“Dog Breed Classifier Using Deep Learning”

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

In

Computer Science and Engineering and Information Technology

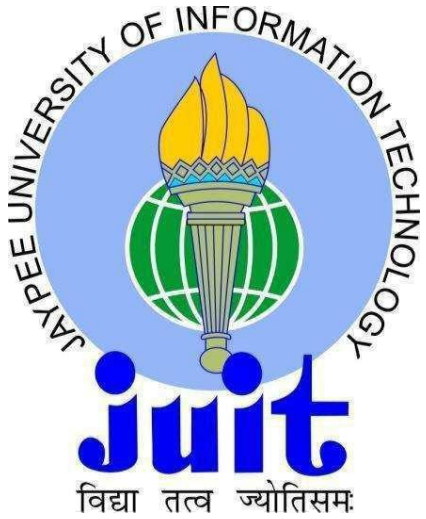
By

Agrim Dogra (151478)
Harsh Massand(151405)

Under the supervision of

(Dr. Amit Kumar Jhakar)

To



Department of Computer Science & Engineering and Information Technology

Jaypee 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 Using Deep Learning"** 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 2018 to May 2019 under the supervision of **Dr. Amit Kumar Jhakar** (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.

Agrim Dogra(151478)
Harsh Massand(151405)

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

Dr. Amit Kumar Jhakar

Assistant Professor, Senior Grade
Computer Science and Engineering and Information Technology
Dated:

ACKNOWLEDGEMENT

We owe our profound gratitude to our project supervisor **Dr. Amit Kumar Jhakar**, who took keen interest and guided us all along in my project work titled - **“Dog Breed Classifier Using Deep Learning** till the completion of our project by providing all the necessary information for developing the project. The project development helped us in research and we got to know a lot of new things in our domain. We are really thankful to him.

TABLE OF CONTENTS

ABSTRACT

Pattern recognition(PR) is realized as a human recognition process which can be completed by computer technology. We should first enter useful information of identifying the object into the computer. For this reason, we must abstract the recognition object and establish its mathematical model to describe it and replace the recognition object for what the machine can process [1] . The description of this object is the pattern. Simply speaking, the pattern recognition is to identify the category to which the object belongs, such as the face in face recognition. Our project is based on PR which is to identify the dog's breed. In our project, based on 10,000+ images of 120 breeds of dogs, we use 4 methods to do the identification. Each method has a different training model. The four models are ResNet18, VGG16, DenseNet161, and Alex Net. Based on our models, we also make some improvements on the optimization methods to increase our identification accuracy. After our comparisons, we find that the Dense Net model is the best, and we take it as our prime model. Our best accuracy can be up to 85.14%.

This project uses computer vision and machine learning techniques to predict dog breeds from images. First, we identify dog facial key points for each image using a convolutional neural network. These key points are then used to extract features via SIFT descriptors and color histograms. We then compare a variety of classification algorithms, which use these features to predict the breed of the dog shown in the image. Our best classifier is an SVM with a linear kernel and it predicts the correct dog breed on its first guess 52% of the time; 90% of the time the correct dog breed is in the top 10 predictions.

CHAPTER 1

DOG BREED CLASSIFIER USING CNN AND IMAGE PROCESSING

INTRODUCTION:

IMAGE PROCESSING:

Image Processing is a very powerful technique used today to convert an image into a form which is either digital or analog so that it can be used to extract some important and useful data. This process takes a raw image as an input and processes it and gives an improved modified image or characteristics associated with that image as an output. Image processing when used in ML algorithms such as CNN can be used to get very interesting results such as image recognition or creating a model to predict some feature from image.
Mainly these three processes are involved in image processing

- Fetching the required image by using any available tool.
- The fetched image is then analyzed and some necessary manipulation is done on it to find some significant patterns in it which are not visible to a naked human eye.
- The last stage is the output stage where the output is either an image or a report based.

What is an image?

An image is a digital representation of a picture i.e electronic form. When an image is stored in rastere form it is called bitmap.
JPEG, PNG, GIF89a, GIF, SVG,

Types of Images

1. Binary (0/1) Image: This is a digital image in which every pixel is capable of taking only two colors i.e. black and white. The black color is denoted by 1 and while by 0, that's why it is also called a 0/1 image.

2. Gray scale images: This is a digital image in which each pixel instead of containing color, contains the intensity information of light. This image is also called black and white image or monochrome image.

3. Colored images: This is a digital image in which every pixel contains the intensity of RGB (Red, Green, Blue) color. RGB is chosen because all other colors can be formed by mixing the intensity of these three colors. A colored image every pixel stores 3 values.

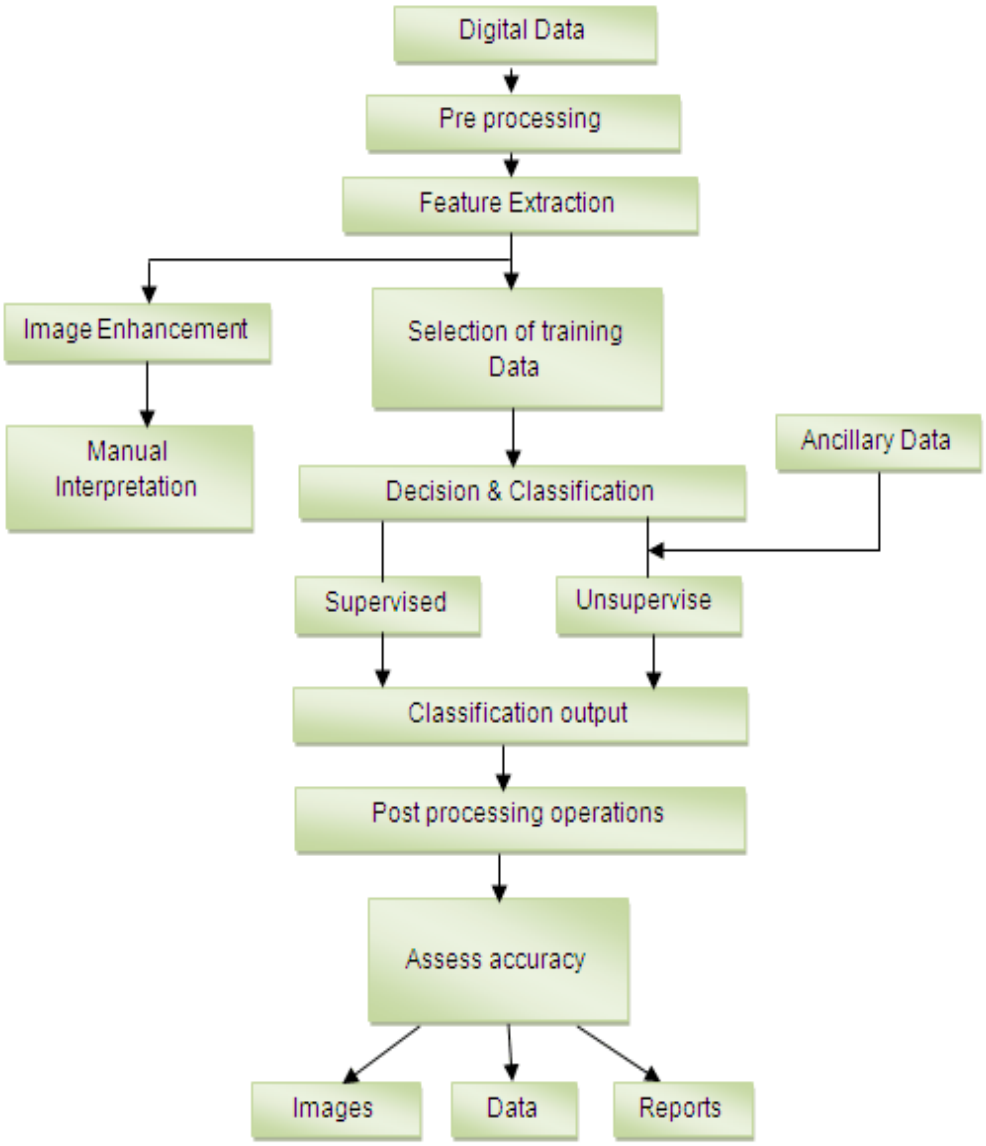


Fig. 1: Flow Chart Showing Diffrent Phases in Digital Image Processing

Types of Digital Images Processing Techniques:

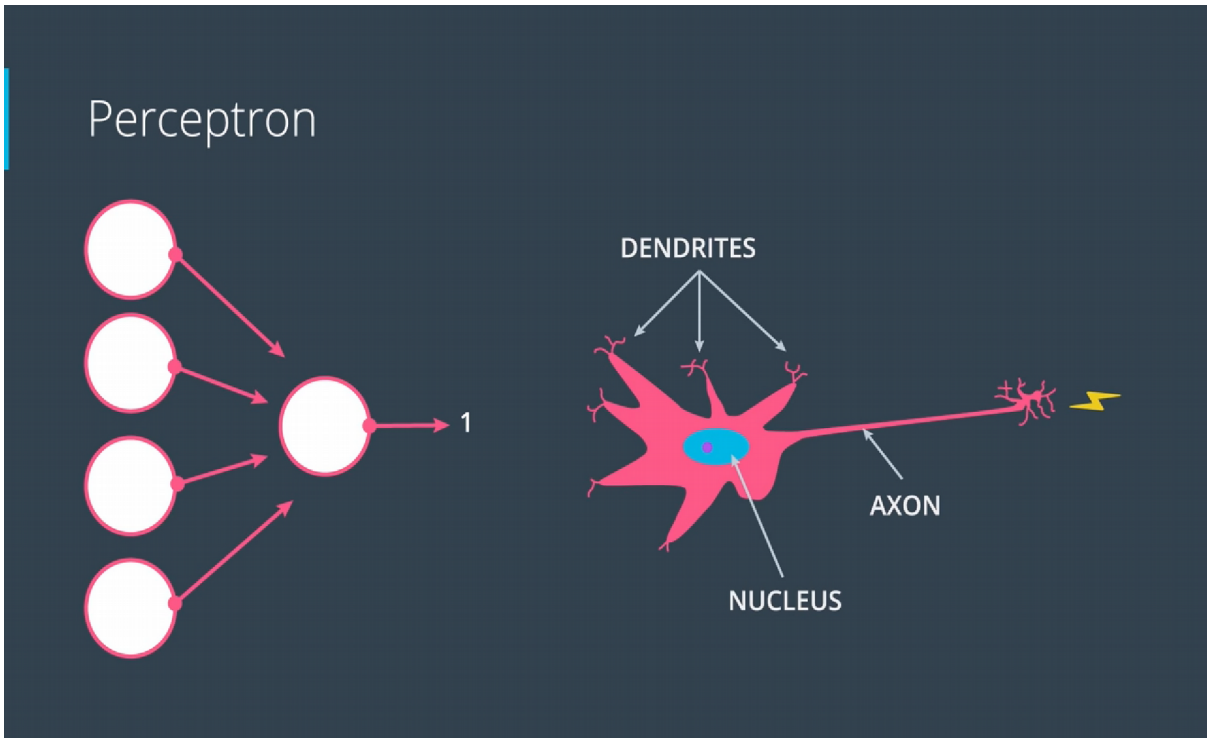
- Editing of the image
- Anisotropic Diffusion
- Linear Filtering of image
- Neural Networks (CNN)
- Image analysis
- Image segmentation
- Image data compression.
- Component Analysis

Convolutional Neural Networks and images(CNN):

Convolution Neural Network is a word taken from neural network present in human body because of their analogy. The working of Convolution Neural Network is exactly same as the working of human neural network.

The human Neural Network is a collection of collection of neurons, dendrites that generates the output signal which is sent to different parts of body. Neural Network is present in brain which performs various computations. The dendrites present at the ends of nerve cells act as sensors and receive the signals, it then converts it to electrical signal which is then passes to the nucleus of the cell through axon where the decision is taken and message is again sent back via similar process.

CNN also work similar to human neural network. This network consists of neurons, weights and biases. This network receives the input and uses the biases and weights to calculate the weighted sum of the input and passes this sum through an activation function and finally comes up with a prediction or an output.



ARTIFICIAL CONVOLUTION NEURAL NETWORKS VS REAL NEURAL NETWORKS.

Activation Function in CNN:

As we know neural network consists of a large number of neurons, the activation function decides whether the neuron need to be activated or not and it is done by calculating the sum of the weights and bias together. This function is mainly required to bring non linearity to our model. Without an activation function our model will just be a linear model and will not be able to train and predict accurately for data which forms a complex boundary. It will only work as linear model which will be of not much use to us.

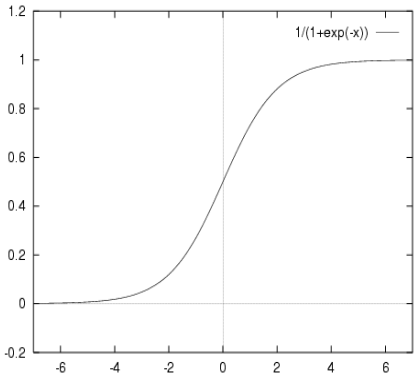
Different types of activation functions:

1. **Sigmoid Function**: The mathematical formula it uses is **f**(a) = 1 / 1 + expp(-a). The values of the function lies between 0 and 1.

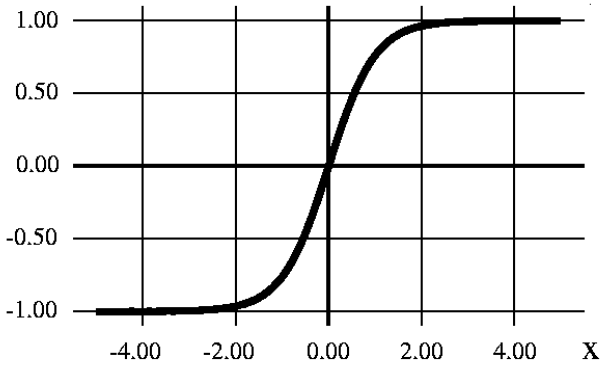
2. **Tanh (Hyperbolic Function):** The mathematical formula used for this function is **f(a) = 1- expp(-2a) / 1 + expp(-2a).** The value of this function is between -1 to 1.

3. **ReLu- Rectified Linear units :** The mathematical formula used for this function is R(a) = maximum (0, a) i.e if a < 0, R(a) = 0 and if a >= 0, R(a) = a.

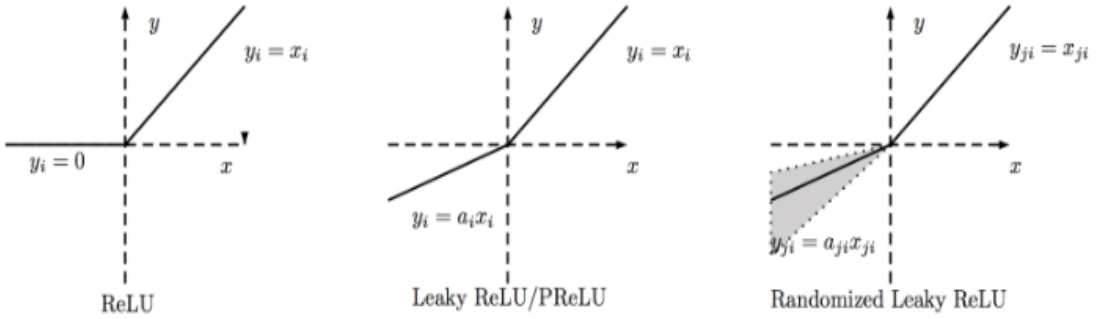
Sigmoid function:



Tanh function:



ReLu function:



1.1) PROBLEM STATEMENT

Build a model using deep learning algorithms and image processing to predict the breed of 133 breeds of dogs by taking their image as input. Use transfer learning to build a model that recognizes up to 133 different dog breeds.

1.2) OBJECTIVE

The main objective of this project is:

- To use deep learning algorithms and predict the breed of the dog.
- To get best possible testing data accuracy.
- To train the model on 133 breeds of dogs on over 1 lakh images.
- To use CNN to recognize the patterns in the train data and make changes to the model.
- To predict the breed irrespective of the child or adult member of that breed.
- Use Python, Keras and image processing tools to filter our data and cleaning of our input data.

1.3) METHODOLOGY

So, initially, the data is converted into blacks and white images. The data is broken into three sets which are basically training set, testing set, and validation set. The training dataset is used for calculating the pattern in images and adjusting the images. This training dataset is passed through a deep neural network. Feedforward is done and backpropagation is done to calculate the error and adjustment of weights. Then we plug in the image to know the breed of a dog. This breed is then revealed by feedforward. The probabilities of dogs breed are predicted and thus output is calculated.

CHAPTER 2

LITERATURE REVIEW

What is Machine Learning?

Machine learning is a field of Artificial Intelligence that uses data analysis and machine learning algorithms to create a model and trains itself according to the training data and learns some parameters according to it and uses them to predict some output according to those parameters. The data provided is divided to training and testing data and the accuracy of the algorithm is calculated on the basis of test score it obtains.

Machine Learning is mainly divided into 3 categories:

- **Supervised Machine Learning:**

In this type of algorithm, the training data is used to analyze the data given ,learns some parameters and creates a function which takes the data entries as input and passes them through that function to calculate the output. In input data the features and output columns are clearly separated.
Almost 70% of the machine learning is Supervised Machine Learning.
Algorithms which use supervised machine learning are:

- Linear Regression
- Logistic Regression
- Decision Trees
- Naive Bays
- KNN
- K-Means
- Random Forest

- **Unsupervised Machine Learning:**

This kind of algorithm is used when the input data is not classified as well as not labeled. These algorithms identify the patterns in the provided data and draw inferences from the data sets to identify the data which is unstructured and unlabeled. However, these algorithms are not capable of predicting very accurate results but still it provides quite efficient results base on the kind of unstructured data provided. These algorithms are more complex and difficult to understand than the supervised algorithms.
Algorithms which use unsupervised learning are:

- Hierarchical clustering
- K-means clustering

- Expectation minimization

- **Semi-supervised Machine Learning Algorithms:**

These are the algorithms which use the combination of both supervised and unsupervised learning techniques. They use both labeled as well as unlabeled data to compute the patters available and create function to predict the output for the rest data. For its working it can use a bit of labeled data and most part of unlabeled data or its vice versa. Classification, Regression and recognition are some of the methods it uses.

Algorithms which use unsupervised learning are:

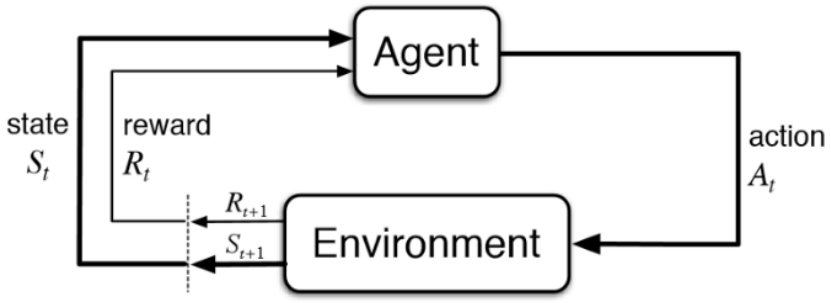
- Face recognition techniques
- Voice recognition techniques

- **Reinforcement Learning Algorithm:**

It is a kind of learning where the model interacts with the real world environment for training itself and discover patterns. Here the model learns from its own outputs. Trial and error method is applied where a model performs a lot of trials on the data and tries to predict the output based on previous learning , and for every correct prediction the model is given some positive reward and for a wrong prediction it is given some negative points. Hence by making more and more trials the accuracy of the model increases significantly. Reward and feedback are required by the agent to learn new features and increase accuracy of the system.

Algorithms which use unsupervised learning are:

- Q Learning Algorithm
- State-Action-Reward-State-Action (SARSA)
- Deep Q Network (DQN)
- Deep Deterministic Policy Gradient (DDPG)



Reinforcement Learning Illustration (https://i.stack.imgur.com/eoeSq.png)

How does a machine look at an image?

The brain possessed by human beings is very powerful. It capable of doing multiple, very different and very difficult things together. For an example we can say our eyes see a lot of things every millisecond and the brain processes the image to figure out what that image is within no time and we never pay attention to it. But machines are not like human brains. We have to make them learn the images so that they come to know about it, and in doing so the first step is to use image processing to convert an image to a digital image so that our machine is able to read that.

An image in straightforward words is a collection of pixels which are very small in size and the intensity of color in a unique and a spatial order. This pixel formation is different for different images, changing which the whole image is changed.

To understand this let us take an example where we want to look an image with 5 written in it, The read this the image will be changed to a lattice of pixels where each pixel will contain a value ranging from 1 to 256 where 1 represents the whitest shade of color and 256 represent the darkest shade of color.

The neural network then identifies the patterns in that matrix to remember that image so that it can later itself recognize that image. This neural network will take the value of the pixels as a feature on which the ML algorithm will work to predict the output. It is almost impossible for a human eye to recognize the pattern which this network learns to identify that image. We can provide different weights and bias to the model to alter our results.

a) Representation of image in pixels.

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

Defining a Convolutional Neural Network

A convolutional neural network is a spatial neural network which is mostly used to recognize patterns in a image an input. CNN basically is defined by the following 3 steps:

1. The Convolutional Layer of CNN
2. The Pooling Layer of CNN
3. The output layer of CNN

These layers are explained as:

The Convolution Layer of CNN:

As the image we have is stored in the form of a large matrix where each entry contains pixel intensity ranging from 0 to 256. So to calculate or recognize the features we create a 2D weight matrix who's each entry contains a weight. Mostly we take it to be a square matrix of any size like 3*3 or4*4 etc. now this weight window is moved all over the pixel matrix and the overlapping pixel value is multiplied with the corresponding weight value, and they are summed up to get a we weight of that window. Now this pixel value is moved to right or down so that the whole image is covered and the calculated numbers for each window are added at the end. These processed values help to recognize patters of the image. The weight matrix behaves like a filter which moves over the whole image an extract the important and useful features of that 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

Original Image

Convoluted image

Multiple filters are used for analyzing and recognizing the depth effect in the input. A single layer filter is not able to get the output with depth as a dimension. Multiple layers will result in a 3D weighted matrix and will take care of depth as a dimension. The output from each filter is summed and compiled together to finally form a convoluted image.

The Pooling Layer

Pooling is technique which is mainly used when the size of the image is to lagre. If the size of the image is too large we need to reduce the number of learning parameters present in that image recognition. And while doing so it is required to insert pooling layers in between them. It is mainly done so that the size of image does not become too large and while doing so the image depth dimensions are not altered hence there is no point that the depth of the image will be affected in any way. Max pooling is one of the most used pooling which is generally applied. The output image looks almost very much similar to the original image and only the dimensions of image are altered

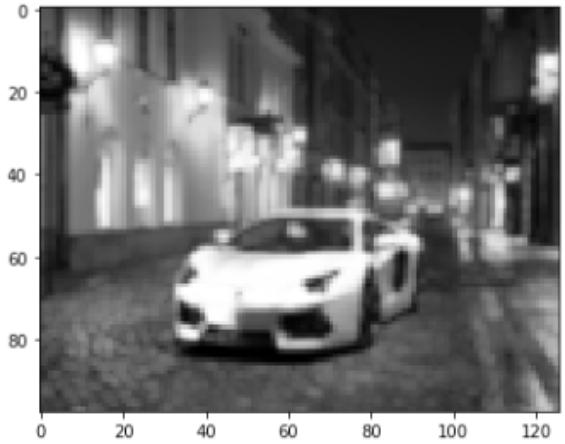
Other types of pooling are average pooling or L2 norm pooling. The size of the output image is controlled by 3 parameters:

- **The number of filters:**
  The number of filters in a network determine the width of the image. Thus the output volume of the image will depend on the number of filters used. Also while pooling the width parameters in filter are not changed at all.

- **Stride:**
  The value of stride decides that by how much amount does the filter shifts to right or down in the pixel matrix. If its value is set to 1 it can move up, down or right by only one-pixel value hence the size of output image will be large. So we use a higher value of stride so that our output size gets reduced.

- **Zero Padding:**
  This feature helps us to maintain and retain the size of our original input image. If this value is set to 0 our original input image will remain as it is but if it is set to a higher value, we can lose our original image.

Convoluted image



MAX Pooling

The Output Layer:

After passing through the convolutional layer and pooling layer we get a compressed image with a lot of features being extracted from it. But after applying the multiple layers to the image we finally have to extract the output class so we have to apply the full connected layers so that we are able to come up with the required output.

For doing so we need to create a 3D activation map to predict whether an image belongs to a particular class or not. The error can also be predicted in the output class by using the loss functions for eg. Cross entropy can be used.

The size of the output image can be calculated using formula: **(<u>(W-F+2P)/S) +1</u>**

Here : W is input volume size.
F is filter size.
P is number of padding applied.
S is number of strides used.

CHAPTER 3

SYSTEM DEVELOPMENT

System development involves a process of designing a software prototype, testing it, converting it to complete model and again applying various testing algorithms to it and finally create the whole software.

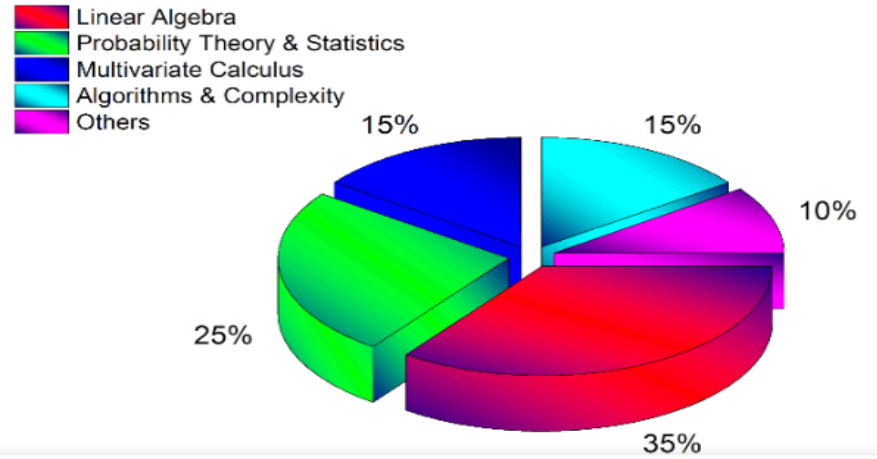
Literature Title	Literature Review	Improvement in Our Models
Dog Breed Classification Using Part Localization (Liu, 2012) **Going Deeper with Convolutions** (Christian, 2014)	The performance of fine-grained classification can be improved by using part localization since dog breeds are similar in common 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.

Mathematical Methods:

Mathematics plays a very important role in creating a model using CNN(Deep ML).
We need mathematical methods to:

- Select the correct algorithm based on the mathematical data available.

- Choosing correct features and parameters for the model so that the predictions are perfect.
- To check the model for over fitting and under fitting of the data.
- Expecting the uncertainty of the model.



Mathematics in ML algorithms

Functions used in mathematical models to calculate errors:

Various functions used are:

- **Mean Square Error Function:**
  It is the mean of all the squared errors obtained by an algorithm in logistic regression. The error is the difference between the actual value and the calculated value, which is calculated between various data points.

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

  Here : wt is the weight associated,
  x(i) is predicted value,
  y(i) is the actual value

- **Euclidean distance metric:**
  This is an another very important performance metric used in ML algorithms to calculate the distance between to outcomes. It takes the feature set as input and applies the formula

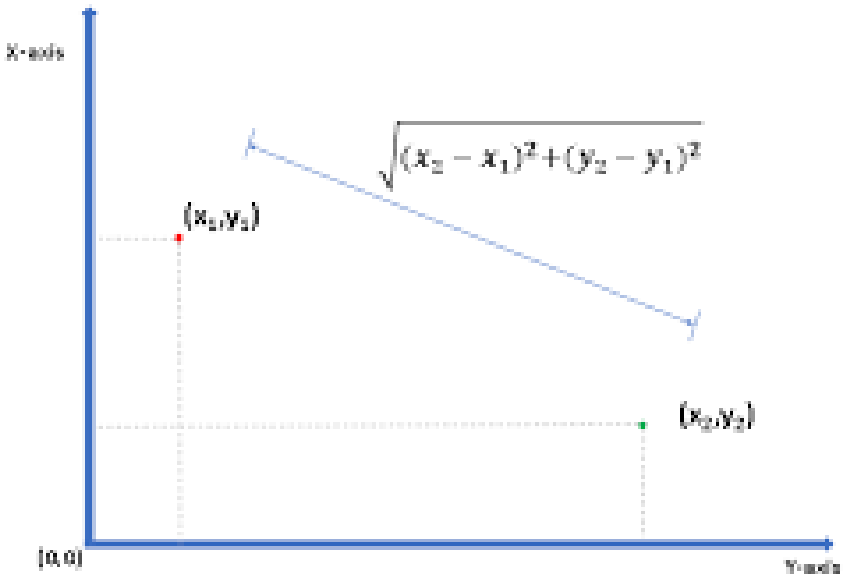
to the values of the parameters of the point to calculate the distance. Euclidean space can also become a capable metric in space. This metric is also known as Pythagorean metric.

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

Here: (x,y) are values of two features of point 1.
(a,b) are values of tw0 features of point 2.

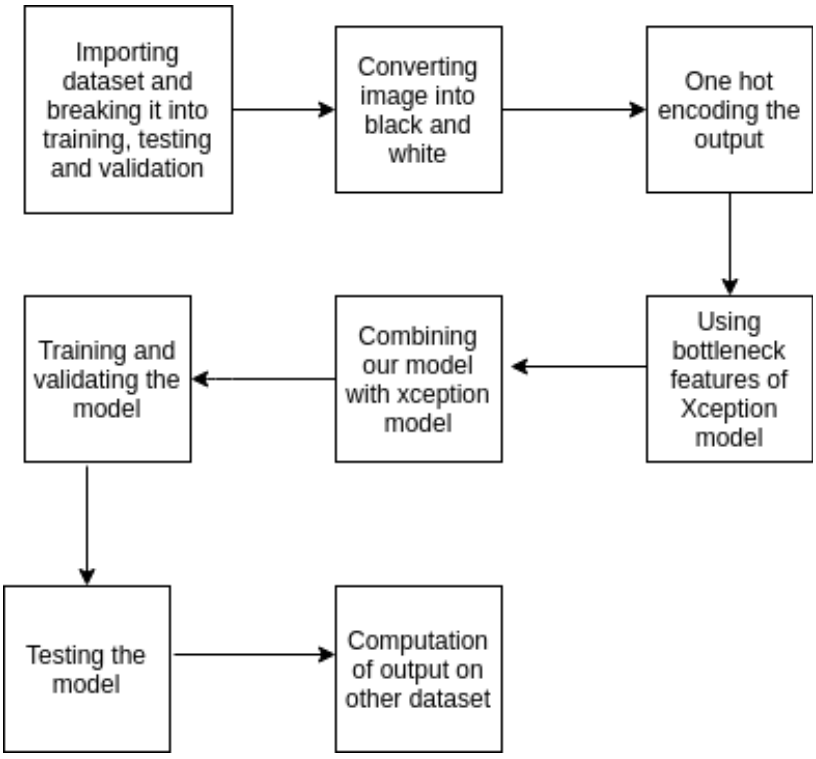
- **Manhattan Distance:**

Manhattan distance is also a distance metric used in machine learning algorithms. It differs by Euclidian distance very slightly. This metric is used we very quickly Want to find the distance between them.



Formula to find the distance.

Flow Chart of our deep learning model:



Steps involved in flowchart:

1. The dataset is imported from the web and is broken into three parts i.e training, validation and testing.
2. The dog images are converted into black and white because we don't need any color in our case to find out the pattern.
3. We have 133 breeds so we have to use one hot encoding and set the position of the data. The breed corresponding to the input is set to one and others are set to zero.
4. We have imported the bottleneck features of xception model for dog breed classification.
5. We have build our model and added in front of the pre trained xception model.

6. Now based on the training dataset and validation dataset the model is trained and validated. The weights are adjusted based on back propagation.
7. Now the dataset is tested on testing dataset and accuracy is computed.
8. We plug in any image of the dog and check for the accuracy.

Architecture of model

```
### TODO: Define your architecture.
from keras.layers import Conv2D, MaxPooling2D, GlobalAveragePooling2D
from keras.layers import Dropout, Flatten, Dense, Activation
from keras.models import Sequential

Xception_model = Sequential()
Xception_model.add(GlobalAveragePooling2D(input_shape=train_Xception.shape[1:]))
Xception_model.add(Dense(133, activation='softmax'))

Xception_model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
global_average_pooling2d_1 ( (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 133)               272517
=================================================================
Total params: 272,517
Trainable params: 272,517
Non-trainable params: 0
_________________________________________________________________
```

Figure 3.1 Architecture of model

So this is architecture of our model in which we have added Xception bottleneck features initially and added our dense layer model with the soft max Activation function. Softmax

activation will convert the data into some probabilities. The probabilities of each breed is predicted and based on the maximum value the output is thus calculated.

This architecture is bit complex because it contains bottleneck features of exception model and this model has a great power in the determination of different types of objects.

So the output shape of the xception model is 2048 which will be the input to our model. The output from the dense layer is 133 which is due to 133 breeds present in our output.

Maximum probable breed will be picked up from our model and result will be displayed based on this.

ALGORITHM

Basic Algorithm structure

Dog Breed Classifier consists of advanced convolution neural network. Algorithm design of convolution network uses an insight knowledge of machine learning as well as neural networks algorithms. Deep convolution neural networks use advanced filters to detect the patterns that are present within the image. We will make a filter, that will be randomized using various distributions like Gaussian, normal, uniform distribution. So we place a randomized filter on an image and figure out the pattern. Multiple filters ,max-pooling layers, fully connected pooling layers, and dense layers are used to detect the pattern in an image. Activation functions and dropout also played a role in the development of our deep convolution network system that is quite robust. We have constructed a model that contains several layered architectures and it is combined with the and passed our dataset through the model to adjust the weights by attaching some of the data in front of the xception model made by Google.

Xception Model

Xception model is built by Google that gains superhuman capabilities in detecting object identification task. Xception model uses depth wise convolution that is much more cost effective than simple convolution which is a much more costly operation than simple convolution. Simple Convolution makes the multiplication with the number of elements present within that kernel. As multiplication is a costly task we just can’t afford this to work on a simple central processing unit. For big matrix multiplication, we have to depend on CUDA that is developed by Nvidia. Xception model object detection sometimes super pass the human capabilities of detection. In this project, we merged our own created model with the so created model xception. Then we trained our model and checked the result and the computation was great. CUDA has a very great ability to run the training of model process on parallel graphics processing units by using Nvidia graphics environment Nvidia graphics environment. Xception model is a model that took many days to get trained and gained very high capabilities to identify the object. We will simply attach our model in front of xception model to predict the dog breeds with high accuracy.

As duplication is an expensive assignment we can't bear the cost of this to chip away at a straightforward focal preparing unit. For huge grid duplication, we need to rely upon CUDA that is created by Nvidia. Xception model article recognition some of the time super pass the human abilities of identification. In this task, we blended our own made model with the so made model xception. At that point, we prepared our model and checked the outcome and the calculation was incredible. CUDA has an exceptionally incredible capacity to run the preparation of model procedure on parallel illustrations handling units by utilizing Nvidia designs condition Nvidia illustrations condition. Xception model is a model that took numerous days to get prepared and increased exceptionally high capacities to recognize the article. We will basically join our model before xception model to foresee the puppy breeds with high exactness.

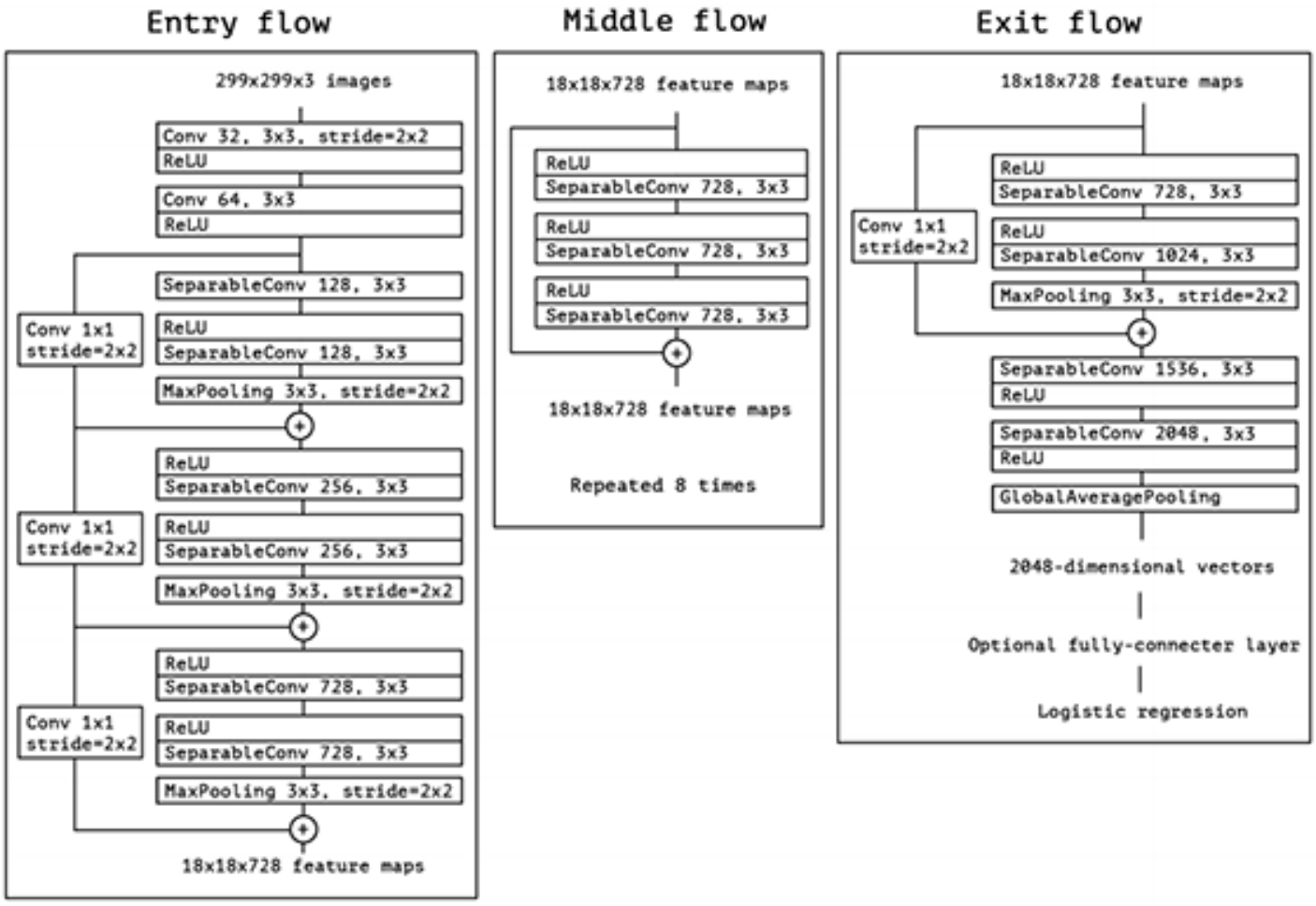


Figure 3.2 A Network Connection

Basic elements for algorithm

Filter is basically used as an agent to detect the patterns within the images. Its main task is to do convolution by continuous movement on the image. There are several types of filters like horizontal filters, vertical filters, inclined filters. Horizontal filter detects the pattern in the

horizontal fashion and similarly, other filters detect patterns based on configuration. In our algorithm, we will use randomized filters that are picked from random probability distribution functions. A filter is basically a matrix with different values of weights. So we will put a filter on an image and then multiply with the corresponding pixels that come under the filter. The value thus computed based on the filter will be passed through max pooling or global pooling layer that is a tool used to increase the depth of an image. As this process of convolution layer and pooling layer goes on we will have multiple filters and based on those filters an image can detect the pattern within patterns and inside those patterns more pattern.

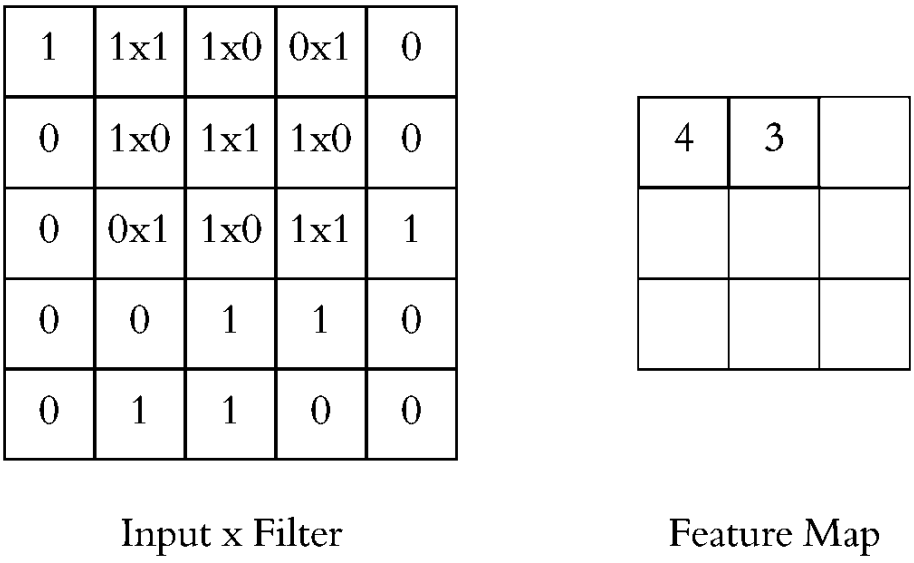


Figure 4.2 A 3*3 filter

Activation Function is basically used to scale the output values or sometimes used to convert the high values to probability so that we can detect the most probable output. In our case, the most probable dog breed identified by the classifier will be the dog breed corresponding to that input dataset pixels. In our case, we have mostly used a relu activation function that converts

negative values to zero. Activation functions can also be used to remove non-necessary values which can be harmful to our classifier. In convolution neural network mostly used activation functions are relu and sigmoid but there are many different activation functions like softmax, tan hyperbolic, exponential, soft plus etc. In our model activation functions have played a very major role in the classification of dog and humans. Activation function can be of a great significane if we talk about the necessary values by just scaling or removing the non-necessary values. This can lead the exponential increase in accuracy of model in case of training data and testing data. Along, with training and testing data validation data will decrease the overfitting of model.

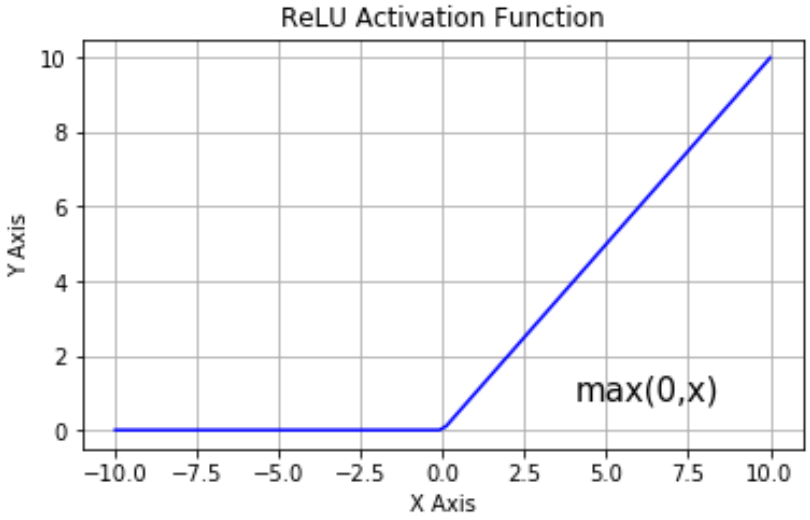


Figure 3.3 Relu Activation Function

Convolution Layers plays a most important role in the classification of any type of images dataset. Convolution layers preserve the relationship between the neighboring pixels which is not possible in the case of dense layers neural networks that are mostly used in sentimental analysis and where there is no image data. Convolution layers are a major part for dog breed classifier we have used many layers of convolution to detect the much more complex patterns that are not at all possible for humans to detect from their eyes. Convolution layer consists of many inputs that

are necessary for the initialization of layer. Convolution layer consists of filter that is used to initialize number of filters, it contains strides meaning ho much the filter has to move after it has completed the work with one set of pixels, it consists of activation function which in our case is mostly relu activation function and padding is set to same so that it does not append any pixels if the convolution filter goes out of an image. Our convolution layer also consists of one more feature kernel_initializer that is used to initialize an n*n filter. The filter will be initialized from a random probability distribution to give out the best result as possible. Without random probability distribution, our hyper parameters may result in very little accuracy of a model. It can be a challenging factor for our dog breed classifier model.

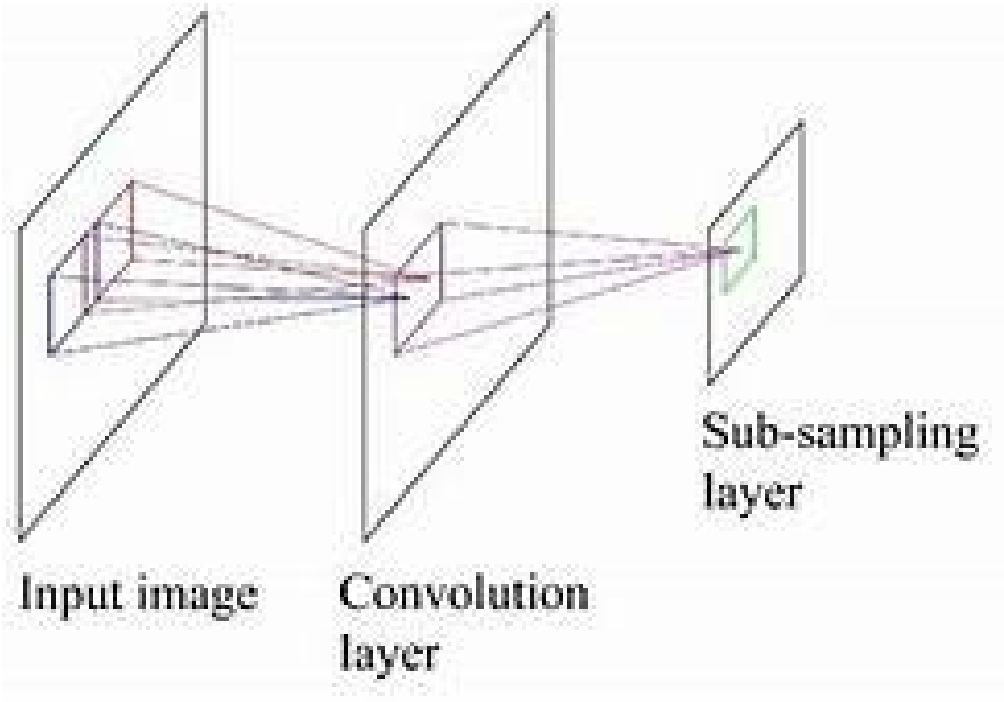


Figure 3.4 Convolution layer

Pooling Layers are used as an intermediate layer between the convolution layer to decrease the height and width of an image that is passed through convolution layer. The main purpose is to make the image as deep as possible by using more and more filters on an image and using the pooling layers to reduce width and height of an passed image. We have several pooling layers but the major pooling layer which our used in our deep learning model are global max pooling layer and global average pooling layer. The output from convolution layer will contain many different matrix based on number of filters. Our task is to apply max Pooling filter on an image with a particular size and that will extract the max from each filter thus reducing the size based on an image. Global average pooling layer will take an average of each output extracted based on different filters and reduce the dimensionality. In our model we will use the global average pooling layer at the end of our neural network before the dense layer with sigmoid activation comes into the picture. Thus we will have many filters and to reduce the dimensionality we have to work with pooling layers that will not bring a significant amount of change in our predictions of the model.

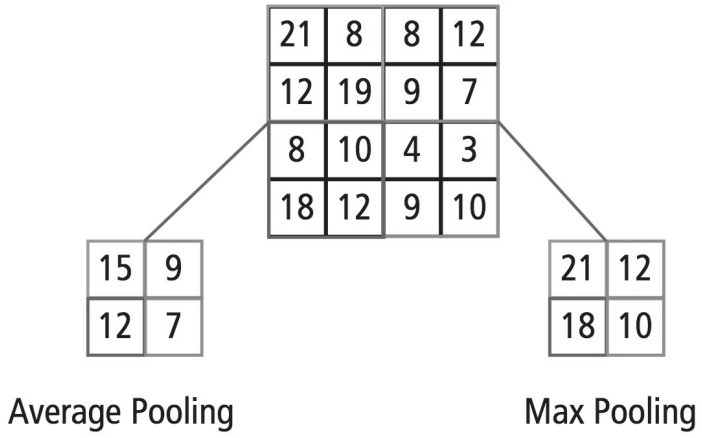


Figure 3.5 Average and max-pooling

Stride and Padding are the two terms which play an important role in the convolution neural network. Stride specifies the movement of a filter on an image, if the stride is set to one there is no size difference between the size of an image and convoluted image we will mostly ignore

stride value as one. If we increase the stride padding to two image size will be almost double of a convoluted image that depends on what is to be done at the corner of an image dataset. Padding specifies whether we have to add extra bits on the corner of an image or not. If we don't add then there is a possibility that we might lose the information so we mostly using padding as same in which we will do padding in corners by adding some extra zero's so that our filter can move easily on an image dataset. Stride and padding plays an important role convolution neural network and can increase or decrease the accuracy of a model to a large extent.

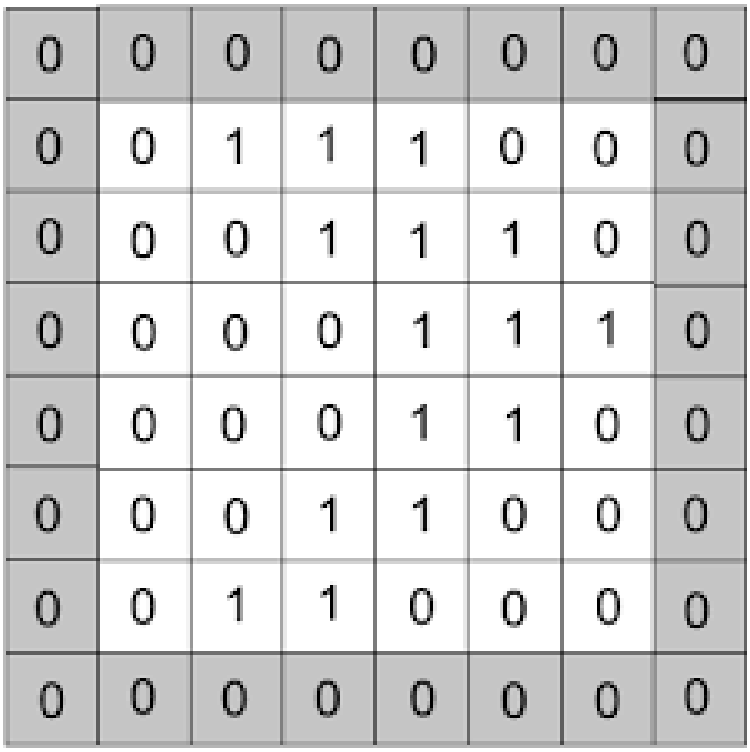


Figure 3.6 Same padding with zeros

Dense layer is a type of layer in which will all the input of current layer is connected to the next layer. In our model we have used dense layer at the end of model which will output 133 breeds of dog by calculating the probability of a certain breed, the one with maximum probability is our

dig corresponding to that image. The actual input to layer can be greater than one so we have used a sigmoid activation function to scale the value between zero and one.

Feed Forward in a model means that we pass an input to the neural network in our case it is convolution neural network, the output is predicted based on the input and intermediate weights are initialized using random distributions. Once we predicted the output we will calculate the error function by comparing the predicted output with the actual output. Based on this error we back propagate to reduce the error. This error can of several types like mean squared error, root mean squared error, categorical cross-entropy etc. There's a lot of calculation involved in this feed forward and back propagation so we have to work with CUDA in our image classifier due to the large dataset of images.

Backpropagation is a technique of optimization of error function by using various optimization techniques like stochastic gradient descent, adagrad, rmsprop, Adam etc. The main purpose is to reduce by adjusting the weights of intermediate layers which requires a lot of calculations. To perform our calculation better we have to work with CUDA enabled graphics card. In our case, we have used rmsprop optimizer to reduce the error function and increase the accuracy of our model.

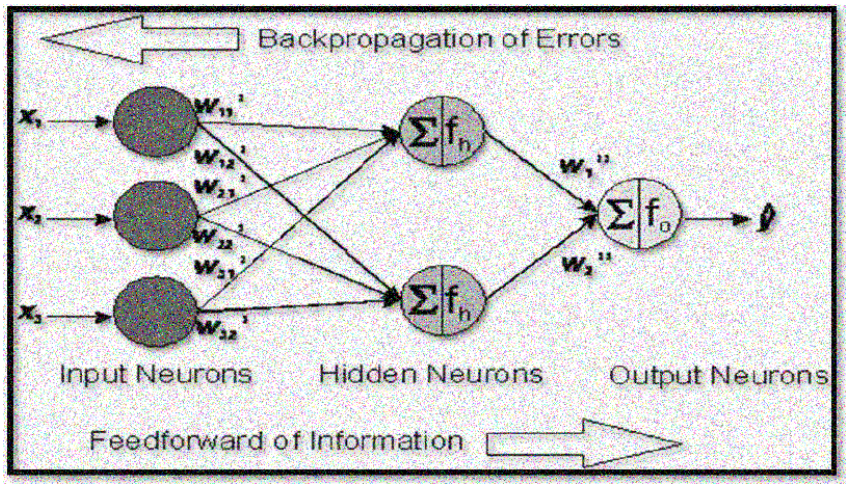


Figure 3.7 Same padding with zeros

Mathematics behind Convolution neural network:

CNN makes use of filters and pooling layer to detect the patterns within the image. Let's do a mathematical part of CNN using a simple image and filter with zero paddings and stride is 1.

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

Input Image matrix

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

Filter

So after applying the convolution layer, the filter is placed on the image and the result is computed.

Let's calculate the result for one and similarly other will be calculated.

Output=Relu((-1)*0+(-1)*1+(1)*1+(-1)*0+1*0+(-1)*0+(1)*0+(-1)*0+(-1)*0)=0

Similary , other values are calculated by moving the filter in an image.

So, final convoluted image matrix will look like

0	0	0
0	0	3
0	3	0

Convoluted Output

Steps involved in algorithms

Initially, we will have a training, testing and validation datasets. Our model configuration will be in a way that it consists convolution layer first and the attributes will be number of filter, size of kernel initializer, type of padding, the activation function and the input shape. In our case input shape is 3d because of colored image. Colored image consists of red layer, green layer and blue layer. The filter used is also spread in three dimension and convert to two dimensions. The next layer that comes into picture is max pooling layer which will convert the image to some appropriate size tensor. Max pooling will select the maximum from the filter and moved over the convoluted input that is output of the previous layer. After max pooling layer we will have consecutive convolution layer and max pooling layer. We keep on increasing number of filters as we go to deeper layer increasing the depth of an image. The training dataset is then plugged in for training purpose along with it validation data comes into the role for increasing the accuracy by avoiding the overfitting. So, the data is plugged in batches and output is predicted known as feedforward. After, this process model calculates the error by calculating the difference or by any other type. We have several error functions but the one we used is a cross categorical loss which is mostly used in classification purpose. The error is calculated and is optimized using rmsprop optimizer which backpropogates and optimizes the error function. Error function is mostly

optimized by calculating the gradient. Now, this steps repeat again and again for several time and in the end model is trained with the good weights set into the model. After the model has been undergone the training process the testing dataset comes into play. Testing dataset judges the accuracy of model by calculating total number of correct predictions divide by total predictions multiplied by hundred. The output in our case is actually probabilities of different breeds. The one breed having most probability is our output . Once, we have trained our model we will save the weights for future work. We can add this model in the backend of various websites by using flask framework or create an android app to judge which category dog breed is the one corresponding to input we plugged in.

TEST PLAN

Downloading the dataset:

So we will initially download whole dataset and break it into 3 categories training data, testing data and validation dataset. The dataset plays the major role in building and training the model. So let’s see how we downloaded the dataset.

So these are the steps involved in downloading the dataset.

1. So we will download the dataset of images of dogs which is the compressed data.
2. Decompression of data is done using !unzip command.
3. Downloading the bottleneck features of xception model.
4. Downloading compressed human dataset.
5. Uncompressing the human dataset and using it for harr cascade.

Decompression is done using !unzip command

```
!wget https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip
!wget http://vis-www.cs.umass.edu/lfw/lfw.tgz
!wget https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/DogXceptionData.npz
!unzip dogImages.zip
!7za x lfw.tgz
```

```
--2019-05-04 05:09:14--  https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip
Resolving s3-us-west-1.amazonaws.com (s3-us-west-1.amazonaws.com)... 52.219.20.5
Connecting to s3-us-west-1.amazonaws.com (s3-us-west-1.amazonaws.com)|52.219.20.5|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1132023110 (1.1G) [application/zip]
Saving to: 'dogImages.zip'

dogImages.zip       100%[===================>]   1.05G  41.9MB/s    in 26s

2019-05-04 05:09:41 (41.0 MB/s) - 'dogImages.zip' saved [1132023110/1132023110]

--2019-05-04 05:09:43--  http://vis-www.cs.umass.edu/lfw/lfw.tgz
Resolving vis-www.cs.umass.edu (vis-www.cs.umass.edu)... 128.119.244.95
Connecting to vis-www.cs.umass.edu (vis-www.cs.umass.edu)|128.119.244.95|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 180566744 (172M) [application/x-gzip]
Saving to: 'lfw.tgz'

lfw.tgz             100%[===================>] 172.20M  71.8MB/s    in 2.4s

2019-05-04 05:09:46 (71.8 MB/s) - 'lfw.tgz' saved [180566744/180566744]
```

```
2019-05-04 05:09:46 (71.8 MB/s) - 'lfw.tgz' saved [180566744/180566744]

--2019-05-04 05:09:47--  https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/DogXceptionData.npz
Resolving s3-us-west-1.amazonaws.com (s3-us-west-1.amazonaws.com)... 52.219.24.9
Connecting to s3-us-west-1.amazonaws.com (s3-us-west-1.amazonaws.com)|52.219.24.9|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3352158914 (3.1G) [application/x-www-form-urlencoded]
Saving to: 'DogXceptionData.npz'

DogXceptionData.npz 100%[===================>]   3.12G  42.2MB/s    in 85s

2019-05-04 05:11:13 (37.6 MB/s) - 'DogXceptionData.npz' saved [3352158914/3352158914]

Archive:  dogImages.zip
   creating: dogImages/
   creating: dogImages/test/
   creating: dogImages/train/
   creating: dogImages/valid/
   creating: dogImages/test/001.Affenpinscher/
  inflating: dogImages/test/001.Affenpinscher/Affenpinscher_00003.jpg
  inflating: dogImages/test/001.Affenpinscher/Affenpinscher_00023.jpg
  inflating: dogImages/test/001.Affenpinscher/Affenpinscher_00036.jpg
  inflating: dogImages/test/001.Affenpinscher/Affenpinscher_00047.jpg
  inflating: dogImages/test/001.Affenpinscher/Affenpinscher_00048.jpg
  inflating: dogImages/test/001.Affenpinscher/Affenpinscher_00058.jpg
  inflating: dogImages/test/001.Affenpinscher/Affenpinscher_00071.jpg
  inflating: dogImages/test/001.Affenpinscher/Affenpinscher_00078.jpg
   creating: dogImages/test/002.Afghan_hound/
```

Figure 3.8 Downloading and decompressing dataset

Data plays a large role in training and testing of model. Sometimes, data insufficiency may lead to wrong result. The output is predicted based on the dataset and more vast dataset and more number of dataset leads to create a good model. Neural Network model uses data to reinitialize already initialized weights and improve the accuracy metrices of model. So ,we have different types of data and how much we have each of them is shown below.

Training Data is a type of data which is used to train the network by using feed forward and backward propagation algorithms. In our training convolution neural network we use training data in combination with the validation data to reduce the over fitting of our model. So, in our case we have 6680 images as are dog data with each having their dimension as 224*224*3. The 3 depth signifies the red, blue and green. The output categories we have are 133 and the model will predict the probability of each and every category. The one breed having maximum probability is our final result. Training data may sometime overfit the model so we use image augmentation also. The data is not so sufficient as considered to number of breeds so the model may give less accuracy so we have to combine our model in front of pre trained xception model to increase the accuracy of our result. The result is evaluated based on the testing data which we will talk further.

Breeds

Affenpinscher, Afghan hound, Airedale terrier, Akita, Alaskan malamute, American eskimo dog, American foxhound, American staffordshire terrier, American water spaniel, Anatolian shepherd dog, Australian cattle dog, Australian shepherd, Australian terrier, Basenji, Basset hound, Beagle, Beauceron, Bedlington terrier, Belgian malinois, Belgian sheepdog, Belgian tervuren, Bernese mountain dog, Bichon frise, Black and tan coonhound, Black russian terrier, Bloodhound, Bluetick coonhound, Border collie, Border terrier, Borzoi, Boston terrier, Bouvier des flandres, Boykin spaniel, Briard, Brittany, Brussels griffon, Bull terrier, Bulldog, Bullmastiff, Cairn terrier, Canaan dog, Cane corso, Cardigan welsh corgi, Cavalier king charles spaniel, Chesapeake bay retriever, Chihuahua, Chinese crested, Chinese shar-pei, Clumber spaniel, Cocker spaniel, Collie,

Curly-coated retriever, Dachshund, Dalmatian, Dandie dinmont terrier, Doberman pinscher, Dogue de bordeaux, English cocker spaniel, English setter, English springer spaniel, English toy spaniel, Entlebucher mountain dog, Field spaniel, Bearded collie, Boxer, Chow chow, Finnish spitz, Icelandic sheepdog, Maltese, Pembroke welsh corgi, Flat-coated retriever, French bulldog, German pinscher, German shepherd dog, German shorthaired pointer, German wirehaired pointer, Giant schnauzer, Glen of imaal terrier, Golden retriever, Gordon setter, Great dane, Great pyrenees, Greater swiss mountain dog, Greyhound, Havanese, Ibizan hound, Irish red and white setter, Irish setter, Irish terrier, Irish water spaniel, Irish wolfhound, Italian greyhound, Japanese chin, Keeshond, Kerry blue terrier, Komondor, Kuvasz, Labrador retriever, Lakeland terrier, Leonberger, Lhasa apso, Lowchen, Manchester terrier, Mastiff, Miniature schnauzer, Neapolitan mastiff, Newfoundland, Norfolk terrier, Norwegian buhund, Norwegian elkhound, Norwegian lundehund, Norwich terrier, Nova scotia duck tolling retriever, Old english sheepdog, Otterhound, Papillon, Parson russell terrier, Pekingese, Petit basset griffon vendeen, Pharaoh hound, Plott, Pointer, Pomeranian, Poodle, Portuguese water dog, Saint bernard, Silky terrier, Smooth fox terrier, Tibetan mastiff, Welsh springer spaniel, Wirehaired pointing griffon, Xoloitzcuintli, Yorkshire terrier. So these are the 133 breeds in our model which can be predicted.

Figure 3.9 Breed from training dataset

Validation dataset is a dataset which is used to validate the data along with the training dataset. The validation dataset is usually smaller than training dataset and is used to avoid the over fitting of data. In our case validation dataset is 835 dog images and single breed is output corresponding to each breed. The maximum probability of a breed is output. So, when we start to optimize our model the model also predicts the validation loss. At the point of time when validation loss starts increasing the model save the weights at that time by using best weights the accuracy is more and over fitting is less.

`

`

Figure 3.1.1 Breed from training dataset

```
There are 133 total dog categories.
There are 8351 total dog images.

There are 6680 training dog images.
There are 835 validation dog images.
There are 836 test dog images.
```

Figure 3.1.2 Breed from training dataset

Figure 

```
'dogImages/test/097.Lakeland_terrier/Lakeland_terrier_06518.jpg'
'dogImages/test/051.Chow_chow/Chow_chow_03664.jpg'
'dogImages/test/002.Afghan_hound/Afghan_hound_00116.jpg'
'dogImages/test/019.Bedlington_terrier/Bedlington_terrier_01369.jpg'
'dogImages/test/075.Glen_of_imaal_terrier/Glen_of_imaal_terrier_05164.jpg'
'dogImages/test/133.Yorkshire_terrier/Yorkshire_terrier_08337.jpg'
'dogImages/test/071.German_shepherd_dog/German_shepherd_dog_04931.jpg'
'dogImages/test/014.Basenji/Basenji_00978.jpg'
'dogImages/test/123.Pomeranian/Pomeranian_07873.jpg'
'dogImages/test/032.Boston_terrier/Boston_terrier_02281.jpg'
'dogImages/test/091.Japanese_chin/Japanese_chin_06184.jpg'
'dogImages/test/042.Cairn_terrier/Cairn_terrier_02965.jpg'
'dogImages/test/098.Leonberger/Leonberger_06578.jpg'
'dogImages/test/069.French_bulldog/French_bulldog_04771.jpg'
'dogImages/test/056.Dachshund/Dachshund_03967.jpg'
'dogImages/test/095.Kuvasz/Kuvasz_06430.jpg'
'dogImages/test/036.Briard/Briard_02519.jpg'
'dogImages/test/125.Portuguese_water_dog/Portuguese_water_dog_07971.jpg'
'dogImages/test/019.Bedlington_terrier/Bedlington_terrier_01368.jpg'
'dogImages/test/019.Bedlington_terrier/Bedlington_terrier_01392.jpg'
'dogImages/test/092.Keeshond/Keeshond_06268.jpg'
'dogImages/test/113.Old_english_sheepdog/Old_english_sheepdog_07355.jpg'
'dogImages/test/005.Alaskan_malamute/Alaskan_malamute_00330.jpg'
'dogImages/test/109.Norwegian_elkhound/Norwegian_elkhound_07137.jpg'
'dogImages/test/029.Border_collie/Border_collie_02053.jpg'
'dogImages/test/023.Bernese_mountain_dog/Bernese_mountain_dog_01653.jpg'
'dogImages/test/106.Newfoundland/Newfoundland_07009.jpg'
'dogImages/test/080.Greater_swiss_mountain_dog/Greater_swiss_mountain_dog_05486.jpg'
'dogImages/test/035.Boykin_spaniel/Boykin_spaniel_02441.jpg'
'dogImages/test/039.Bull_terrier/Bull_terrier_02767.jpg'
'dogImages/test/004.Akita/Akita_00276.jpg'
'dogImages/test/118.Pembroke_welsh_corgi/Pembroke_welsh_corgi_07652.jpg'
'dogImages/test/081.Greyhound/Greyhound_05542.jpg'
'dogImages/test/130.Welsh_springer_spaniel/Welsh_springer_spaniel_08190.jpg'
'dogImages/test/042.Cairn_terrier/Cairn_terrier_03012.jpg'
'dogImages/test/027.Bloodhound/Bloodhound_01885.jpg'
'dogImages/test/061.English_cocker_spaniel/English_cocker_spaniel_04315.jpg'
```

Figure 3.1.4 Validation dataset sample

Testing dataset is the one which is used to finally predict the accuracy of our model. If the model predicts high accuracy then it's a good model otherwise there's a need of checking over fitting and increase the validation data. In our case the testing dataset consists of 836 images with 133 breeds to be predicted. We just plug in this testing data and calculate how many predictions are correct divided by total number of inputs multiplied by hundred. If testing data is bit augmented our classifier may give wrong predictions. The best thing is to train the model on image augmentation also. Testing can be done on simple central processing unit but training task is more of computation based on feed forward and back propagation so we require CUDA enabled graphic processing unit. The model trains at a faster rate which is better. Some, bigger models like xception, inception, resnet took many days to get train on graphic processing unit also .The graphic processing unit is costly to buy so we have to train the model online using cloud computing. In cloud computing we plug the model and training is done based on the image dataset which is set as an input to convolution neural networks. Testing plays a crucial role in development of deep neural network model. Deep reinforcement learning also works on the principal of training and testing dataset. The interaction of the agent is seen from the environment and reward is given. In case of positive reward it is good to have. Negative reward might be taken sometime. So our training dataset consists of around 800 which are sufficient for performing the testing and calculating the accuracy of model. Testing dataset is somewhat shuffled I.e augmentation of dataset is performed to have a good testing dataset and get a good insight and knowledge of our model. Profound support adapting additionally deals with the importance of preparing and testing dataset. The collaboration of the operator is seen from the environment and reward is given. If there should be an occurrence of positive reward it is a great idea to have. The negative reward may be taken at some point. So our preparation dataset comprises of around 800 which are adequate for playing out the testing and ascertaining the exactness of model. A testing dataset is to some degree rearranged I.e enlargement of a dataset is performed to have a decent testing dataset and get a decent understanding and information of our model.

```
'dogImages/test/014.Basenji/Basenji_00978.jpg'
'dogImages/test/123.Pomeranian/Pomeranian_07873.jpg'
'dogImages/test/032.Boston_terrier/Boston_terrier_02281.jpg'
'dogImages/test/091.Japanese_chin/Japanese_chin_06184.jpg'
'dogImages/test/042.Cairn_terrier/Cairn_terrier_02965.jpg'
'dogImages/test/098.Leonberger/Leonberger_06578.jpg'
'dogImages/test/069.French_bulldog/French_bulldog_04771.jpg'
'dogImages/test/056.Dachshund/Dachshund_03967.jpg'
'dogImages/test/095.Kuvasz/Kuvasz_06430.jpg'
'dogImages/test/036.Briard/Briard_02519.jpg'
'dogImages/test/125.Portuguese_water_dog/Portuguese_water_dog_07971.jpg'
'dogImages/test/019.Bedlington_terrier/Bedlington_terrier_01368.jpg'
'dogImages/test/019.Bedlington_terrier/Bedlington_terrier_01392.jpg'
'dogImages/test/092.Keeshond/Keeshond_06268.jpg'
'dogImages/test/113.Old_english_sheepdog/Old_english_sheepdog_07355.jpg'
'dogImages/test/005.Alaskan_malamute/Alaskan_malamute_00330.jpg'
'dogImages/test/109.Norwegian_elkhound/Norwegian_elkhound_07137.jpg'
'dogImages/test/029.Border_collie/Border_collie_02053.jpg'
'dogImages/test/023.Bernese_mountain_dog/Bernese_mountain_dog_01653.jpg'
'dogImages/test/106.Newfoundland/Newfoundland_07009.jpg'
'dogImages/test/080.Greater_swiss_mountain_dog/Greater_swiss_mountain_dog_05486.jp
'dogImages/test/035.Boykin_spaniel/Boykin_spaniel_02441.jpg'
'dogImages/test/039.Bull_terrier/Bull_terrier_02767.jpg'
'dogImages/test/004.Akita/Akita_00276.jpg'
'dogImages/test/118.Pembroke_welsh_corgi/Pembroke_welsh_corgi_07652.jpg'
'dogImages/test/081.Greyhound/Greyhound_05542.jpg'
'dogImages/test/130.Welsh_springer_spaniel/Welsh_springer_spaniel_08190.jpg'
'dogImages/test/042.Cairn_terrier/Cairn_terrier_03012.jpg'
'dogImages/test/027.Bloodhound/Bloodhound_01885.jpg'
```

Figure 3.1.5 Testing dataset sample

Figure 3.1.6 Testing dataset dog image

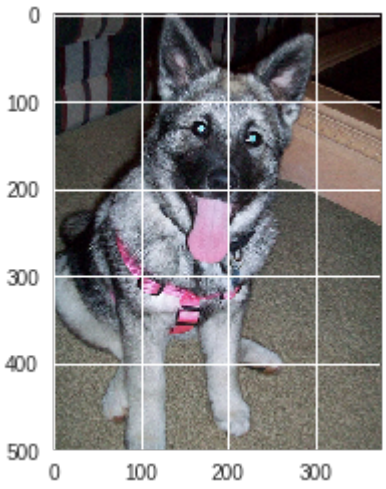


Figure 3.1.7 Testing dataset dog image

```
Train on 6680 samples, validate on 835 samples
Epoch 1/5
6660/6680 [============================>.] - ETA: 0s - loss: 4.8842 - acc: 0.0105Epoch 00001: val_loss improved from inf
to 4.86613, saving model to saved_models/weights.best.from_scratch.hdf5
6680/6680 [==============================] - 175s 26ms/step - loss: 4.8840 - acc: 0.0105 - val_loss: 4.8661 - val_acc: 0.
0168
Epoch 2/5
6660/6680 [============================>.] - ETA: 0s - loss: 4.8660 - acc: 0.0104Epoch 00002: val_loss improved from 4.86
613 to 4.84878, saving model to saved_models/weights.best.from_scratch.hdf5
6680/6680 [==============================] - 174s 26ms/step - loss: 4.8659 - acc: 0.0106 - val_loss: 4.8488 - val_acc: 0.
0168
Epoch 3/5
6660/6680 [============================>.] - ETA: 0s - loss: 4.8298 - acc: 0.0144Epoch 00003: val_loss improved from 4.84
878 to 4.81567, saving model to saved_models/weights.best.from_scratch.hdf5
6680/6680 [==============================] - 174s 26ms/step - loss: 4.8299 - acc: 0.0144 - val_loss: 4.8157 - val_acc: 0.
0168
Epoch 4/5
6660/6680 [============================>.] - ETA: 0s - loss: 4.7898 - acc: 0.0165Epoch 00004: val_loss improved from 4.81
567 to 4.81315, saving model to saved_models/weights.best.from_scratch.hdf5
6680/6680 [==============================] - 174s 26ms/step - loss: 4.7898 - acc: 0.0165 - val_loss: 4.8131 - val_acc: 0.
0228
Epoch 5/5
6660/6680 [============================>.] - ETA: 0s - loss: 4.7599 - acc: 0.0219Epoch 00005: val_loss improved from 4.81
315 to 4.77454, saving model to saved_models/weights.best.from_scratch.hdf5
6680/6680 [==============================] - 174s 26ms/step - loss: 4.7601 - acc: 0.0220 - val_loss: 4.7745 - val_acc: 0.
0192
```

Figure 3.1.8 Training and validation alongside

Chapter-4 RESULT AND PERFORMANCE ANALYSIS

Analysis

Result of our model truly depends on type of data we are using. The training data should be augmented while training our model. The model has a high chance that it might not detect the output for flipped image. In this case we randomly flip some images at an angle and try to train the model and along with validation keeps on going. Accuracy merely depends upon the type of data we are using and type of data we are testing on.

Accuracy

We have constructed our model using convolution neural network but the dataset is very less as compared to the number of breeds so the accuracy of our model is about 2 percent which is very less for any deep learning model. The accuracy is very low due to an insufficiency of data and image augmentation. So, we tried different methods to improve such a low accuracy. Accuracy plays a major role to tell how well the model has set the weights. In the field of deep learning even the experts don’t know the initial values of hyper parameters to be initialized to get a great accuracy. Accuracy depends upon various hyper parameters like initialization of weights, number of epochs, learning rate. If we initialize all weights to one it will a very bad accuracy to our model so we use random distributions like guassian, normal. Uniform depends on the type of problem we want to solve.

Method 1

We have not used any prebuild model and made our model and computed the accuracy. The accuracy is very low about 2 percent which is fine for the various reasons. Dataset is very less in

comparison to number of breeds and no CUDA enabled graphic card is available so if complex model is build it will halt the whole laptop. CUDA will help the model to train itself parallel on the cores of graphic card. Graphic card contains of about 1000 cores which can make our computation fast.

Method 2

We have used VGG16 prebuild model and made our model and computed the accuracy. The accuracy is still low about 43 percent which is fine for the various reasons. Dataset is very less in comparison to number of breeds and no CUDA enabled graphic card is available so if complex model is build it will halt the whole laptop. CUDA will help the model to train itself parallel on the cores of graphic card. Graphic card contains of about 1000 cores which can make our computation fast.

Method 3

We have used Xception pre build model and made our model and computed the accuracy. The accuracy is still low about 85 percent which is fine for the various reasons. Dataset is very less in comparison to number of breeds and no CUDA enabled graphic card is available so if complex model is build it will halt the whole laptop. CUDA will help the model to train itself parallel on the cores of graphic card. Graphic card contains of about 1000 cores which can make our computation fast.

```
Train on 6680 samples, validate on 835 samples
Epoch 1/5
6660/6680 [============================>.] - ETA: 0s - loss: 4.8842 - acc: 0.0105Epoch 00001: val_loss improved from ir
to 4.86613, saving model to saved_models/weights.best.from_scratch.hdf5
6680/6680 [==============================] - 175s 26ms/step - loss: 4.8840 - acc: 0.0105 - val_loss: 4.8661 - val_acc:
0168
Epoch 2/5
6660/6680 [============================>.] - ETA: 0s - loss: 4.8660 - acc: 0.0104Epoch 00002: val_loss improved from 4.
613 to 4.84878, saving model to saved_models/weights.best.from_scratch.hdf5
6680/6680 [==============================] - 174s 26ms/step - loss: 4.8659 - acc: 0.0106 - val_loss: 4.8488 - val_acc:
0168
Epoch 3/5
6660/6680 [============================>.] - ETA: 0s - loss: 4.8298 - acc: 0.0144Epoch 00003: val_loss improved from 4.
878 to 4.81567, saving model to saved_models/weights.best.from_scratch.hdf5
6680/6680 [==============================] - 174s 26ms/step - loss: 4.8299 - acc: 0.0144 - val_loss: 4.8157 - val_acc:
0168
Epoch 4/5
6660/6680 [============================>.] - ETA: 0s - loss: 4.7898 - acc: 0.0165Epoch 00004: val_loss improved from 4.
567 to 4.81315, saving model to saved_models/weights.best.from_scratch.hdf5
6680/6680 [==============================] - 174s 26ms/step - loss: 4.7898 - acc: 0.0165 - val_loss: 4.8131 - val_acc:
0228
Epoch 5/5
6660/6680 [============================>.] - ETA: 0s - loss: 4.7599 - acc: 0.0219Epoch 00005: val_loss improved from 4.
315 to 4.77454, saving model to saved_models/weights.best.from_scratch.hdf5
6680/6680 [==============================] - 174s 26ms/step - loss: 4.7601 - acc: 0.0220 - val_loss: 4.7745 - val_acc:
0192

<keras.callbacks.History at 0x7fa2ab9b08d0>
```

Figure 4.1 Training accuracy about 20 percent in method 1 on training dataset

```
Train on 6680 samples, validate on 835 samples
Epoch 1/20
6480/6680 [============================>.] - ETA: 0s - loss: 11.7961 - acc: 0.1346Epoch 00001: val_loss improved from inf
to 10.04986, saving model to saved_models/weights.best.VGG16.hdf5
6680/6680 [==============================] - 2s 263us/step - loss: 11.7496 - acc: 0.1377 - val_loss: 10.0499 - val_acc: 0
.2431
Epoch 2/20
6620/6680 [============================>.] - ETA: 0s - loss: 9.5869 - acc: 0.3086Epoch 00002: val_loss improved from 10.0
4986 to 9.39210, saving model to saved_models/weights.best.VGG16.hdf5
6680/6680 [==============================] - 2s 242us/step - loss: 9.5698 - acc: 0.3102 - val_loss: 9.3921 - val_acc: 0.3
150
Epoch 3/20
6460/6680 [============================>.] - ETA: 0s - loss: 9.0465 - acc: 0.3715Epoch 00003: val_loss improved from 9.39
210 to 9.15534, saving model to saved_models/weights.best.VGG16.hdf5
6680/6680 [==============================] - 2s 240us/step - loss: 9.0386 - acc: 0.3717 - val_loss: 9.1553 - val_acc: 0.3
437
Epoch 4/20
6460/6680 [============================>.] - ETA: 0s - loss: 8.7492 - acc: 0.4110Epoch 00004: val_loss improved from 9.15
534 to 9.05054, saving model to saved_models/weights.best.VGG16.hdf5
6680/6680 [==============================] - 2s 239us/step - loss: 8.7750 - acc: 0.4097 - val_loss: 9.0505 - val_acc: 0.3
461
Epoch 5/20
6460/6680 [============================>.] - ETA: 0s - loss: 8.6197 - acc: 0.4294Epoch 00005: val_loss improved from 9.05
054 to 8.91443, saving model to saved_models/weights.best.VGG16.hdf5
6680/6680 [==============================] - 2s 240us/step - loss: 8.6249 - acc: 0.4295 - val_loss: 8.9144 - val_acc: 0.3
844
Epoch 6/20
6500/6680 [============================>.] - ETA: 0s - loss: 8.5192 - acc: 0.4446Epoch 00006: val_loss did not improve
6680/6680 [==============================] - 2s 237us/step - loss: 8.5083 - acc: 0.4448 - val_loss: 8.9700 - val_acc: 0.3
653
Epoch 7/20
6480/6680 [============================>.] - ETA: 0s - loss: 8.3169 - acc: 0.4562Epoch 00007: val_loss improved from 8.91
443 to 8.61775, saving model to saved_models/weights.best.VGG16.hdf5
6680/6680 [==============================] - 2s 239us/step - loss: 8.3257 - acc: 0.4554 - val_loss: 8.6177 - val_acc: 0.3
808
Epoch 8/20
6460/6680 [============================>.] - ETA: 0s - loss: 8.0891 - acc: 0.4706Epoch 00008: val_loss improved from 8.61
775 to 8.38292, saving model to saved_models/weights.best.VGG16.hdf5
6680/6680 [==============================] - 2s 240us/step - loss: 8.0695 - acc: 0.4714 - val_loss: 8.3829 - val_acc: 0.4
000
Epoch 9/20
6480/6680 [============================>.] - ETA: 0s - loss: 7.8992 - acc: 0.4877Epoch 00009: val_loss improved from 8.38
292 to 8.32290, saving model to saved_models/weights.best.VGG16.hdf5
6680/6680 [==============================] - 2s 240us/step - loss: 7.8756 - acc: 0.4891 - val_loss: 8.3229 - val_acc: 0.4
144
Epoch 10/20
6480/6680 [============================>.] - ETA: 0s - loss: 7.8377 - acc: 0.4960Epoch 00010: val_loss improved from 8.32
290 to 8.23419, saving model to saved_models/weights.best.VGG16.hdf5
6680/6680 [==============================] - 2s 239us/step - loss: 7.8156 - acc: 0.4970 - val_loss: 8.2342 - val_acc: 0.4
144
Epoch 11/20
6460/6680 [============================>.] - ETA: 0s - loss: 7.7163 - acc: 0.5062Epoch 00011: val_loss did not improve
6680/6680 [==============================] - 2s 239us/step - loss: 7.6977 - acc: 0.5073 - val_loss: 8.2556 - val_acc: 0.4
180
Epoch 12/20
6440/6680 [============================>..] - ETA: 0s - loss: 7.6583 - acc: 0.5123Epoch 00012: val_loss improved from 8.23
419 to 8.11279, saving model to saved_models/weights.best.VGG16.hdf5
6680/6680 [==============================] - 2s 240us/step - loss: 7.6324 - acc: 0.5141 - val_loss: 8.1128 - val_acc: 0.4
287
Epoch 13/20
6480/6680 [============================>.] - ETA: 0s - loss: 7.5847 - acc: 0.5193Epoch 00013: val_loss did not improve
6680/6680 [==============================] - 2s 238us/step - loss: 7.5985 - acc: 0.5183 - val_loss: 8.2157 - val_acc: 0.4
108
Epoch 14/20
6460/6680 [============================>.] - ETA: 0s - loss: 7.5377 - acc: 0.5232Epoch 00014: val_loss did not improve
6680/6680 [==============================] - 2s 239us/step - loss: 7.5226 - acc: 0.5243 - val_loss: 8.1645 - val_acc: 0.4
299
```

```
Epoch 15/20
6500/6680 [============================>.] - ETA: 0s - loss: 7.4959 - acc: 0.5266Epoch 00015: val_loss improved from 8.11
279 to 8.09030, saving model to saved_models/weights.best.VGG16.hdf5
6680/6680 [==============================] - 2s 239us/step - loss: 7.4967 - acc: 0.5268 - val_loss: 8.0903 - val_acc: 0.4
263
Epoch 16/20
6460/6680 [============================>.] - ETA: 0s - loss: 7.4953 - acc: 0.5288Epoch 00016: val_loss improved from 8.09
030 to 8.00362, saving model to saved_models/weights.best.VGG16.hdf5
6680/6680 [==============================] - 2s 239us/step - loss: 7.4833 - acc: 0.5295 - val_loss: 8.0036 - val_acc: 0.4
347
Epoch 17/20
6460/6680 [============================>.] - ETA: 0s - loss: 7.4842 - acc: 0.5328Epoch 00017: val_loss did not improve
6680/6680 [==============================] - 2s 238us/step - loss: 7.4720 - acc: 0.5335 - val_loss: 8.0432 - val_acc: 0.4
407
Epoch 18/20
6480/6680 [============================>.] - ETA: 0s - loss: 7.4531 - acc: 0.5346Epoch 00018: val_loss improved from 8.00
362 to 8.00013, saving model to saved_models/weights.best.VGG16.hdf5
6680/6680 [==============================] - 2s 238us/step - loss: 7.4712 - acc: 0.5335 - val_loss: 8.0001 - val_acc: 0.4
443
Epoch 19/20
6500/6680 [============================>.] - ETA: 0s - loss: 7.4852 - acc: 0.5332Epoch 00019: val_loss improved from 8.00
013 to 7.98943, saving model to saved_models/weights.best.VGG16.hdf5
6680/6680 [==============================] - 2s 241us/step - loss: 7.4656 - acc: 0.5343 - val_loss: 7.9894 - val_acc: 0.4
419
Epoch 20/20
6460/6680 [============================>.] - ETA: 0s - loss: 7.4683 - acc: 0.5316Epoch 00020: val_loss improved from 7.98
943 to 7.94966, saving model to saved_models/weights.best.VGG16.hdf5
6680/6680 [==============================] - 2s 241us/step - loss: 7.4369 - acc: 0.5335 - val_loss: 7.9497 - val_acc: 0.4
527
```

Figure 4.2 Training accuracy about 53 percent using method 2 on training set

```
Epoch 9/20
6528/6680 [============================>.] - ETA: 0s - loss: 0.1149 - acc: 0.9717Epoch 00009: val_loss did not improve
6680/6680 [==============================] - 2s 251us/step - loss: 0.1154 - acc: 0.9716 - val_loss: 0.4558 - val_acc: 0.8
563
Epoch 10/20
6528/6680 [============================>.] - ETA: 0s - loss: 0.0987 - acc: 0.9779Epoch 00010: val_loss did not improve
6680/6680 [==============================] - 2s 249us/step - loss: 0.0990 - acc: 0.9778 - val_loss: 0.4524 - val_acc: 0.8
623
Epoch 11/20
6528/6680 [============================>.] - ETA: 0s - loss: 0.0858 - acc: 0.9804Epoch 00011: val_loss did not improve
6680/6680 [==============================] - 2s 264us/step - loss: 0.0867 - acc: 0.9802 - val_loss: 0.4489 - val_acc: 0.8
623
Epoch 12/20
6528/6680 [============================>.] - ETA: 0s - loss: 0.0746 - acc: 0.9853Epoch 00012: val_loss did not improve
6680/6680 [==============================] - 2s 261us/step - loss: 0.0745 - acc: 0.9853 - val_loss: 0.4626 - val_acc: 0.8
599
Epoch 13/20
6528/6680 [============================>.] - ETA: 0s - loss: 0.0659 - acc: 0.9859Epoch 00013: val_loss did not improve
6680/6680 [==============================] - 2s 262us/step - loss: 0.0655 - acc: 0.9861 - val_loss: 0.4711 - val_acc: 0.8
563
Epoch 14/20
6528/6680 [============================>.] - ETA: 0s - loss: 0.0575 - acc: 0.9888Epoch 00014: val_loss did not improve
6680/6680 [==============================] - 2s 254us/step - loss: 0.0576 - acc: 0.9889 - val_loss: 0.4780 - val_acc: 0.8
635
Epoch 15/20
6528/6680 [============================>.] - ETA: 0s - loss: 0.0511 - acc: 0.9902Epoch 00015: val_loss did not improve
6680/6680 [==============================] - 2s 250us/step - loss: 0.0509 - acc: 0.9904 - val_loss: 0.4832 - val_acc: 0.8
575
Epoch 16/20
6528/6680 [============================>.] - ETA: 0s - loss: 0.0440 - acc: 0.9922Epoch 00016: val_loss did not improve
6680/6680 [==============================] - 2s 254us/step - loss: 0.0444 - acc: 0.9922 - val_loss: 0.4917 - val_acc: 0.8
635
Epoch 17/20
6528/6680 [============================>.] - ETA: 0s - loss: 0.0385 - acc: 0.9928Epoch 00017: val_loss did not improve
6680/6680 [==============================] - 2s 250us/step - loss: 0.0384 - acc: 0.9930 - val_loss: 0.4881 - val_acc: 0.8
707
Epoch 18/20
6528/6680 [============================>.] - ETA: 0s - loss: 0.0354 - acc: 0.9940Epoch 00018: val_loss did not improve
6680/6680 [==============================] - 2s 259us/step - loss: 0.0349 - acc: 0.9942 - val_loss: 0.4995 - val_acc: 0.8
659
Epoch 19/20
6528/6680 [============================>.] - ETA: 0s - loss: 0.0311 - acc: 0.9951Epoch 00019: val_loss did not improve
6680/6680 [==============================] - 2s 265us/step - loss: 0.0311 - acc: 0.9951 - val_loss: 0.5041 - val_acc: 0.8
599
Epoch 20/20
6528/6680 [============================>.] - ETA: 0s - loss: 0.0275 - acc: 0.9957Epoch 00020: val_loss did not improve
6680/6680 [==============================] - 2s 264us/step - loss: 0.0275 - acc: 0.9958 - val_loss: 0.5143 - val_acc: 0.8
635

<keras.callbacks.History at 0x7fdea4261240>
```

Figure 4.3 Training accuracy about 99 percent on method 3 on training data

Methods used are basically based on mathematical modeling, but as a deep learning student I know that for getting the accuracy we have to try experimentation by constructing different

number of layers and playing around with hyper parameters . So the three above model used are based on experimentation .There are many models which leads to almost same result, but the chosen xception model is a little better than other models.

Comparison with different methods

Since dataset is very small pre build trained model is not effective to compute the breed. In the case of method 1 we have directly constructed the model without using transfer learning. Method 1 gave us the accuracy of about 2 percent which is not at all acceptable. Method 2 improved the accuracy because we have used the transfer learning. In method 2 accuracy was about 43 percent which has increased in comparison to the first model. In method 3 accuracy has increased to 85 percent due to the usage of powerful xception model. Xception model took several days to get trained.

Plot between epochs and accuracy

```
import matplotlib.pyplot as plt


# Plot training & validation accuracy values
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()

# Plot training & validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```

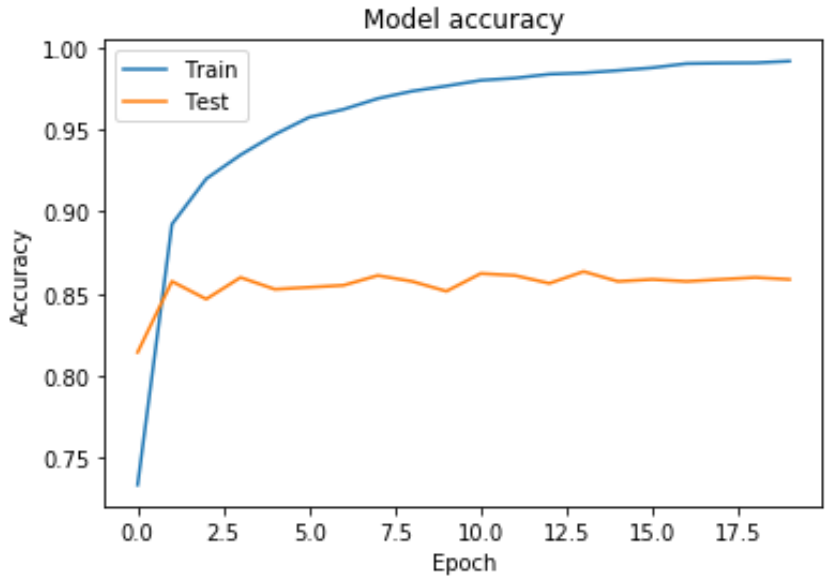


figure 4.4 Accuracy and epochs comparison

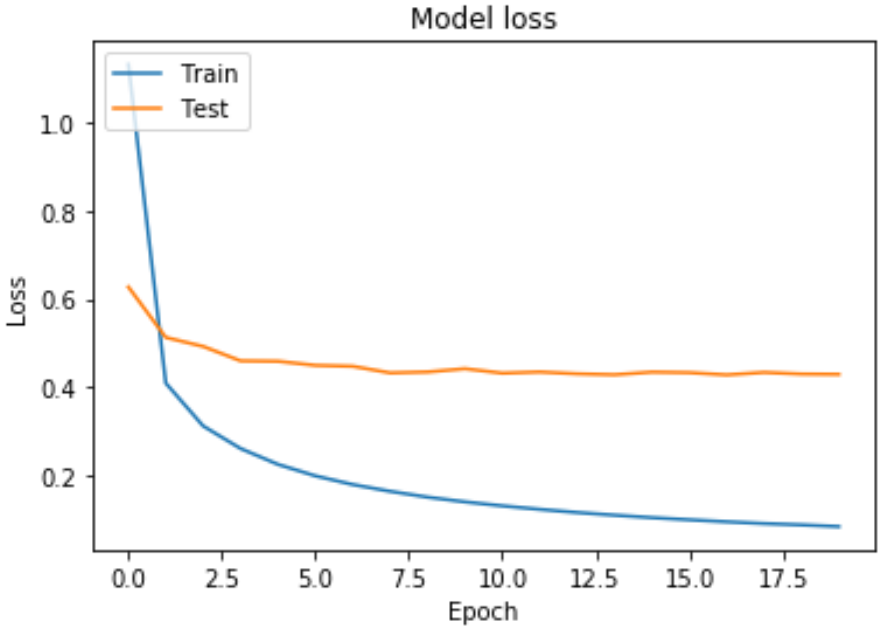


figure 4.5 Loss and epochs comparison

Result with output:

```
Xception_predict_breed('ter.jpg')
```

```
Dog breed is Bull_terrier
```

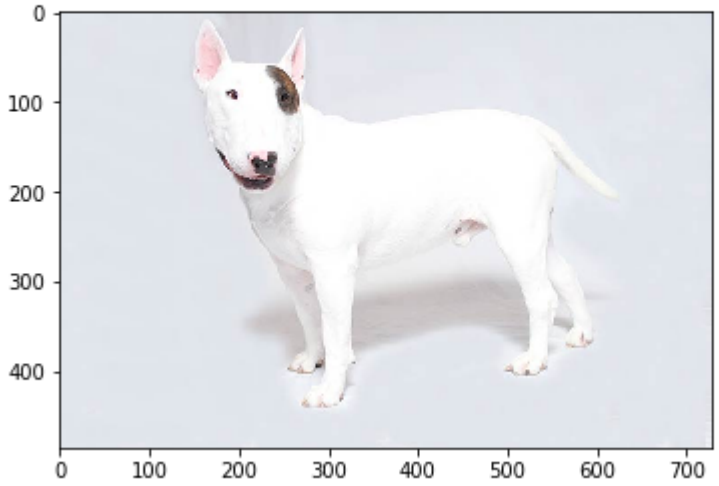
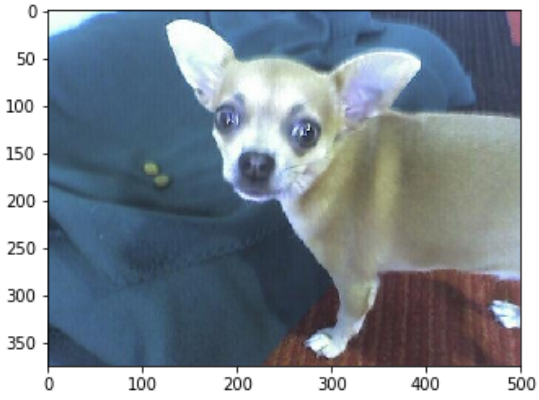


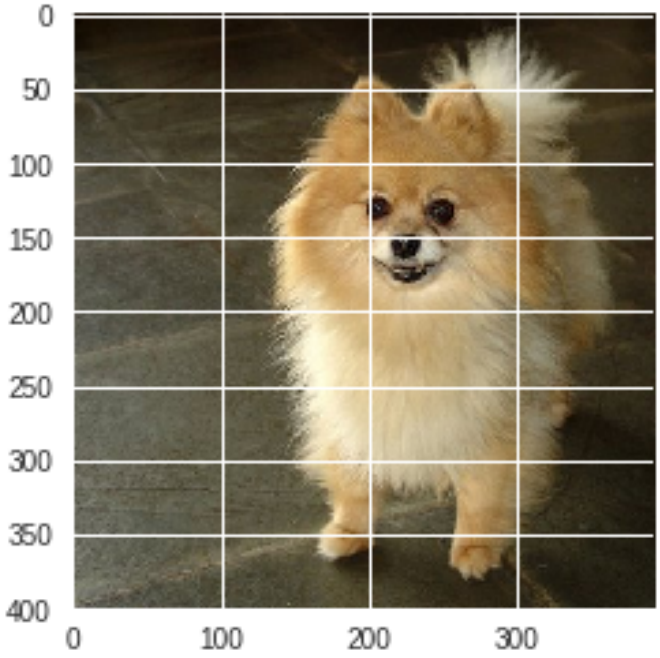
Figure 4.6 Predicting the breed

```
Xception_predict_breed('dogImages/train/048.Chihuahua/Chihuahu
```

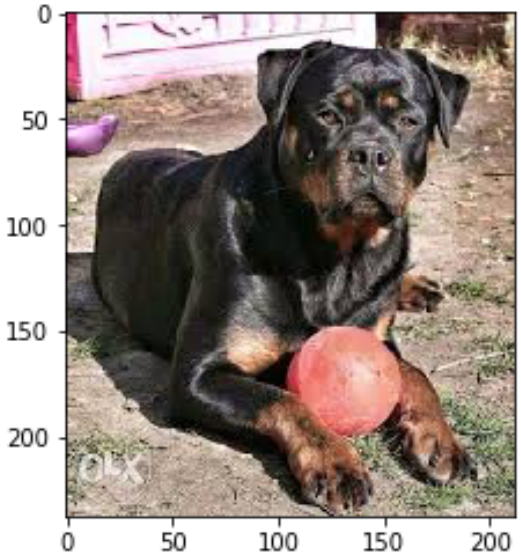
```
Downloading data from https://github.com/fchollet/deep-learnir
83689472/83683744 [==============================] - 1s 0us/st
Dog breed is Chihuahua
```



Dog breed is Pomeranian



Dog breed is Cane_corso



Chapter-7 CONCLUSION

Conclusions

In the end, we concluded that deep learning models have a very great capability to surpass the human potential if the data provided is sufficient. Engineers and scientists are still working on the deep learning field because till now the exploration of deep learning is limited. In the future, the deep learning will create another deep learning models on its own and deep learning model will write codes and surpass the human capabilities. Deep learning has a lot of scope in medical sciences by analyzing the images by deep convolution neural network. Deep learning may be one of the possible reason for the destruction of humankind. Dog breed classifier is one of the mini projects of deep learning developed using xception model and advance neural networks. Transfer learning has a great scope in the future by combining a prebuilt model with the model we constructed.

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 hyperparameters optimization", *Proc. Adv. Neural Inf. Process. Syst.*, pp. 2546-2554, 2011.

[3] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," in IEEE Transactions on Knowledge 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.

[7] A. Krizhevsky, I. Sutskever, G. E. Hinton, "Imagenet classification with deep convolutional neural networks", *Proc. Adv. Neural Inf. Process. Syst.*, pp. 1097-1105, 2012.

[8] J. Li, X. Liang, Y. Wei, T. Xu, J. Feng, S. Yan, "Perceptual generative adversarial networks for small object detection", *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 1951-1959, Jul. 2017.

[9] H. Li, Z. Lin, X. Shen, J. Brandt, G. Hua, "A convolutional neural network cascade for face detection", *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 5325-5334, Jun. 2015.

[10] C. Szegedy et al., "Going deeper with convolutions," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, 2015, pp. 1-9. doi: 10.1109/CVPR.2015.7298594