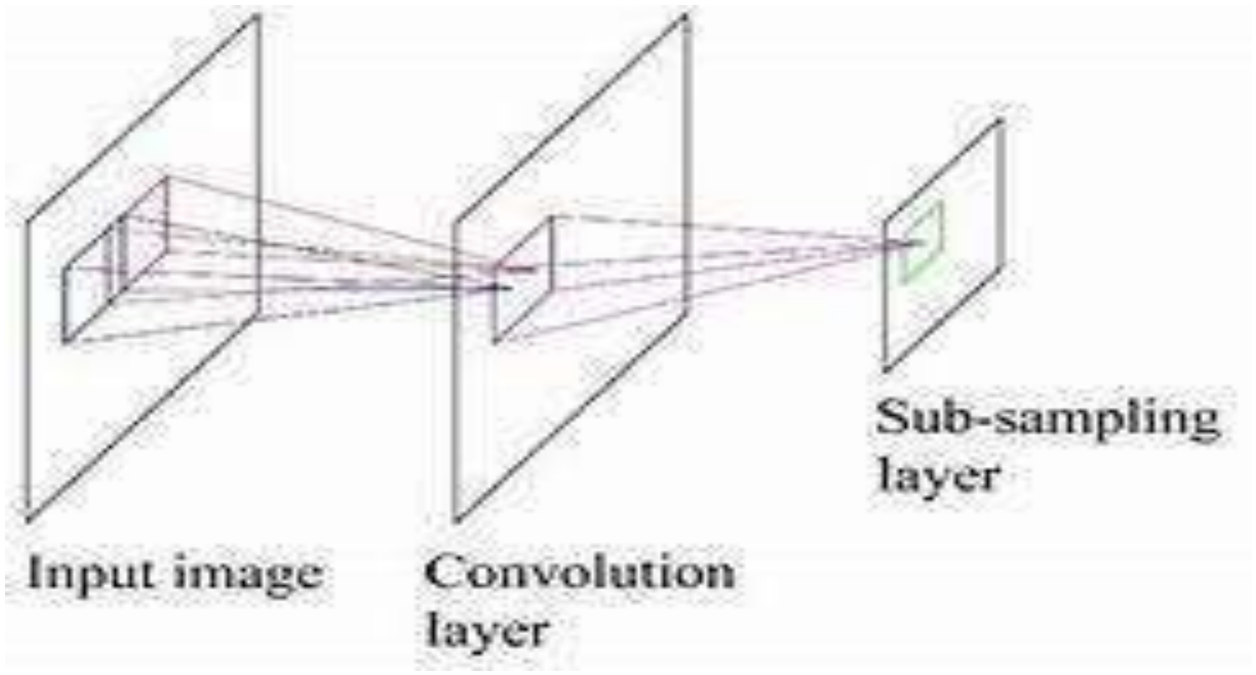


Figure 3.7 input image and it’s journey

For decreasing the height and width of the image insertion of pooling layer is done between the convolution layer and also there is extra advantage that the image will become more and more deep i.e depth effect due to more filters used. Pooling layers haveactuallymanytypesbutfortherequirementsourdatasethaswemostlyuseglobal max and global average pooling layer. The resultant image we get is full of random matrix so by if we want further any eliminations we can use multiple filters and in this case the global max pooling is used to extract some of the important features like size etc and global average pooling is used to take the average of all the effects that have been allowed by the global max pooling. Its basic purpose is regarding the dimensions oftheimageonthebasisofdifferentfiltersanditisusedjustbeforethesigmoidfunction and dense layer. These changes are very important as far the predictions of the model are taken into considerations as per accuracy point of view.

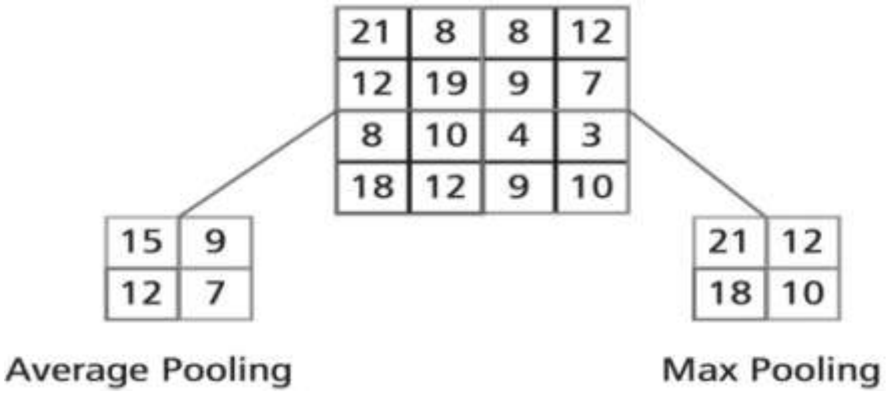
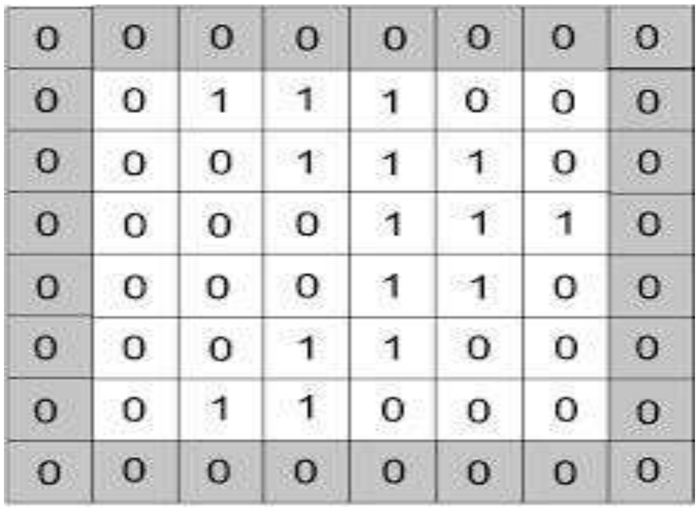


Figure 3.8 Difference between average pooling and Max Pooling

Strideisanothersignificanttermwhenitcomestothemovementofthefiltersandthere are the cases when the value of stride is not taken into consideration like when the convolutedimageandtheoriginalimageareofsamesizethenifthevalueofthestride is even one. Whether or not we should pad zeroes in the convoluted image or the requiredimageitstrictlydependsuponthefactthatwhatwewanttodowithdataset.In general, all the padding work depends upon the fact how easily we want to move the filter across the image and also, we become careless enough we can destroy the convoluted image and also, we can lose some information from that image. So, these factors are quite clear to determine the accuracy of the model we want totrain.



Feed Forward:

Itclearlytellsusaboutfeedingsomethingforwardandwhenwearetalkingaboutneural networks the data of one layer is forwarded to another to give us resultant output by making decisions on the basis of internal weights it possess and random distributions. After this resultant the only thing that makes a model successful is its accuracy so that is why we compare the actual value with the estimated value like root mean square or entropy etc. Evidently there will a lot of work due to calculations of largedataset.

Back propagation:

It consists of techniques like stochastic gradient descent for the purpose of error evaluation and its removal as much as much possible and it can also be done by the adjustments of the weights in between the layers. As there are lot of calculations involved so that is why we should work with CUDA graphics to improve the efficiency and accuracy of the model.

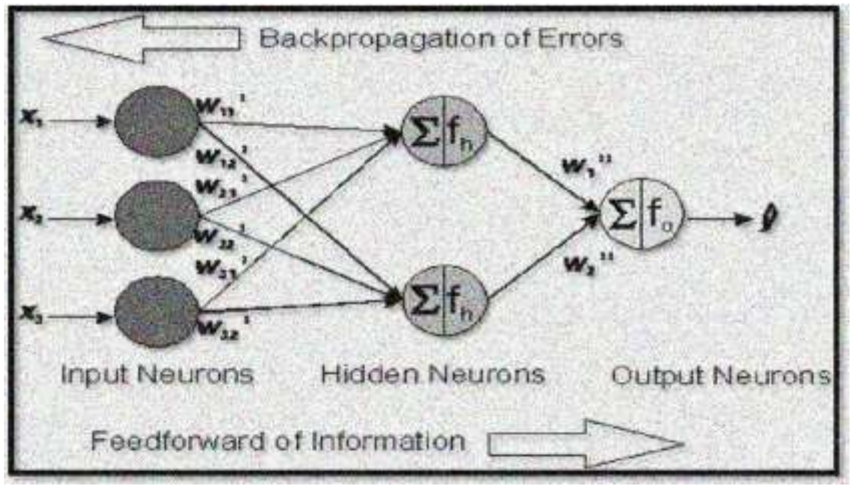
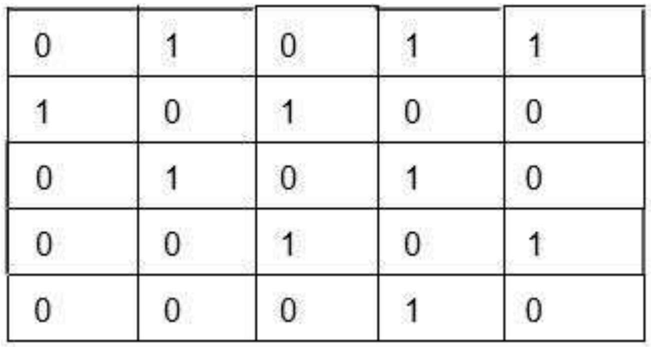


Figure 3.9 Feed Forward and Back Propagation

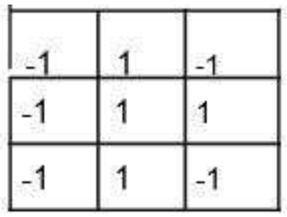
Mathematics application behind Convolutions:

In CNN, it involves the usingfiltering layers& the pool surfacefor highlighting certain specificmatchings inside of input picture. Now, we see an example of the mathematical part of CNN withthehelpofaverynormalpicture&simplefiltering surfaceconsisting of 0pad valueandthemoving from one pixel to other in the steps=1, that is the stride is1.



0	1	0	1	1
1	0	1	0	0
0	1	0	1	0
0	0	1	0	1
0	0	0	1	0

Is the matrix form of the Input Image



-1	1	-1
-1	1	1
-1	1	-1

Is the Filter

Now, when we apply the convolution layer, we place the filter over the input image and then we compute the result.

Calculationcorrespondingtotheresultforoneisasfollowandsimilarlywecancalculate for other.

Output=Relu[(-1)*0+(1)*1+(-1)*0+(-1)*1+(1)*(0)+(1)*1+(-1)*0+(1)*1+(- 1)*0]=2.

In this way we can calculate the value of other pixels by moving the filter by amount of stride over the image and the final convoluted image is obtained as follow:

2	-3	0
-2	2	-2
2	-2	2

Output of convolution

Various steps that are involved in the algorithms

At first, we will have dataset that contain training images, testing images andvalidation images. The model we are developing shall-be configured withfeatures that it contains the convolutional surface firstly&simply various characteristicsin the convolution layer will be the number of filters, padding type, the activation function, size of kernel initializer and theinputshapeoftheimage.Theprovidedstructureof image will bea three dimensionalbecausewearetaking a colored image. There are three layers in a colored image and those are (RGB) reddish layers,greenishlayers&bluishlayers.Heremaskweareusing,a3d one which is converted into two dimensions. The layers after the convolution layer known as maxpooling layers. This layer converts a picture from the larger size to a smaller size. Max pooling layer select the maximum out of the four pixel values and moves in the step of 2 pixels over the convoluted output of the previous layer. In this way, we will have multiple combinations of such one after other convolutional and maxpooling layers. It will be continuously incrementing the countregarding masks if it will be going inside deepest layer and hence will be increasing the depth of the image. Then we willsupply the training dataset tolearnthe data and the validating dataset that role of increasing the accuracy of our model and avoid the overfitting problem. Here, the data will be supplied in batches (foe example batches of 8 images etc.) and will predict the outputandthisisknownas“feedforward”.Alongwith initspredictionbycalculatingthedifferencebetweenactualandpredictedoutput.There are different error functions that we can use but here we will be using the cross categorical loss function because this function is mostly used for classification problems. After calculating the error, we will try to optimize our model using rmsprop optimizer. The rmsprop optimizer backpropagates while suggesting changes to the previous layers and thus resulting in optimizing the error function. Optimization of error functionwillbedonebycalculatingthegradientwhichspecifythechangeintheweights ofalayer.Thesestepswillbe repeatedagainandagainmanytimesandattheendwe will have a model trained with very much accurate weights at different layers of the model. when the training of the model is completed, the testing dataset will be used to test the model. Testing dataset will check the accuracy of model by checking the total numberofcorrectpredictionsmadebythemodeldividedbytotalnumberofpredictions, followed by multiplication by hundred. Our model will predict the probabilities that the dog in the image belong to different breeds. So, the breed with the highest probability will be the predicted output. After training to a good extentvalues of our modelwill be saved keeping the predictions in mind.Usingthismodelinthebackendinawebsiteoranapp,we can use that to check the breed of the dog in the inputimage.

Figure 3.10Catimage

Figure 3.11 Dogimage

TESTING PLAN FOR LEARNING THE CONCEPTS

Dataset is downloaded first:

First of all, we need our dataset. We will download the dataset from the web and then it is divided in three classes, those are training dataset, testing dataset & validation datasets. First set is required in learningofmodels required. Now we will take look on downloadinga set from a website.

Following is procedure that is required to obtaining ofdesiredset:

I am trying to classify Dogs vs Cats for our learning purpose and we are working on google colabs environment.

1. We start downloadingof set having the required pictures from websites and it is in compressed form.

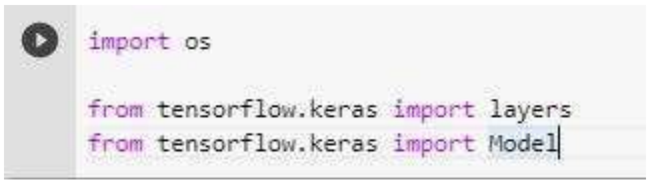
2. Data is decompressed using zipfile.ZipFilecommand.

3. Data is brought in particular folder using extractallcommand.

4. Download the inceptionV3 model fromkeras.applications.

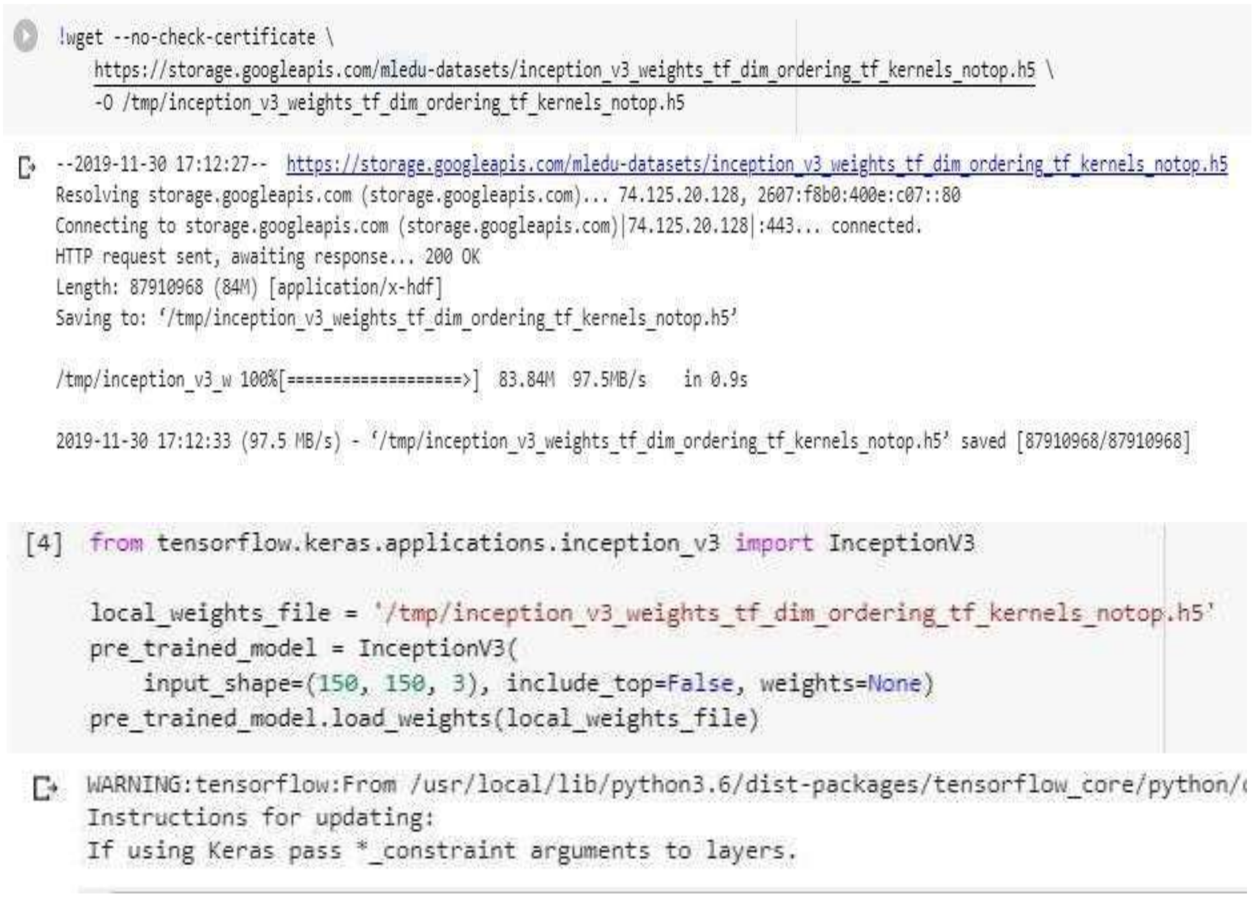
5. Load the pretrained weights of themodel.

Decompression the dataset is done using the zipfile and extractall command. For our actual project we will have to use google xception model later on.



```
import os

from tensorflow.keras import layers
from tensorflow.keras import Model
```

Figure 3.12 importing required modules



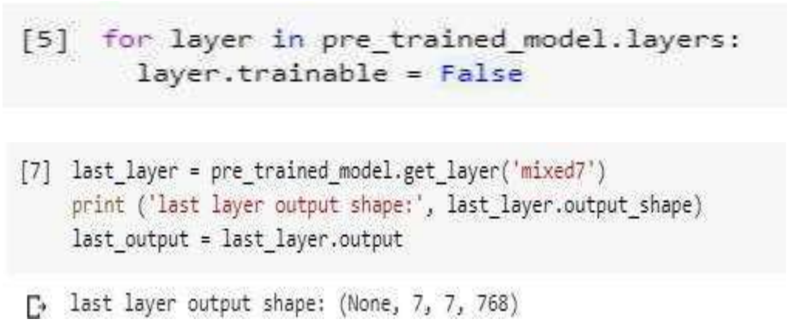
```
!wget --no-check-certificate \
    https://storage.googleapis.com/mledu-datasets/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5 \
    -O /tmp/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
```

```
--2019-11-30 17:12:27--  https://storage.googleapis.com/mledu-datasets/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.20.128, 2607:f8b0:400e:c07::80
Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.20.128|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 87910968 (84M) [application/x-hdf]
Saving to: '/tmp/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5'

/tmp/inception_v3_w 100%[===================>]  83.84M  97.5MB/s    in 0.9s

2019-11-30 17:12:33 (97.5 MB/s) - '/tmp/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5' saved [87910968/87910968]
```

```
[4] from tensorflow.keras.applications.inception_v3 import InceptionV3

    local_weights_file = '/tmp/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5'
    pre_trained_model = InceptionV3(
        input_shape=(150, 150, 3), include_top=False, weights=None)
    pre_trained_model.load_weights(local_weights_file)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_core/python/
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
```

Figure 3.13 importing inception_v3 model

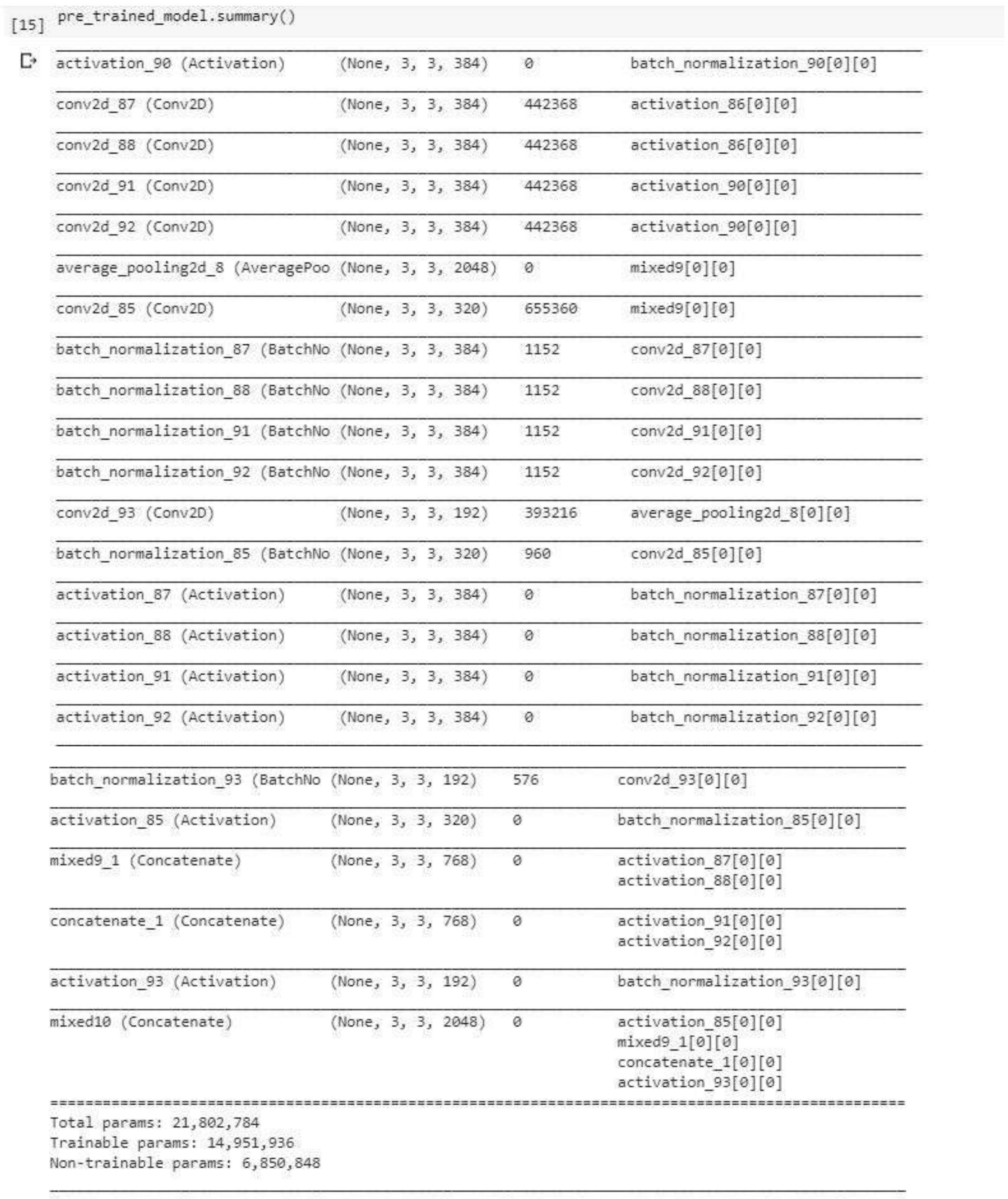


```
[5] for layer in pre_trained_model.layers:
      layer.trainable = False
```

```
[7] last_layer = pre_trained_model.get_layer('mixed7')
    print ('last layer output shape:', last_layer.output_shape)
    last_output = last_layer.output
```

```
last layer output shape: (None, 7, 7, 768)
```

Figure 3.14 deciding layer to join to our model



```
[15] pre_trained_model.summary()

 activation_90 (Activation)     (None, 3, 3, 384)    0           batch_normalization_90[0][0]
 __________________________________________________________________________________________________
 conv2d_87 (Conv2D)             (None, 3, 3, 384)    442368      activation_86[0][0]
 __________________________________________________________________________________________________
 conv2d_88 (Conv2D)             (None, 3, 3, 384)    442368      activation_86[0][0]
 __________________________________________________________________________________________________
 conv2d_91 (Conv2D)             (None, 3, 3, 384)    442368      activation_90[0][0]
 __________________________________________________________________________________________________
 conv2d_92 (Conv2D)             (None, 3, 3, 384)    442368      activation_90[0][0]
 __________________________________________________________________________________________________
 average_pooling2d_8 (AveragePoo (None, 3, 3, 2048)  0           mixed9[0][0]
 __________________________________________________________________________________________________
 conv2d_85 (Conv2D)             (None, 3, 3, 320)    655360      mixed9[0][0]
 __________________________________________________________________________________________________
 batch_normalization_87 (BatchNo (None, 3, 3, 384)   1152        conv2d_87[0][0]
 __________________________________________________________________________________________________
 batch_normalization_88 (BatchNo (None, 3, 3, 384)   1152        conv2d_88[0][0]
 __________________________________________________________________________________________________
 batch_normalization_91 (BatchNo (None, 3, 3, 384)   1152        conv2d_91[0][0]
 __________________________________________________________________________________________________
 batch_normalization_92 (BatchNo (None, 3, 3, 384)   1152        conv2d_92[0][0]
 __________________________________________________________________________________________________
 conv2d_93 (Conv2D)             (None, 3, 3, 192)    393216      average_pooling2d_8[0][0]
 __________________________________________________________________________________________________
 batch_normalization_85 (BatchNo (None, 3, 3, 320)   960         conv2d_85[0][0]
 __________________________________________________________________________________________________
 activation_87 (Activation)     (None, 3, 3, 384)    0           batch_normalization_87[0][0]
 __________________________________________________________________________________________________
 activation_88 (Activation)     (None, 3, 3, 384)    0           batch_normalization_88[0][0]
 __________________________________________________________________________________________________
 activation_91 (Activation)     (None, 3, 3, 384)    0           batch_normalization_91[0][0]
 __________________________________________________________________________________________________
 activation_92 (Activation)     (None, 3, 3, 384)    0           batch_normalization_92[0][0]
 __________________________________________________________________________________________________
__________________________________________________________________________________________________
batch_normalization_93 (BatchNo (None, 3, 3, 192)    576         conv2d_93[0][0]
__________________________________________________________________________________________________
activation_85 (Activation)      (None, 3, 3, 320)    0           batch_normalization_85[0][0]
__________________________________________________________________________________________________
mixed9_1 (Concatenate)          (None, 3, 3, 768)    0           activation_87[0][0]
                                                                 activation_88[0][0]
__________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 3, 3, 768)    0           activation_91[0][0]
                                                                 activation_92[0][0]
__________________________________________________________________________________________________
activation_93 (Activation)      (None, 3, 3, 192)    0           batch_normalization_93[0][0]
__________________________________________________________________________________________________
mixed10 (Concatenate)           (None, 3, 3, 2048)   0           activation_85[0][0]
                                                                 mixed9_1[0][0]
                                                                 concatenate_1[0][0]
                                                                 activation_93[0][0]
==================================================================================================
Total params: 21,802,784
Trainable params: 14,951,936
Non-trainable params: 6,850,848
__________________________________________________________________________________________________
```

Figure 3.15 inception_v3 model summary

In training the model and testing of model, data play a very crucial role. If data is not sufficient then it may lead to undesired results. Our model predicts the output is dependingonthedatasetweareusingandadatasetcontaininglargenumberofimages and variety and a greater number of datasets will result in a well-trained very good model.Dataisusedtomodifythepreinitializedvalues&improvisetheprecisionfeaturein the Neural Network prototype. It has distinguishable information corresponding to different breeds and the amount of each type of data is asfollow:

Training Dataset (Dataset for training our model):

Thisdataisusedfortrainingtheneuralnetworkmodelwith thefeed-forwardtechnique & the well-known backward-propagation algorithm. We will be using the trainingdata as well as validating datasetfor reductionof chances in overfitting case while training the neuralnetworkprototype. This dataset,has 20580picturesofthedogscorresponding to 120 total breeds with each breed having approx. 150 images and the dimension of images is not fixed or same. All images are colored images. Since the images are colored that means we have red, green and blue layers. We have a total of 120 classification classes and we will train our model to predict the probability of each one of the 120 categories. Out of the predicted probabilities, the breed with highest probabilitywillbeourmodel'spredictedoutput.Therearechancesthatthedatasetmight beable to overfit neural network and to avoid this uses image-augmentation. Dataset we have will be insufficient enough and because ofthisthemodelmayshowlessaccuracyandthat'swhyweneedto mergethisprototype,at the bottom will bepretrainedinception_v3forincrementinaccurate detection in themodel.After that we will evaluate our model's result depending upon the testing data about which we will discuss lateron.

Dog Breeds in Dataset

Afghan hound, American eskimo dog, Airedale terrier, Affenpinscher, Akita, Alaskan malamute, American foxhound, Anatolian-shepherd-dog, Australian-cattle-dog, American staffordshire terrier, Australian-shepherd, , Basset hound, Australian-terrier, Basenji, Beagle, Beauceron, American water Bedlington-terrier, Belgian-tervuren, Belgian-malinois, spaniel, Belgian-sheepdog, Bernese-mountain dog, Black & tan- coonhound, Bloodhound, Black-russian terrier, Bichon frise, Bluetick-coonhound, Border terrier, Boston terrier, Border collie, Bouvier des flandres, Boykin spaniel, Brussels griffon, Borzoi, Briard, Brittany, Bull-terrier, Chesapeake bay retriever, Bulldog, Cairn terrier, Cardigan welsh corgi, Canaan-dog, Cane-corso, Chihuahua, Curly-coatedretriever,Bullmastiff,Chinesecrested,Chineseshar-pei,Clumberspaniel, Collie, Dachshund, Cocker spaniel, Doberman pinscher, English cocker spaniel, Dalmatian, English springer spaniel, English setter, English toy spaniel, Field spaniel, Boxer, Bearded collie, Entlebucher mountain dog, Icelandic sheepdog, Pembroke welsh corgi, German pinscher, Flat-coated retriever, Maltese, French bulldog, German shorthaired pointer, Gordon setter, German wirehaired pointer, German shepherddog, Giant schnauzer, Greater swiss mountain dog, Golden retriever , Great dane, Greyhound, Irish red and white setter, Ibizan hound, Irish setter, Irish water spaniel, Irish terrier, Irish wolfhound, Japanese chin, Italian greyhound, Keeshond, Komondor, Kerry blue terrier, Kuvasz, Labrador retriever, Lhasa apso, Lakeland terrier,Norwegian buhund, Manchester terrier, Neapolitan mastiff, Newfoundland, Xoloitzcuintli, Norfolk terrier, Norwegian elkhound, Mastiff, Nova scotia duck tolling retriever, Miniature schnauzer, Norwich terrier, Papillon, Parson russell terrier, Otterhound, Pekingese, Pharaoh hound, Pomeranian, Petit basset griffon vendeen, Poodle, Silky terrier, Portuguese water dog, Wirehaired pointing griffon, Smooth-fox terrier, Welsh-springer spaniel, Yorkshire terrier Tibetan mastiff. Above mentioned are the 120 different dog breeds on which our model is trained and which our model canpredict.

Figure 3.16 Dog image from dataset

Validation dataset (Dataset for validating our model):

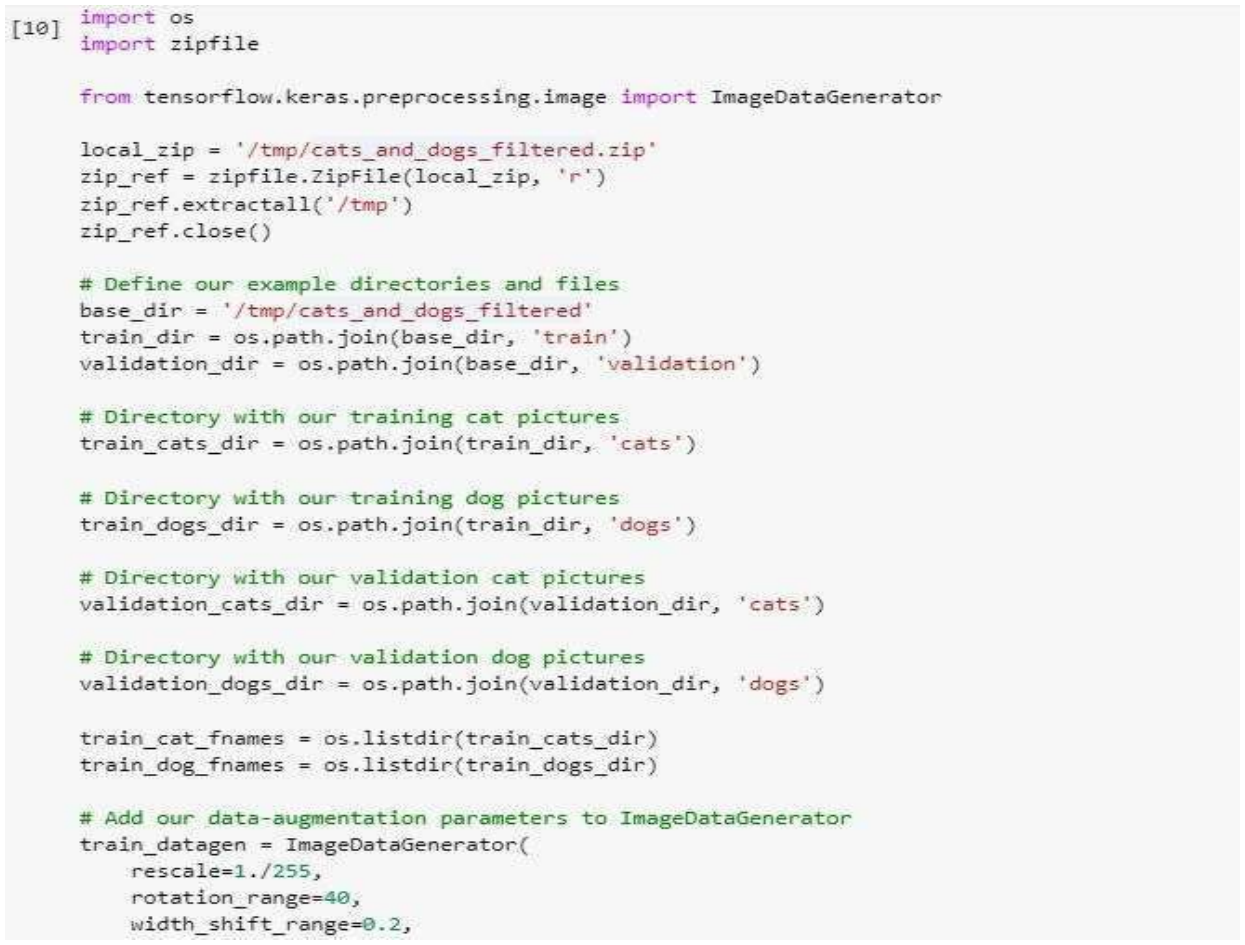
This dataset is used for the purpose of validating the data along with the dataset used fortrainingpurpose.Thesizeofthevalidationdatasetisgenerallysmallascompareto learning set & this dataset is in usage for problem of overfitting cases in deep learning. The maximum of all the probabilities represent the predicted output breed. So, as we try for the correction of neural network, whichcantellus about losses in the given set along with general loss. When the time comes when thereis increment in previously discussed loss, in this case it will adjust values of each layer of our neural network model andatthattimewewillhavethebestweights,wecanhavemoreaccuracyandtheover fitting will beless.

Figure 3.17 images from training dataset

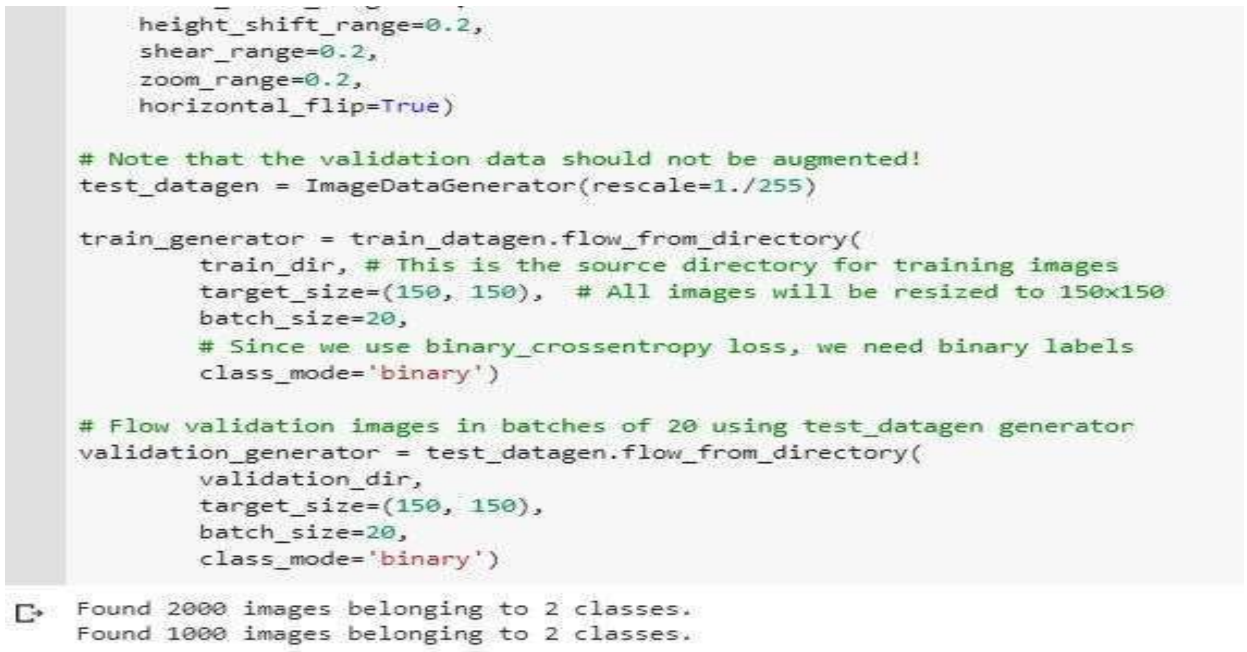
Testing dataset (Dataset for testing our model):

This set known as dataset that will be in usagefor final predictionof accuracy'sextent in neural network. A neural network is said to be good if it predicts high accuracy else, we need to check for the over fitting of data and along with that we need to increase the validation dataset. In testing dataset, we have few images for testing the prediction of all 120 breeds of dogs. We will input the dataset& check the count of the total judgement's accuracyasnumerator & totality of judgements as denominator, multiplyit with100. Note:in casethedatasethasaugmentationthenthismightleadclassificationtoincorrectjudgements. So, one option can be training our neural network with training dataset alongwithimageaugmentationsalso.Thetaskoftestingourmodelmaybeeasy withan ordinary brain of computer(CPU)the main challenge istrainingourmodelismorecomplex asitinvolvesalotofcomputationsdependinguponfeed-forward&back-propagation techniques that are used in there. So, we need graphic processing unit (GPU) for such heavyandverycomplextaskincludinglotsofcalculations.WithGPUtheneuralnetwork modelgetstrainedatagreaterspeedascomparetowhendonewithCPUandthisisa lot better. The larger models such as google inception, xception and resnet etc. took largenumberofdaystogettrainedongraphicprocessingunit(GPU)andtheirweights are well adjusted for providing higher accuracy. As compare to Central Processing Unit(CPU), a graphic processing unit (GPU) is costly to buy and that's why we will be using google colabs environment to train our model online. In google colabs, we can directly import our dataset to our environment using "!wget" command and we can importourinceptionmodealso.Itissimilartojupyternotebookenvironmentbutprovide largeamountofonlineRAMandGPU.Thus,wewilltrainourmodelontrainingdataset as we provide the training dataset as an input to convolution neural networks model. Oneofthemajorrolesforthedevelopmentofdeepneuralnetworkshasplayed task of testing of our model. Criteria in training-set& testing-dataset is also a main principle behind all working of deep reinforcement learning. The agent interacts with the environment and based on that interaction a reward is given. If we get a positive rewardthenthisisgreat.Sometimewealsoallowustohavenegativereward.Wehave sufficientenoughtrainingdatasetthatcontainsaround20000imageswhichareenough for training the model and testing the model's performance and at last for calculating the accuracy of our neural network model. Dataset we are using for testing our neural network model is somewhat smaller, so we need to use various image augmentation techniques for increasing our testing-dataset & thus getting great informationaboutthe neural network-model. The efficiency of the agent is observed from the actions toward environment and the award what was received. If the reward is positive then it is fantastic to have it. If the reward is negative then it is also acceptable sometimes. The datasetfortestingourneuralnetworkmodelisenlargedtosomedegreetohavealarge enough dataset for testing our model and thus it helps us in getting a good enough understanding of our model and provide a lot of information about ourmodel.

Figure 3.18 images from training dataset



```
[10] import os
     import zipfile

     from tensorflow.keras.preprocessing.image import ImageDataGenerator

     local_zip = '/tmp/cats_and_dogs_filtered.zip'
     zip_ref = zipfile.ZipFile(local_zip, 'r')
     zip_ref.extractall('/tmp')
     zip_ref.close()

     # Define our example directories and files
     base_dir = '/tmp/cats_and_dogs_filtered'
     train_dir = os.path.join(base_dir, 'train')
     validation_dir = os.path.join(base_dir, 'validation')

     # Directory with our training cat pictures
     train_cats_dir = os.path.join(train_dir, 'cats')

     # Directory with our training dog pictures
     train_dogs_dir = os.path.join(train_dir, 'dogs')

     # Directory with our validation cat pictures
     validation_cats_dir = os.path.join(validation_dir, 'cats')

     # Directory with our validation dog pictures
     validation_dogs_dir = os.path.join(validation_dir, 'dogs')

     train_cat_fnames = os.listdir(train_cats_dir)
     train_dog_fnames = os.listdir(train_dogs_dir)

     # Add our data-augmentation parameters to ImageDataGenerator
     train_datagen = ImageDataGenerator(
         rescale=1./255,
         rotation_range=40,
         width_shift_range=0.2,
```



```
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True)

# Note that the validation data should not be augmented!
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        train_dir, # This is the source directory for training images
        target_size=(150, 150),  # All images will be resized to 150x150
        batch_size=20,
        # Since we use binary_crossentropy loss, we need binary labels
        class_mode='binary')

# Flow validation images in batches of 20 using test_datagen generator
validation_generator = test_datagen.flow_from_directory(
        validation_dir,
        target_size=(150, 150),
        batch_size=20,
        class_mode='binary')
```

```
Found 2000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
```

Figure 3.19 showing number of images and classes

Chapter-4

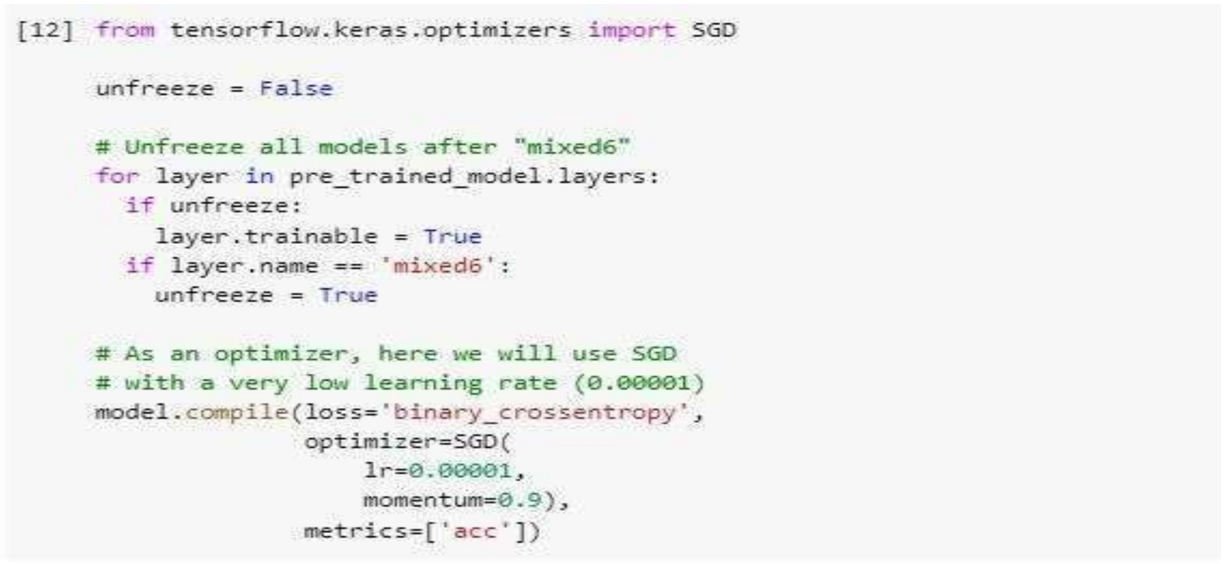
PERFORMANCE ANALYSIS AND RESULT

Analysis:

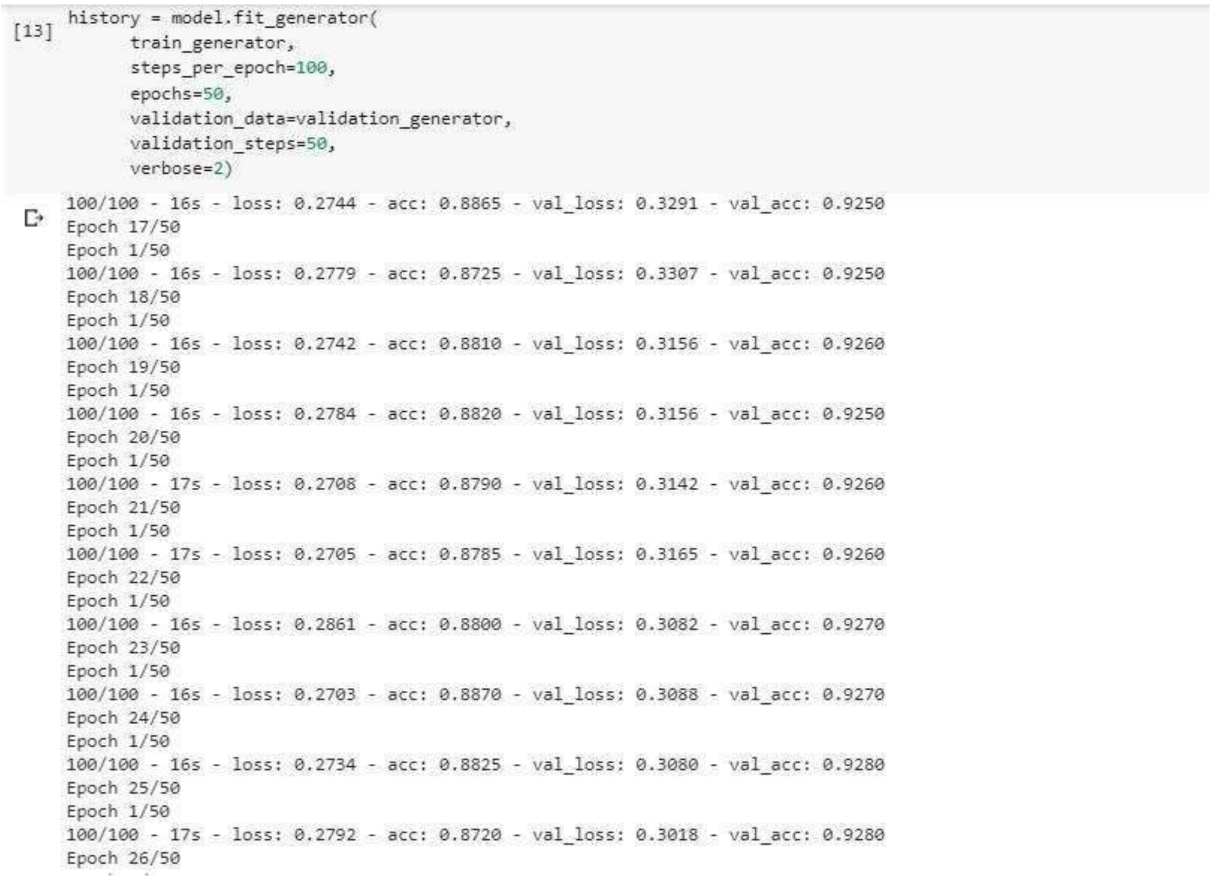
Ourmodelshowstheresultsdependinguponthedatathatwewillbeusingfortraining, testing and validating our model. The data being used for training should be enlarged by using augmentation techniques while our model is being trained. There are high chancesthatourneuralnetworkmodelcannotgive us accuratedog’s typeifpicturesare flipped. To overcome this, we randomly flip and rotate some of the images from dataset at some angular variation& then aimfor training our neural network-model &by the help of that the validating of our neural network model keeps going on. The model’s accuracy depends only on the type and the variety of training data that we are using for training our model and the testing data that we are using for testing ourmodel.

Accuracy

Our model consists of convolution neural networks and these types of model require a large dataset but the dataset we have is very small as compare what we need. As we have only about 150 images per breed. And that’s why the accuracy of our neural networkmodelafterbeingtrainedisabout2percentwhichisnotacceptableasit’squite lowincase of deep-learning-model. Because of availability of sufficient data, theprecision ofourmodelisverylow.Itisbecauseofaccuracythatwecometoknowabouthowwell ourweightsaresetinourneuralnetworkmodelatdifferentlayersofourneuralnetwork. Thefieldofdeeplearningisverycomplexandhereweusehyperparametersandvalue of these hyperparameters is very hard to decide initially. It is very hard even for the experts to know the initial values of these hyperparameters so that at the end we can get a very high accuracy. The different hyperparameters on which the accuracy depends are like initial weights at each layer, number of epochs (how many times a batchofimagemovesforwardandbackward)andlearningrateetc.Supposeifall weights are initialized as1&thisshall give us quitelowprecision in thetraining-model and because of this usage ofrandomized weights-distribution for example normal-distribution, gaussian distribution etc.

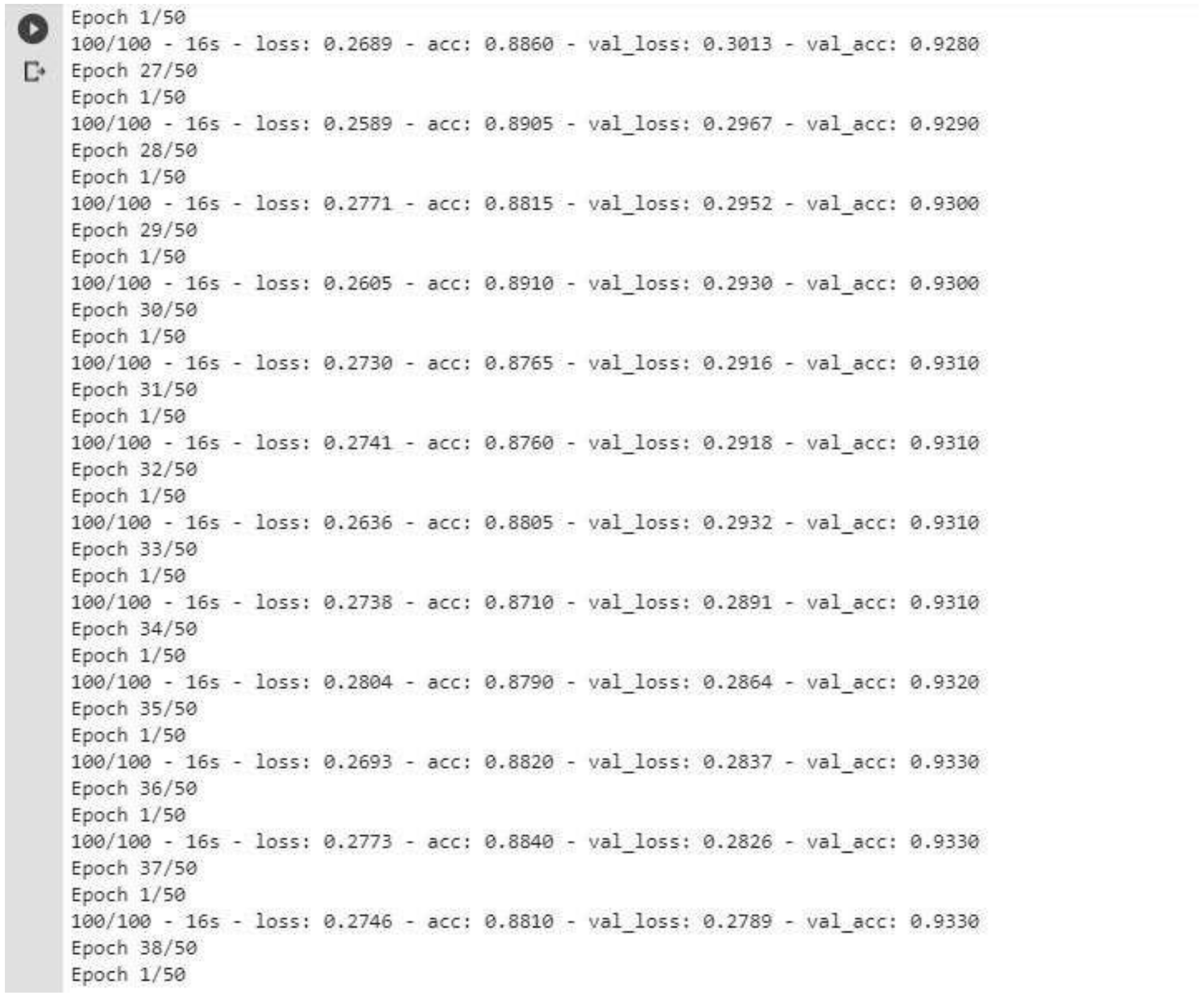


```
[12] from tensorflow.keras.optimizers import SGD

     unfreeze = False

     # Unfreeze all models after "mixed6"
     for layer in pre_trained_model.layers:
       if unfreeze:
         layer.trainable = True
       if layer.name == 'mixed6':
         unfreeze = True

     # As an optimizer, here we will use SGD
     # with a very low learning rate (0.00001)
     model.compile(loss='binary_crossentropy',
                   optimizer=SGD(
                       lr=0.00001,
                       momentum=0.9),
                   metrics=['acc'])
```



```
[13] history = model.fit_generator(
           train_generator,
           steps_per_epoch=100,
           epochs=50,
           validation_data=validation_generator,
           validation_steps=50,
           verbose=2)
```

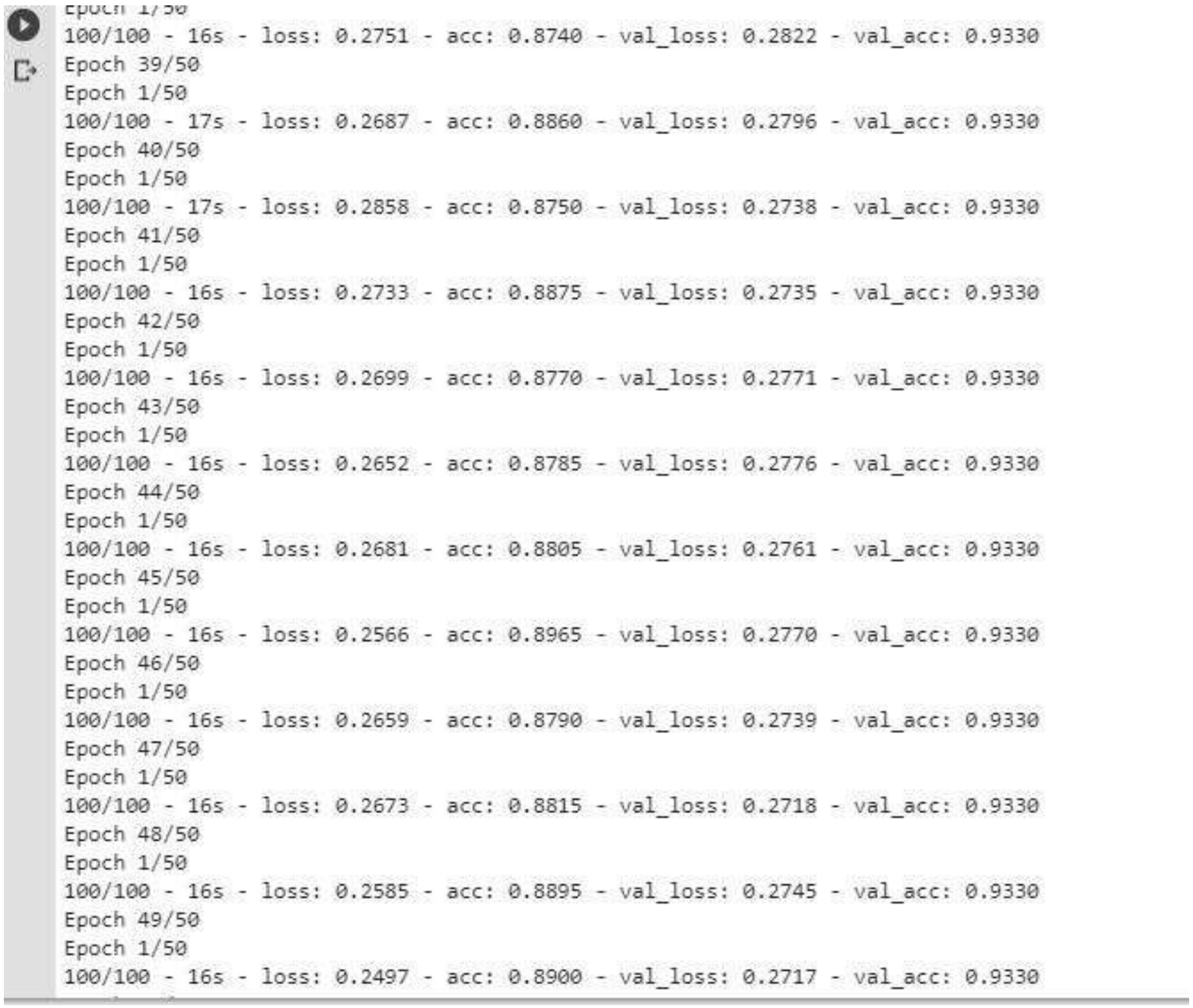
```
100/100 - 16s - loss: 0.2744 - acc: 0.8865 - val_loss: 0.3291 - val_acc: 0.9250
Epoch 17/50
Epoch 1/50
100/100 - 16s - loss: 0.2779 - acc: 0.8725 - val_loss: 0.3307 - val_acc: 0.9250
Epoch 18/50
Epoch 1/50
100/100 - 16s - loss: 0.2742 - acc: 0.8810 - val_loss: 0.3156 - val_acc: 0.9260
Epoch 19/50
Epoch 1/50
100/100 - 16s - loss: 0.2784 - acc: 0.8820 - val_loss: 0.3156 - val_acc: 0.9250
Epoch 20/50
Epoch 1/50
100/100 - 17s - loss: 0.2708 - acc: 0.8790 - val_loss: 0.3142 - val_acc: 0.9260
Epoch 21/50
Epoch 1/50
100/100 - 17s - loss: 0.2705 - acc: 0.8785 - val_loss: 0.3165 - val_acc: 0.9260
Epoch 22/50
Epoch 1/50
100/100 - 16s - loss: 0.2861 - acc: 0.8800 - val_loss: 0.3082 - val_acc: 0.9270
Epoch 23/50
Epoch 1/50
100/100 - 16s - loss: 0.2703 - acc: 0.8870 - val_loss: 0.3088 - val_acc: 0.9270
Epoch 24/50
Epoch 1/50
100/100 - 16s - loss: 0.2734 - acc: 0.8825 - val_loss: 0.3080 - val_acc: 0.9280
Epoch 25/50
Epoch 1/50
100/100 - 17s - loss: 0.2792 - acc: 0.8720 - val_loss: 0.3018 - val_acc: 0.9280
Epoch 26/50
```



```
Epoch 1/50
100/100 - 16s - loss: 0.2689 - acc: 0.8860 - val_loss: 0.3013 - val_acc: 0.9280
Epoch 27/50
Epoch 1/50
100/100 - 16s - loss: 0.2589 - acc: 0.8905 - val_loss: 0.2967 - val_acc: 0.9290
Epoch 28/50
Epoch 1/50
100/100 - 16s - loss: 0.2771 - acc: 0.8815 - val_loss: 0.2952 - val_acc: 0.9300
Epoch 29/50
Epoch 1/50
100/100 - 16s - loss: 0.2605 - acc: 0.8910 - val_loss: 0.2930 - val_acc: 0.9300
Epoch 30/50
Epoch 1/50
100/100 - 16s - loss: 0.2730 - acc: 0.8765 - val_loss: 0.2916 - val_acc: 0.9310
Epoch 31/50
Epoch 1/50
100/100 - 16s - loss: 0.2741 - acc: 0.8760 - val_loss: 0.2918 - val_acc: 0.9310
Epoch 32/50
Epoch 1/50
100/100 - 16s - loss: 0.2636 - acc: 0.8805 - val_loss: 0.2932 - val_acc: 0.9310
Epoch 33/50
Epoch 1/50
100/100 - 16s - loss: 0.2738 - acc: 0.8710 - val_loss: 0.2891 - val_acc: 0.9310
Epoch 34/50
Epoch 1/50
100/100 - 16s - loss: 0.2804 - acc: 0.8790 - val_loss: 0.2864 - val_acc: 0.9320
Epoch 35/50
Epoch 1/50
100/100 - 16s - loss: 0.2693 - acc: 0.8820 - val_loss: 0.2837 - val_acc: 0.9330
Epoch 36/50
Epoch 1/50
100/100 - 16s - loss: 0.2773 - acc: 0.8840 - val_loss: 0.2826 - val_acc: 0.9330
Epoch 37/50
Epoch 1/50
100/100 - 16s - loss: 0.2746 - acc: 0.8810 - val_loss: 0.2789 - val_acc: 0.9330
Epoch 38/50
Epoch 1/50
```

Fig 4(1) Training-accuracy 88 %in training-data

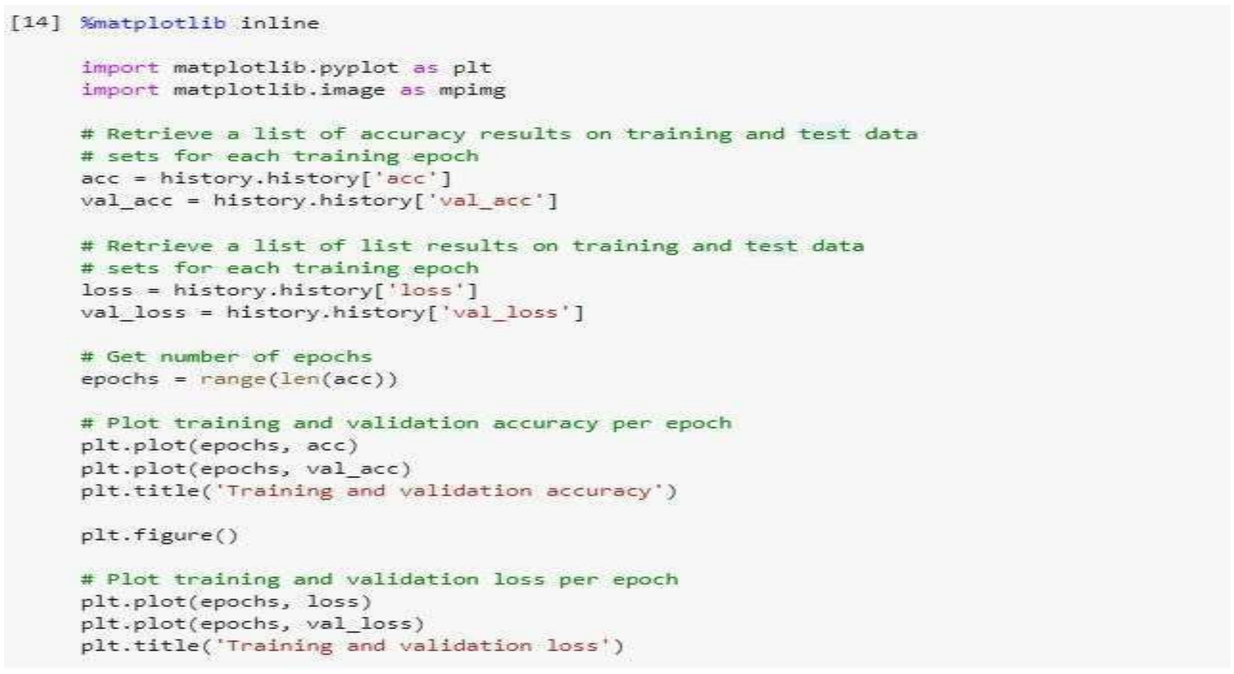
Fig 4(2) Image from training-data



```
Epoch 1/50
100/100 - 16s - loss: 0.2751 - acc: 0.8740 - val_loss: 0.2822 - val_acc: 0.9330
Epoch 39/50
Epoch 1/50
100/100 - 17s - loss: 0.2687 - acc: 0.8860 - val_loss: 0.2796 - val_acc: 0.9330
Epoch 40/50
Epoch 1/50
100/100 - 17s - loss: 0.2858 - acc: 0.8750 - val_loss: 0.2738 - val_acc: 0.9330
Epoch 41/50
Epoch 1/50
100/100 - 16s - loss: 0.2733 - acc: 0.8875 - val_loss: 0.2735 - val_acc: 0.9330
Epoch 42/50
Epoch 1/50
100/100 - 16s - loss: 0.2699 - acc: 0.8770 - val_loss: 0.2771 - val_acc: 0.9330
Epoch 43/50
Epoch 1/50
100/100 - 16s - loss: 0.2652 - acc: 0.8785 - val_loss: 0.2776 - val_acc: 0.9330
Epoch 44/50
Epoch 1/50
100/100 - 16s - loss: 0.2681 - acc: 0.8805 - val_loss: 0.2761 - val_acc: 0.9330
Epoch 45/50
Epoch 1/50
100/100 - 16s - loss: 0.2566 - acc: 0.8965 - val_loss: 0.2770 - val_acc: 0.9330
Epoch 46/50
Epoch 1/50
100/100 - 16s - loss: 0.2659 - acc: 0.8790 - val_loss: 0.2739 - val_acc: 0.9330
Epoch 47/50
Epoch 1/50
100/100 - 16s - loss: 0.2673 - acc: 0.8815 - val_loss: 0.2718 - val_acc: 0.9330
Epoch 48/50
Epoch 1/50
100/100 - 16s - loss: 0.2585 - acc: 0.8895 - val_loss: 0.2745 - val_acc: 0.9330
Epoch 49/50
Epoch 1/50
100/100 - 16s - loss: 0.2497 - acc: 0.8900 - val_loss: 0.2717 - val_acc: 0.9330
```

Fig 4(3) Training-accuracy 89 %in training-data

Fig 4(4) image from training-data



```
[14] %matplotlib inline

     import matplotlib.pyplot as plt
     import matplotlib.image as mpimg

     # Retrieve a list of accuracy results on training and test data
     # sets for each training epoch
     acc = history.history['acc']
     val_acc = history.history['val_acc']

     # Retrieve a list of list results on training and test data
     # sets for each training epoch
     loss = history.history['loss']
     val_loss = history.history['val_loss']

     # Get number of epochs
     epochs = range(len(acc))

     # Plot training and validation accuracy per epoch
     plt.plot(epochs, acc)
     plt.plot(epochs, val_acc)
     plt.title('Training and validation accuracy')

     plt.figure()

     # Plot training and validation loss per epoch
     plt.plot(epochs, loss)
     plt.plot(epochs, val_loss)
     plt.title('Training and validation loss')
```

```
Text(0.5, 1.0, 'Training and validation loss')
```

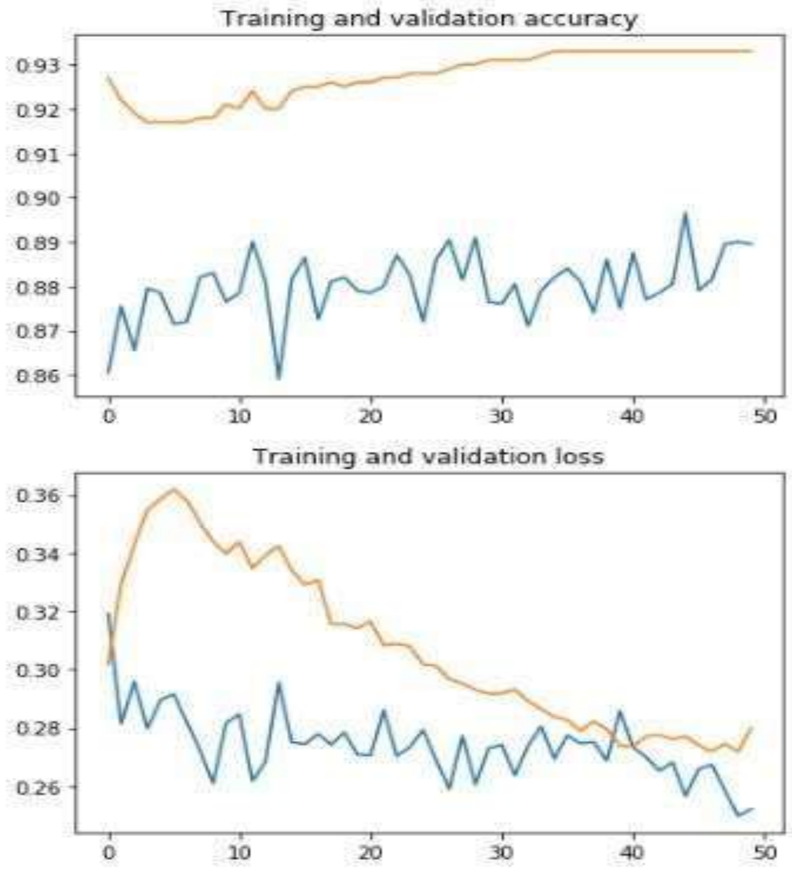
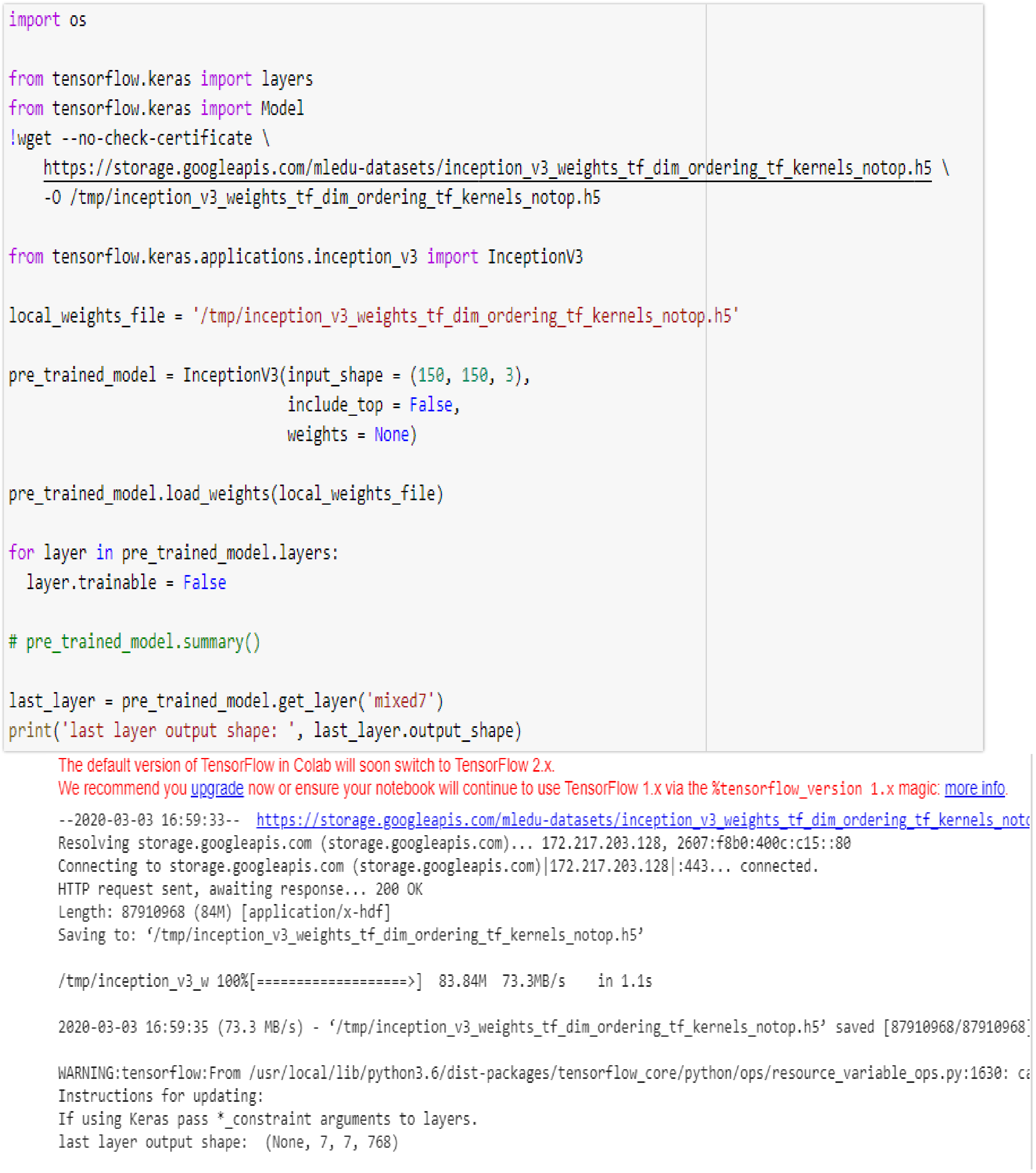


Fig 4(5) Plot showing training and validation accuracy and loss

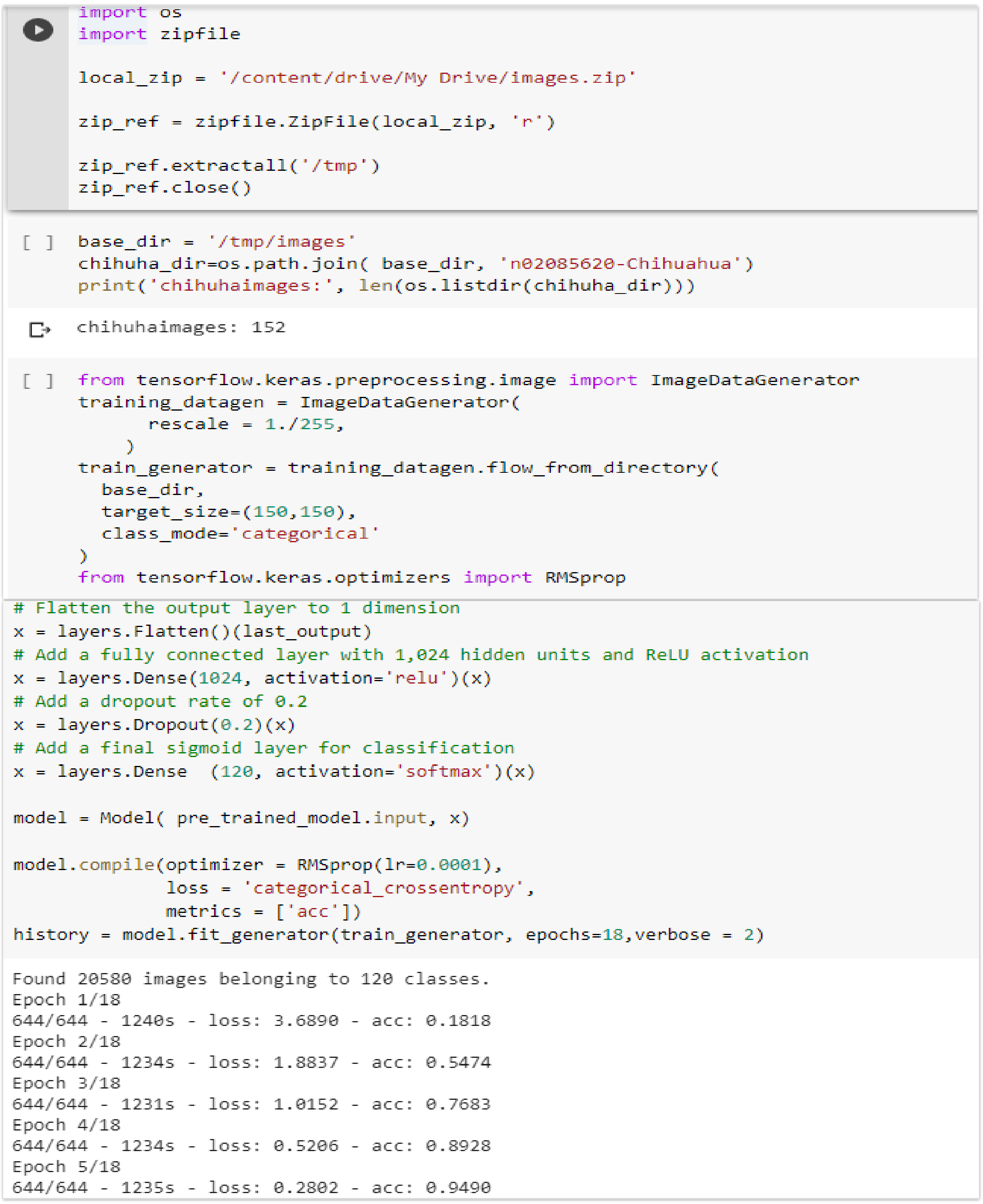
Final output:



```
import os

from tensorflow.keras import layers
from tensorflow.keras import Model
!wget --no-check-certificate \
    https://storage.googleapis.com/mledu-datasets/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5 \
    -O /tmp/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5

from tensorflow.keras.applications.inception_v3 import InceptionV3

local_weights_file = '/tmp/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5'

pre_trained_model = InceptionV3(input_shape = (150, 150, 3),
                                include_top = False,
                                weights = None)

pre_trained_model.load_weights(local_weights_file)

for layer in pre_trained_model.layers:
  layer.trainable = False

# pre_trained_model.summary()

last_layer = pre_trained_model.get_layer('mixed7')
print('last layer output shape: ', last_layer.output_shape)
```

```
The default version of TensorFlow in Colab will soon switch to TensorFlow 2.x.
We recommend you upgrade now or ensure your notebook will continue to use TensorFlow 1.x via the %tensorflow_version 1.x magic: more info.
--2020-03-03 16:59:33--  https://storage.googleapis.com/mledu-datasets/inception_v3_weights_tf_dim_ordering_tf_kernels_noto
Resolving storage.googleapis.com (storage.googleapis.com)... 172.217.203.128, 2607:f8b0:400c:c15::80
Connecting to storage.googleapis.com (storage.googleapis.com)|172.217.203.128|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 87910968 (84M) [application/x-hdf]
Saving to: '/tmp/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5'

/tmp/inception_v3_w 100%[===================>]  83.84M  73.3MB/s    in 1.1s

2020-03-03 16:59:35 (73.3 MB/s) - '/tmp/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5' saved [87910968/87910968]

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_core/python/ops/resource_variable_ops.py:1630: ca
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
last layer output shape:  (None, 7, 7, 768)
```

Fig 4(6) code



```
import os
import zipfile

local_zip = '/content/drive/My Drive/images.zip'

zip_ref = zipfile.ZipFile(local_zip, 'r')

zip_ref.extractall('/tmp')
zip_ref.close()
```

```
[ ] base_dir = '/tmp/images'
    chihuha_dir=os.path.join( base_dir, 'n02085620-Chihuahua')
    print('chihuhaimages:', len(os.listdir(chihuha_dir)))
```

```
chihuhaimages: 152
```

```
[ ] from tensorflow.keras.preprocessing.image import ImageDataGenerator
    training_datagen = ImageDataGenerator(
          rescale = 1./255,
        )
    train_generator = training_datagen.flow_from_directory(
      base_dir,
      target_size=(150,150),
      class_mode='categorical'
    )
    from tensorflow.keras.optimizers import RMSprop
```

```
# Flatten the output layer to 1 dimension
x = layers.Flatten()(last_output)
# Add a fully connected layer with 1,024 hidden units and ReLU activation
x = layers.Dense(1024, activation='relu')(x)
# Add a dropout rate of 0.2
x = layers.Dropout(0.2)(x)
# Add a final sigmoid layer for classification
x = layers.Dense  (120, activation='softmax')(x)

model = Model( pre_trained_model.input, x)

model.compile(optimizer = RMSprop(lr=0.0001),
              loss = 'categorical_crossentropy',
              metrics = ['acc'])
history = model.fit_generator(train_generator, epochs=18,verbose = 2)
```

```
Found 20580 images belonging to 120 classes.
Epoch 1/18
644/644 - 1240s - loss: 3.6890 - acc: 0.1818
Epoch 2/18
644/644 - 1234s - loss: 1.8837 - acc: 0.5474
Epoch 3/18
644/644 - 1231s - loss: 1.0152 - acc: 0.7683
Epoch 4/18
644/644 - 1234s - loss: 0.5206 - acc: 0.8928
Epoch 5/18
644/644 - 1235s - loss: 0.2802 - acc: 0.9490
```

Fig 4(7) code

Dog Breed(images) in Dataset

Figure 3.17 Chihuahua_

Figure 3.18 Japanese-spaniel

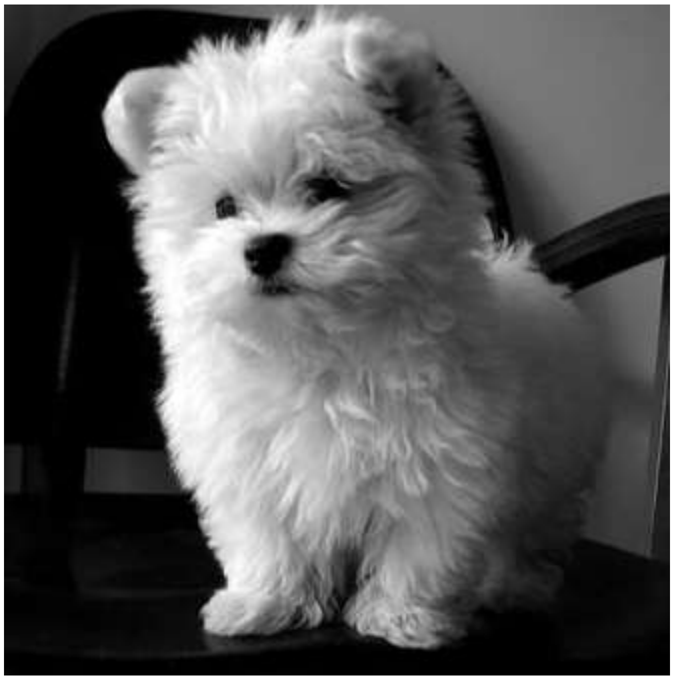


Figure 3.19-Maltese-dog

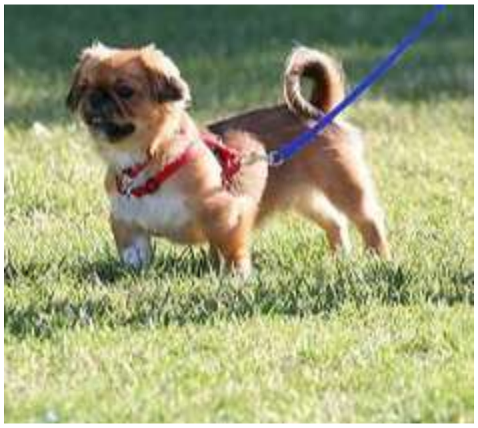


Figure 3.20-Pekinese -dog

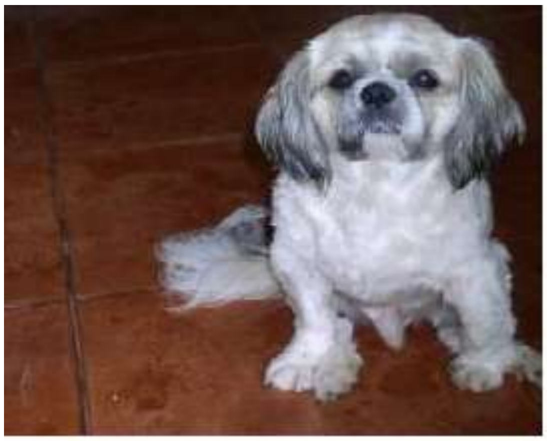


Figure 3.21-Shih-Tzu-dog

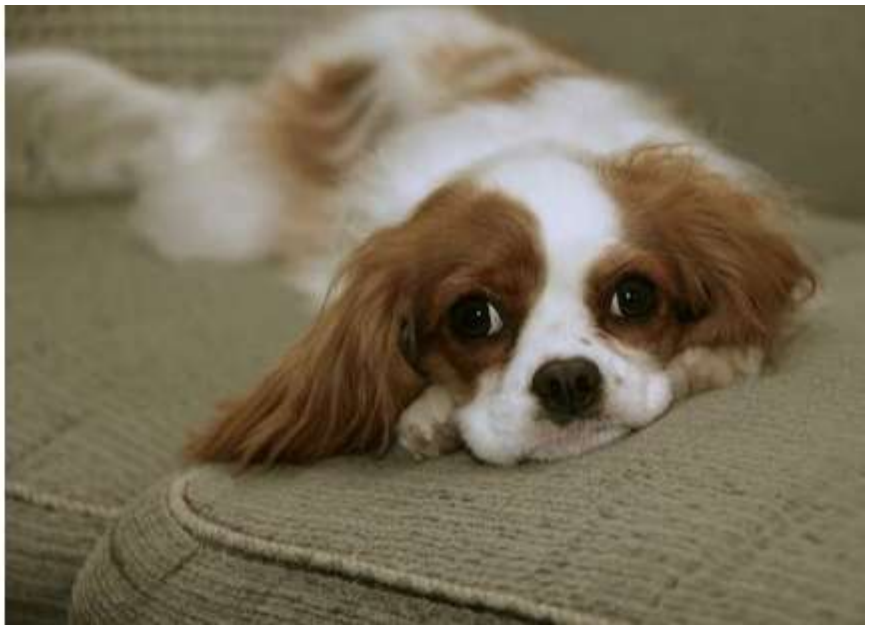


Figure 3.22–Blenheim-spaniel-dog

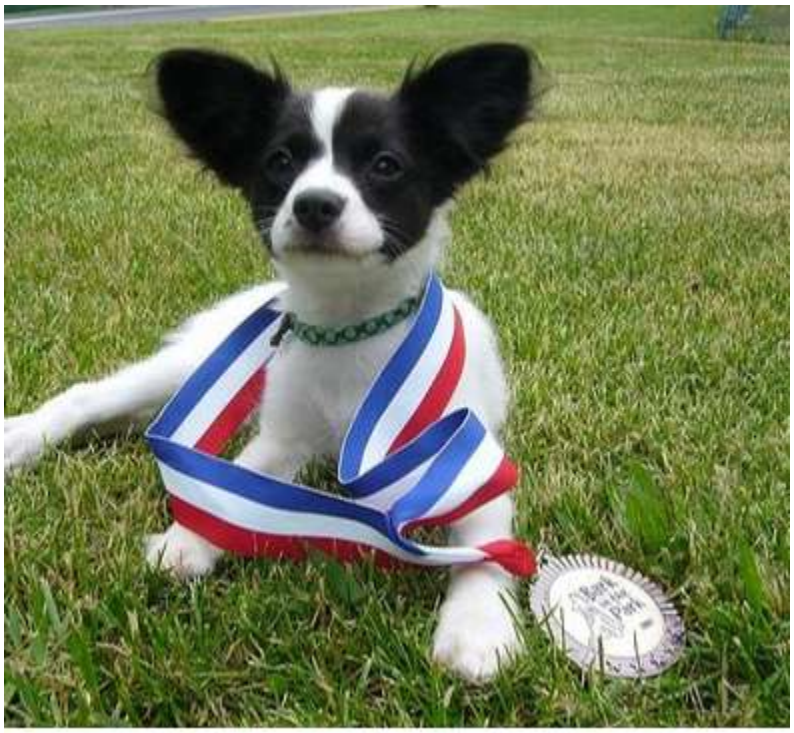


Figure 3.23-papillon-dog

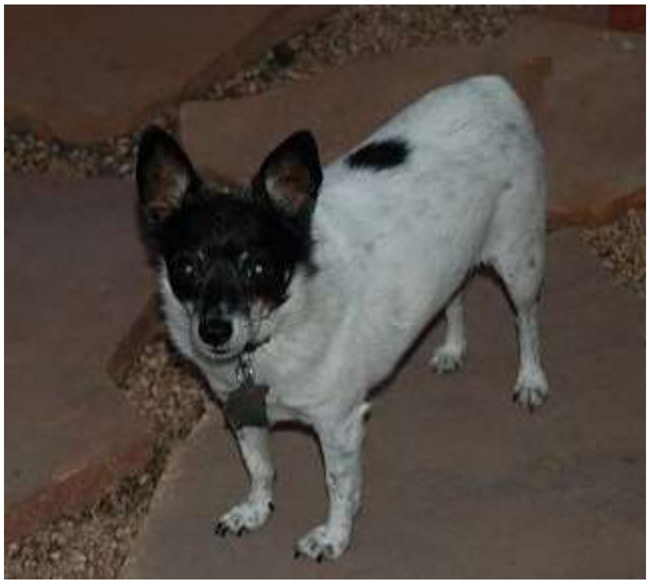
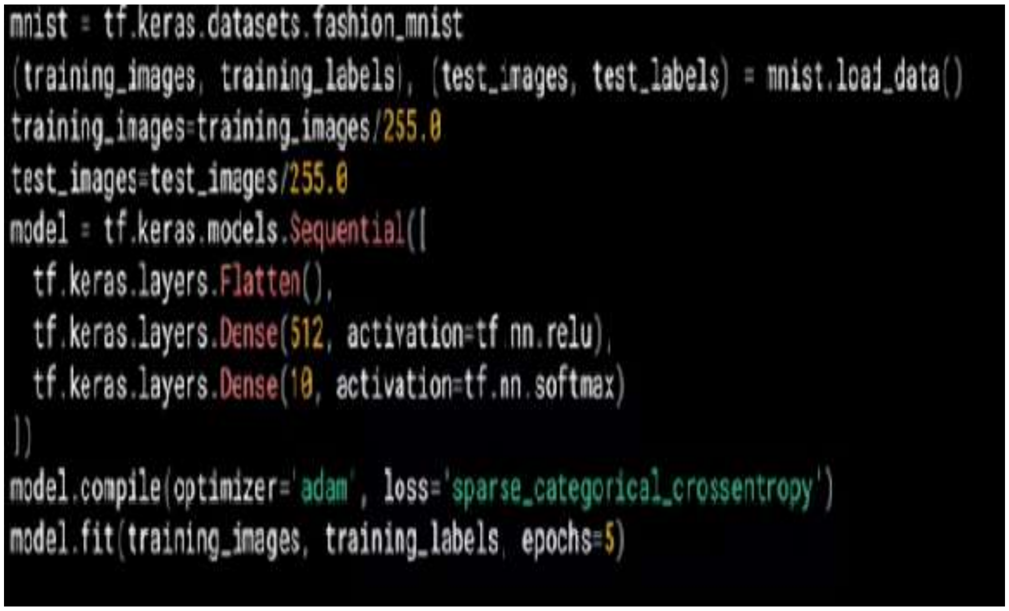


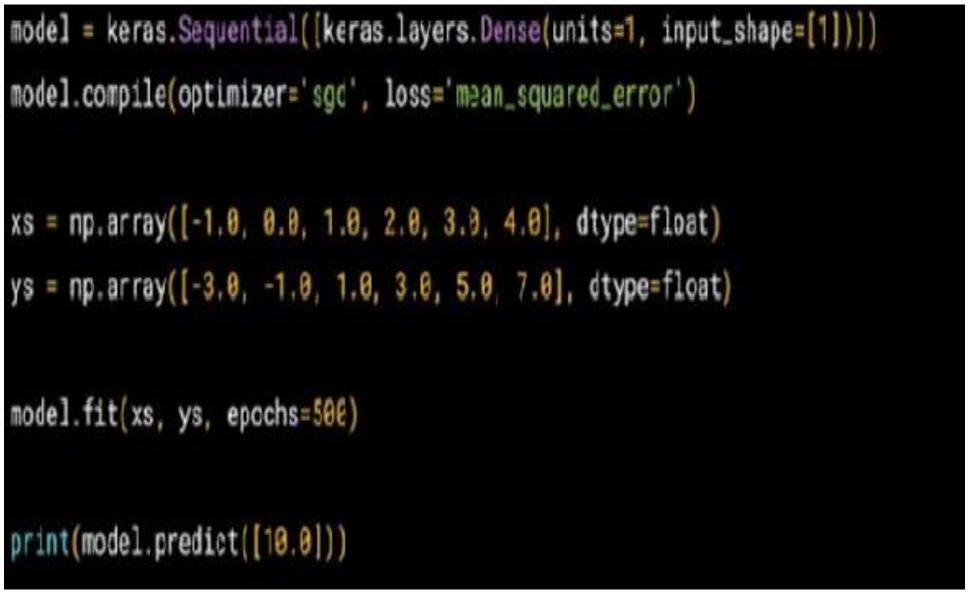
Figure 3.24--toy-terrier -dog

Keras:



```
mnist = tf.keras.datasets.fashion_mnist
(training_images, training_labels), (test_images, test_labels) = mnist.load_data()
training_images=training_images/255.0
test_images=test_images/255.0
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy')
model.fit(training_images, training_labels, epochs=5)
```

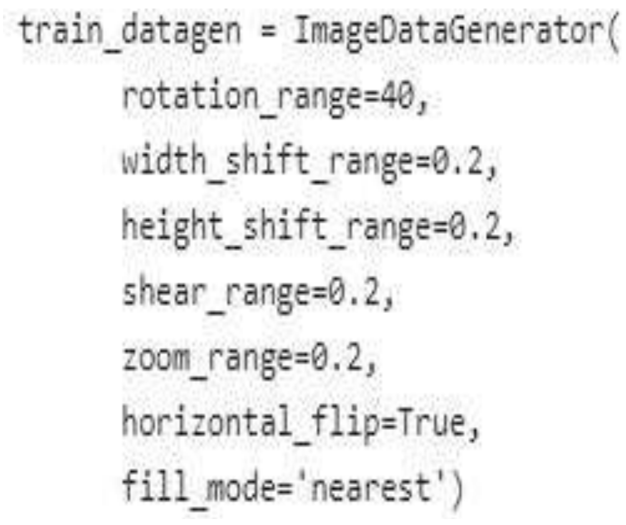
Figure 3.24–keras



```
model = keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')

xs = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
ys = np.array([-3.0, -1.0, 1.0, 3.0, 5.0, 7.0], dtype=float)

model.fit(xs, ys, epochs=500)

print(model.predict([10.0]))
```

Figure 3.25–neural network creation

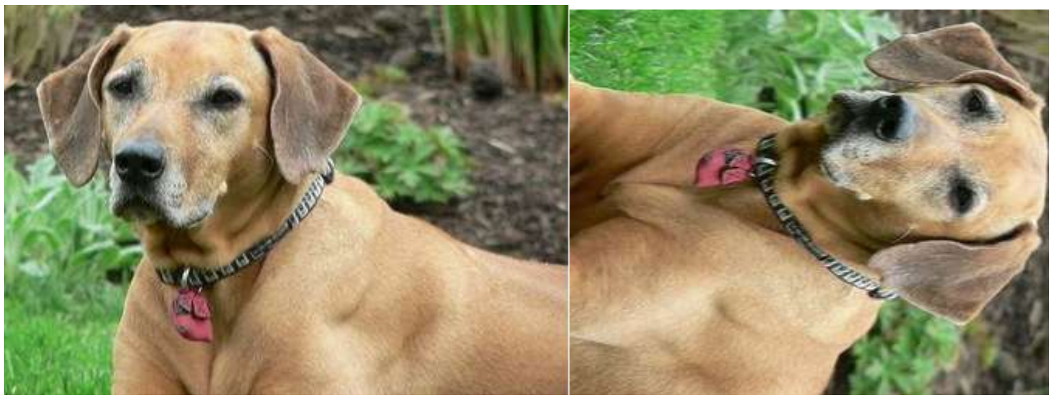
Augmentation-techniques:



```
train_datagen = ImageDataGenerator(
        rotation_range=40,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest')
```

Figure 3.26–neural network creation

Augmentation-examples:



Operations under Augmentation-techniques:

1. Rotation_range
2. Shear_range
3. Height-shift range
4. Width-shift range
5. Zoom-range
6. Horizontal_flip
   Fill_mode

Examples:

Chapter-5

CONCLUSION

Attheend,wecansaythedeep-learning-modelsarequitecapabletoevolveand dominate mankind’sintellects but there exists a condition that the data should be provided to them and it should be sufficient. Work is still being carried out by the engineers and scientists on the field of deep-learning as to thispointthis field of deep-learning consists very little knowledge. It is expected that in the future, this field have capabilities to havedeeplearningmodelsthatcanparticipate in creationofdeep-learning-modelsby themselves &deep-learningmodelsthatcantypecoding material&willchangemankind’svisionvery soon. There is a very high scope that the deep learning can be used in the field of medical-sciences with their analytical skills shown on the medical pictures with deep-convolution-neural-networks. On the bad side it can be presumed that deep-learning givestheprobabilityregarding the reasons whichwillobliterate mankind in future. Dog-breed-classifier that we are preparing is alsoasmallportionin deep-learning, which will be made betterby usage of google x-ception-model & advanced neural-network. Transfer-learning gives uspossibilitythat accuracy can be increased with combination of prebuilt-model that are trained on very large datasets for so many days, with the model we willconstructing.

REFERENCES

[1] M.D.Zeiler, R. Fergus, "Visualizing and understanding convolution neural network ", ECCV, 2014.

[2] J. Bergstra, R. Bardenet, Y. Bengio, B. Kegl, "Algorithms for hyperparametersoptimization",

Proc. Adv. Neural Inf. Process. Syst., pp. 2546-2554, 2011.

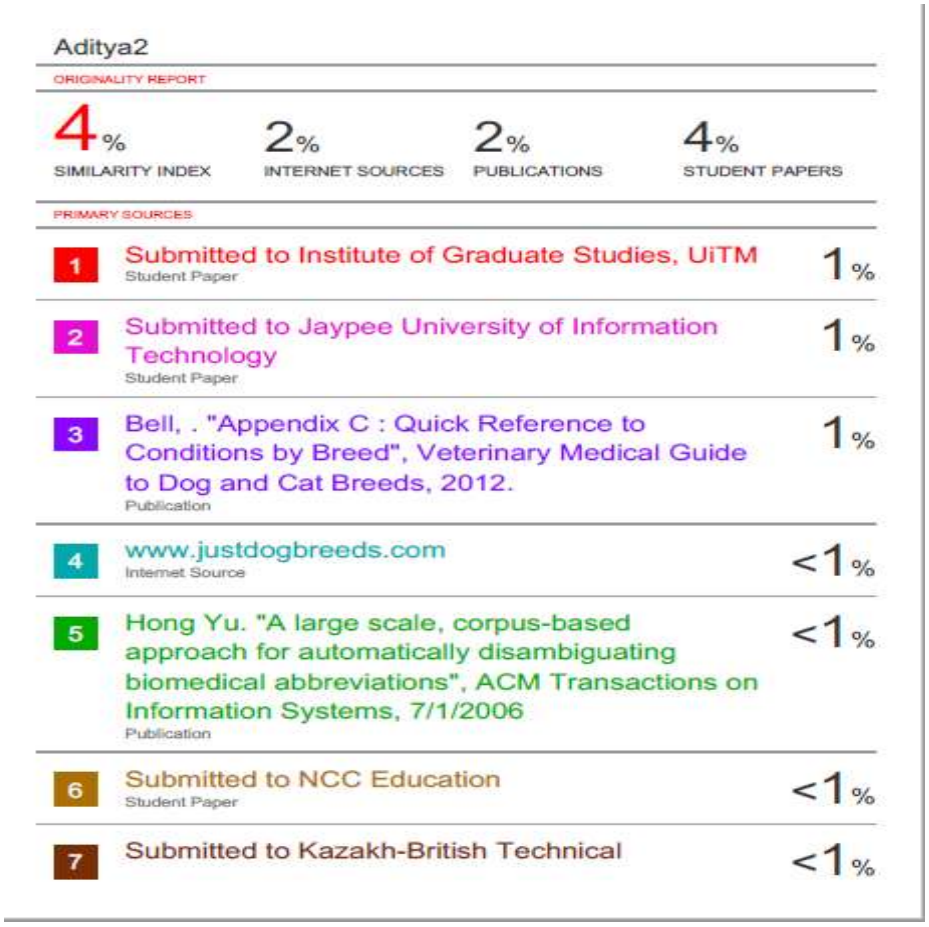
[3] S.J.PanandQ.Yang,"ASurveyonTransferLearning,"inIEEETransactionsonKnowledge and Data Engineering, vol. 22, no. 10, pp. 1345-1359, Oct. 2010.

[4] J. Liu, A. Kanazawa, D. Jacobs, and P. Belhumeur, “Dog Breed Classification Using Part Localization”, Computer Vision–ECCV 2012. Springer Berlin Heidelberg, 2012.172-185.

[5] G. Csurka, C. Dance, L. Fan, J. Willamowski, C. Bray, "Visual categorization with bags of keypoints", *Workshop on statistical learning in ECCV*, vol. 1, pp. 1-2, 2004.

[6] M. Abadi et al., "TensorFlow: A System for Large-Scale Machine Learning TensorFlow: A system for large-scale machine learning", *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI'16)*, pp. 265-284, 2016.

PLAGIARISM REPORT



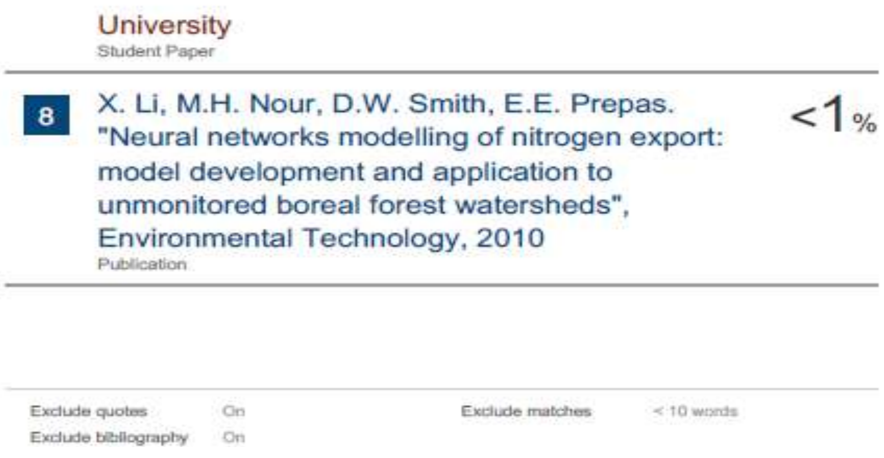
Aditya2

ORIGINALITY REPORT

4%	2%	2%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Institute of Graduate Studies, UiTM Student Paper	1%
2	Submitted to Jaypee University of Information Technology Student Paper	1%
3	Bell, . "Appendix C : Quick Reference to Conditions by Breed", Veterinary Medical Guide to Dog and Cat Breeds, 2012. Publication	1%
4	www.justdogbreeds.com Internet Source	<1%
5	Hong Yu. "A large scale, corpus-based approach for automatically disambiguating biomedical abbreviations", ACM Transactions on Information Systems, 7/1/2006 Publication	<1%
6	Submitted to NCC Education Student Paper	<1%
7	Submitted to Kazakh-British Technical	<1%



	University Student Paper	
8	X. Li, M.H. Nour, D.W. Smith, E.E. Prepas. "Neural networks modelling of nitrogen export: model development and application to unmonitored boreal forest watersheds", Environmental Technology, 2010 Publication	<1%

Exclude quotes	On	Exclude matches	< 10 words
Exclude bibliography	On		

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: 16-07-2020……………………

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | [✓ B.Tech Project] | Paper

Name: Aditya Thakur **Department:** Computer Science **EnrolmentNo** 161274

ContactNo. 7018924126 **E-mail.** thakuraditya696@gmail.com

Name oftheSupervisor: Dr. Suman Saha

Title of the Thesis/Dissertation/Project Report/Paper (In Capitalletters):

DOG BREED CLASSIFIER USING CNN AND IMAGE PROCESSING

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages=57
- Total No. of Preliminary pages=5
- Total No. of pages accommodate bibliography/references=1

[illegible]

(Signature of Student)

FOR DEPARTMENT USE

Wehavecheckedthethesis/reportaspernormsandfound**SimilarityIndex**at .... 4(%). Therefore,we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

[illegible]

(SignatureofGuide/Supervisor) **Signature ofHOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	**Excluded**	**Similarity Index (%)**	**Generated Plagiarism Report Details** (Title, Abstract & Chapters)	
	• AllPreliminary Pages • Bibliography/Images/Quotes • 14 WordsString		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name&Signature **Librarian**

……

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com