Final Report: Image Classification Using Convolutional Neural Networks

Submitted to:

Prof. Qun Liu

School of Computing

Dublin City University

Report Prepared By:

Juhi Shrivastava

16212548

Juhi.shrivastava2@mail.dcu.ie

Module: Machine Learning, CA684

April 23, 2017

Declaration:

In submitting this project, I declare that the project material, which I now submit, is my own work. I make this declaration in the knowledge that a breach of the rules pertaining to project submission may carry serious consequences. I have also identified and included the source of all facts, ideas, opinions, viewpoints of others in the assignment references.

TABLE OF CONTENTS:

1 INTRODUCTION:

Some of the most ingenious inventions today in the field of Machine Learning are driven by the Deep Learning (DL) and Neural Networks(NN) owing to their implausible capacity to learn from the data and high adaptability to various data types. Application of Deep Learning in the field of Computer Vision have taken the technology to altogether a new pinnacle like Automated Face Tagging as used by Facebook and Object search used in the Google Photos. This project aims at understanding the Image Recognition by Classifying the Images of the Cats and Dogs into their respective classes using the dataset acquired from Kaggle's Dogs vs. Cats Competition. A machine algorithm will be fed with the raw images that have been labelled already using Convolutional Neural Networks which have a unique architecture particularly suitable for the Images have been used for accomplishing this task of Classification.

1.1 Resources and Structure of the Report:

RESOURCES:

Github Repository:
https://github.com/juhi23/CA684_End_SEM_Project_Image_Recognition

Technology Used: Python 3.5
Anaconda Jupyter Notebook
TensorFlow

Video link:
https://drive.google.com/a/mail.dcu.ie/file/d/0B2qpd-iskXzNOGJXQjlYV0FPMms/view?usp=sharing

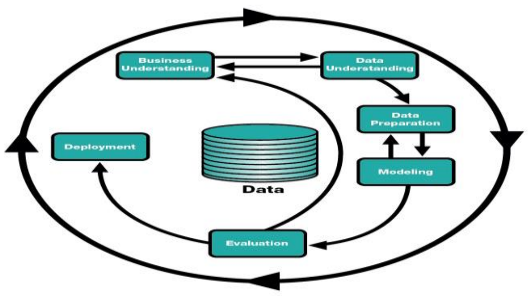
Presentation Slides: CA684_16212548_final_presentation.docx

STRUCTURE OF THE REPORT:

Methodology Used: CRISP-DM Lifecycle.

The report is structured covering all the Development steps, according to the CRISP-DM Methodology in the below sections to ensure a systematic approach.

- Business Understanding
- Data Understanding
- Data Preparation
- Modelling
- Evaluation
- Deployment



2 BUSINESS UNDERSTANDING:

2.1 What is Image Classification?

Computers do not perceive the images as we Humans do with our eyes. It is easy for humans to differentiate a cat from dog by looking at them but evidently it is particularly difficult to tell cats and dogs apart automatically. Image Classification is a process of identifying the images based on the assumption that the Image to be identified has a number of features and these features belong to one or more distinct class. Image Classification analyses these features numerically and categorizes them into the classes.

2.2 Challenges:

For a computer, a colour picture is just a 3-dimensional array of numbers to a computer. So, if a computer wants to distinguish between the images (12,288 independent integers (64x64x3)), there are some roadblock while doing the object identification:

- Variations in the viewpoint

- Difference in Illumination

- Hidden Parts of image

- Background Clutter

To overcome these roadblocks, the technique most popularly used is Convolutional Neural Networks. It is a pillar algorithm for image classification in deep learning and by far the best models available for perceptual problems. As the CNN has the ability to learn from scratch there is no need for custom feature engineering.

3 DATA UNDERSTANDING:

3.1 Dataset Link

https://www.kaggle.com/c/dogs-vs-cats/data

3.2 Understanding the dataset

This dataset is to solve the CAPTCHA – Completely Automated Public Turing test to tell Computers and Humans Apart and has a very high dimensional input samples – Images.
There are 2 categories of the Images – **Cats and Dogs**
The data is split into training and Test as:
Training Data: 25000 images of cats and dogs with label cat and dog
Test data: 12500 images of cats and dogs with no label.

4 DATA PREPARATION:

4.1 Shuffling:

As the format has images of cats followed by images of dogs and total is 25000, the very first images of the training session will be only of cats and will be of single class, so the model will not learn anything therefore it is essential to shuffle the data to have nice merge of cats and dogs.

4.2 Train – Test – Validation split:

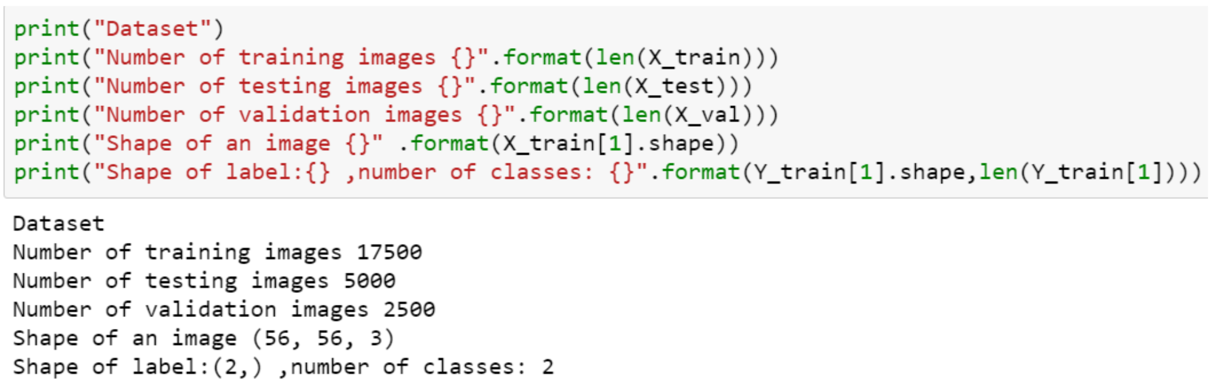
Train -- 70% of data
Test -- 30% of data
Validation -- 10% of data

4.3 Dimension reduction

The Dimension of the Images i.e the Image shape in terms of pixels is reduced to 56*56*3, otherwise a lot of RAM will be use slowing down the training process of the Algorithm.

4.4 Categorical to Binary Vectors

The images(Training_Data) are loaded with the labels and these labels are converted into Binary Vectors:
Cat - [0 , 1]
Dog – [1 , 0]



```
print("Dataset")
print("Number of training images {}".format(len(X_train)))
print("Number of testing images {}".format(len(X_test)))
print("Number of validation images {}".format(len(X_val)))
print("Shape of an image {}" .format(X_train[1].shape))
print("Shape of label:{} ,number of classes: {}".format(Y_train[1].shape,len(Y_train[1])))
```

```
Dataset
Number of training images 17500
Number of testing images 5000
Number of validation images 2500
Shape of an image (56, 56, 3)
Shape of label:(2,) ,number of classes: 2
```

5 MODELLING:

The Model used for the Image Recognition Task of the Cat and Dog Images is Convolution Neural network. The below section first explains what a CNN is and how it works and later on explains the how the Modelling is done for this particular Project:

5.1 Why Convolutional Neural Networks?

There are traditional approaches for enhancing the computer vision like the KNN(K-Nearest Neighbour) and they are only suitable for the simple tasks as they have some drawbacks attached to them like:

- If the location of the object in the 2 different images is different than the KNN will give highly non-zero distance for the images, hence both the images might get classified into different classes.
- Linear Classifiers considers each pixel of the image as a parameter therefore, it will be difficult for the classifiers to overcome the roadblocks as mentioned in the section above.

Due to the vast variations in the images and the drawbacks of the traditional approaches, a newer approach which consecutively models small pieces of information and combines them into a network is evolved named as Convolutional Neural Networks.

5.2 How CNN Works?

In a CNN, the first layer tries to **Detect Edges** and forms templates for detecting the edges. The next set of layers combines these edges into **Simpler Shapes** and forms templates for different object **position, scale, illumination etc.** The final layers match the input with all these templates generated by the previous layers and tries to **predict the image** which is a weighted sum of all the previous templates.

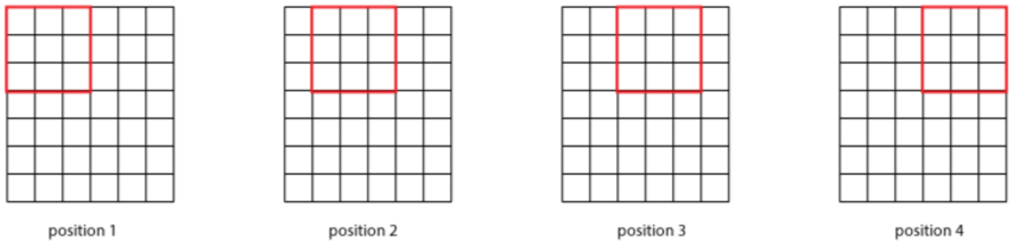
A CNN Consists of three layers:

- Convolutional Layer – A Convolutional Layer can be picturized in the form of a cuboid with depth of 3 depicting the 3 colours – Red, Green, Blue. A Filter is run

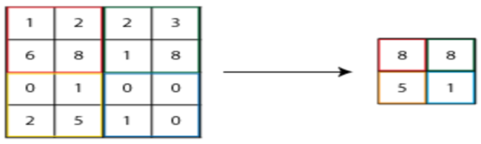
over the image which is another block of cuboid with smaller height and depth. The filters are learned using the **Backpropogation.** In general, if an image has dimension NxNxd and filter's dimensions are FxFxd. The stride is (S), which is the number of cells to move in each step.
The size of the output will be:

output size = (N – F)/S + 1

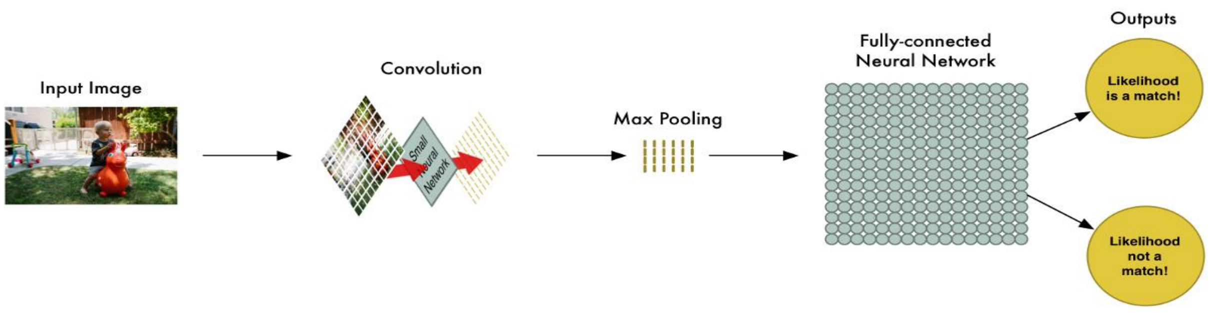
The Hyperparameters to be considered while designing the Convolution Layer are:
- Number of Filters
- Filter Size
- Stride
- Amount of Padding (additional zero-value pixels at the border of the image)



- Pooling Layer – A Pooling Layer is required to reduce the size of the image as the use of padding in the convolutional Layer makes the size of the image constant. Generally Max pooling is used in which the matrix is divided into smaller matrices with the maximum value.



- Fully Connected Layer – After the convolutional and the Max Pooling Layer, fully-connected Layer is used by the Network in which each pixel is treated as a separate neuron. It will contain as many neurons as the number of the class to be predicted.



So, a CNN owing to the deep structure, models complex variations and behaviours of images giving highly accurate predictions.

5.3 Tensor Flow:

For the implementation of the Network Architecture chosen, Tensor flow is Used. As training a large neural network is expensive in terms of computational speed there Google's Tensor flow is used which is built for speed. As the Tensor Flow doesn't run in Python therefore sessions are created and passed to the backend for processing.
Library: TFLEARN
TENSORFLOW

5.4 Network Architecture:

The Network Architecture used for this task is a Convolutional Neural Network which is a modified version of ALEXNET. Alexnet was the winning solution of the IMAGENET challenge in 2012.
ALEXNET: It is a CNN which runs on GPU's implemented in CUDA. Alexnet contains in total 8 layers out of which 5 are the Convolutional Layers and remaining are the fully connected layers.

Modified Version i.e the architecture used for the prediction contains:

- 3 Convolutional Layers
- 2 Fully connected layer
- 1 Softmax output layer

5.5 CNN Layers – Hyperparameters:

- **Layer 1:** Convolutional Layer 1
  Hyperparameters:
  Filters = 64
  Filter_size = 5
  Strides = [1,1,1,1]
  Padding = 'same' (Tries to pad zeros evenly left and right)
  Activation = 'relu' (Rectified Linear activation Function)
  Regularizer = "L2"

- **Max Pooling Layer 1:** Ouput of Layer 1

- **Layer 2:** Convolutional Layer 2
  Hyperparameters:
  Filters = 128
  Filter_size = 5
  Strides = [1,1,1,1]

Padding = 'same' (Tries to pad zeros evenly left and right)
Activation = 'relu' (Rectified Linear activation Function)
Regularizer = "L2"

- **Max Pooling Layer 2:** Ouput of Layer 2

- **Layer 3:** Convolutional Layer 3
  Hyperparameters:
  Filters = 128
  Filter_size = 5
  Strides = [1,1,1,1]
  Padding = 'same' (Tries to pad zeros evenly left and right)
  Activation = 'relu' (Rectified Linear activation Function)
  Regularizer = "L2" (Prevent Overfitting)

- **Layer 4:** Fully Connected Layer 1
  Hyperparameters:
  Activation = 'relu' (Rectified Linear activation Function)

- **Dropout Layer:** Prevent Overfitting in Layer 4

- **Layer 5:** Fully Connected Layer 2
  Hyperparameters:
  Activation = 'relu' (Rectified Linear activation Function)

- **Dropout Layer:** Prevent Overfitting in Layer 4

- **Layer 6: Softmax Layer**
  Hyperparameters:
  Activation = “Softmax”

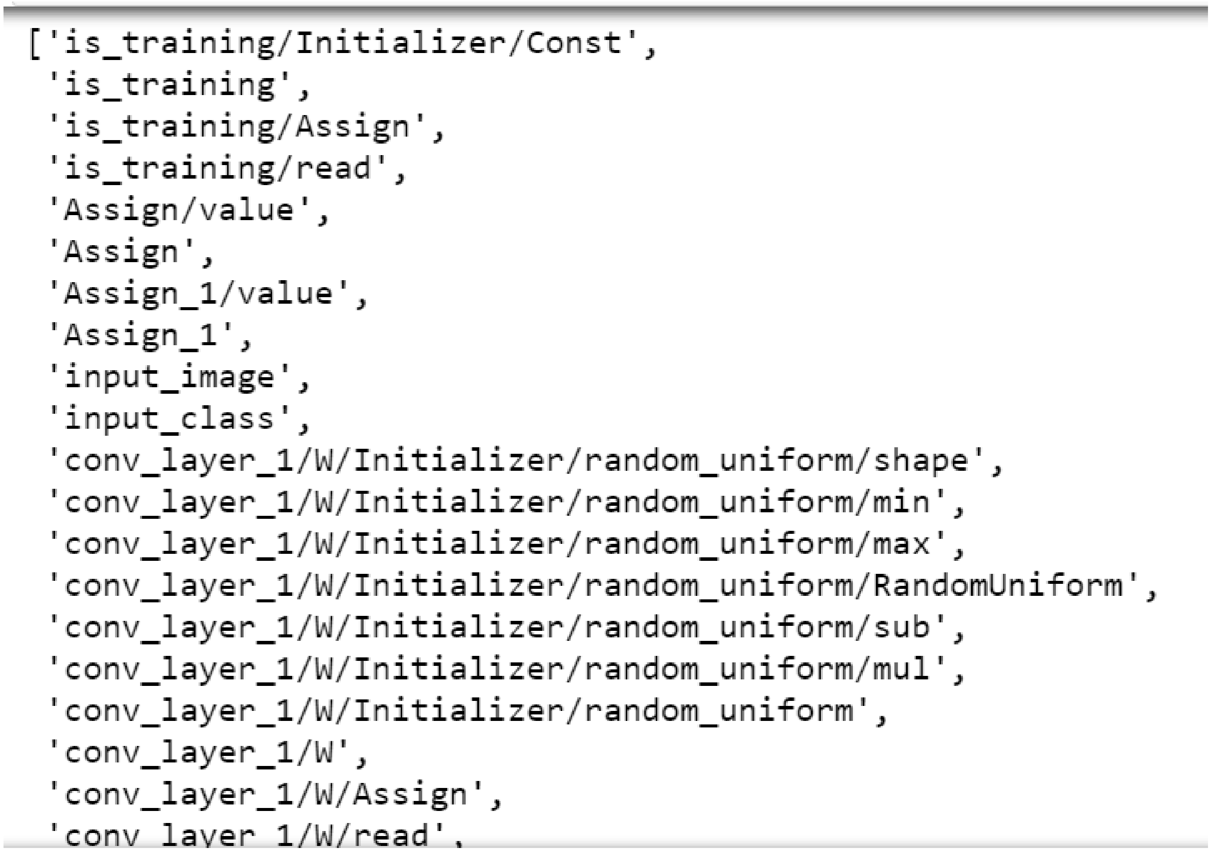
5.6 Loss Function and Optimizer:

Loss Function:
A loss function represents the price paid for the wrong or inaccurate predictions in classification problems.
The Loss function used is the Maximum Likelihood Loss for Classification
The model produces a vector of class probabilities, so for class c
the likelihood is the cth component of
f: P[c|f(x;w)]
known as the **Cross-Entropy.**

Optimizer: Adam Optimizer
Adam optimizer is the go to optimizer in case of Convolutional Networks because of its little memory requirement and suitability for the large data and parameters.

6 EVALUATION:

Before evaluating our model, the session of the Tensor flow is saved in the local machine initializing all the variables as it takes a lot of time to learn through the layers.
Once the Model is saved, all the operations of the Graph can be displayed by using the **tf.get_default_graph()** function.



```
['is_training/Initializer/Const',
 'is_training',
 'is_training/Assign',
 'is_training/read',
 'Assign/value',
 'Assign',
 'Assign_1/value',
 'Assign_1',
 'input_image',
 'input_class',
 'conv_layer_1/W/Initializer/random_uniform/shape',
 'conv_layer_1/W/Initializer/random_uniform/min',
 'conv_layer_1/W/Initializer/random_uniform/max',
 'conv_layer_1/W/Initializer/random_uniform/RandomUniform',
 'conv_layer_1/W/Initializer/random_uniform/sub',
 'conv_layer_1/W/Initializer/random_uniform/mul',
 'conv_layer_1/W/Initializer/random_uniform',
 'conv_layer_1/W',
 'conv_layer_1/W/Assign',
 'conv_layer_1/W/read',
```

6.1 Testing Data: 5000 Images

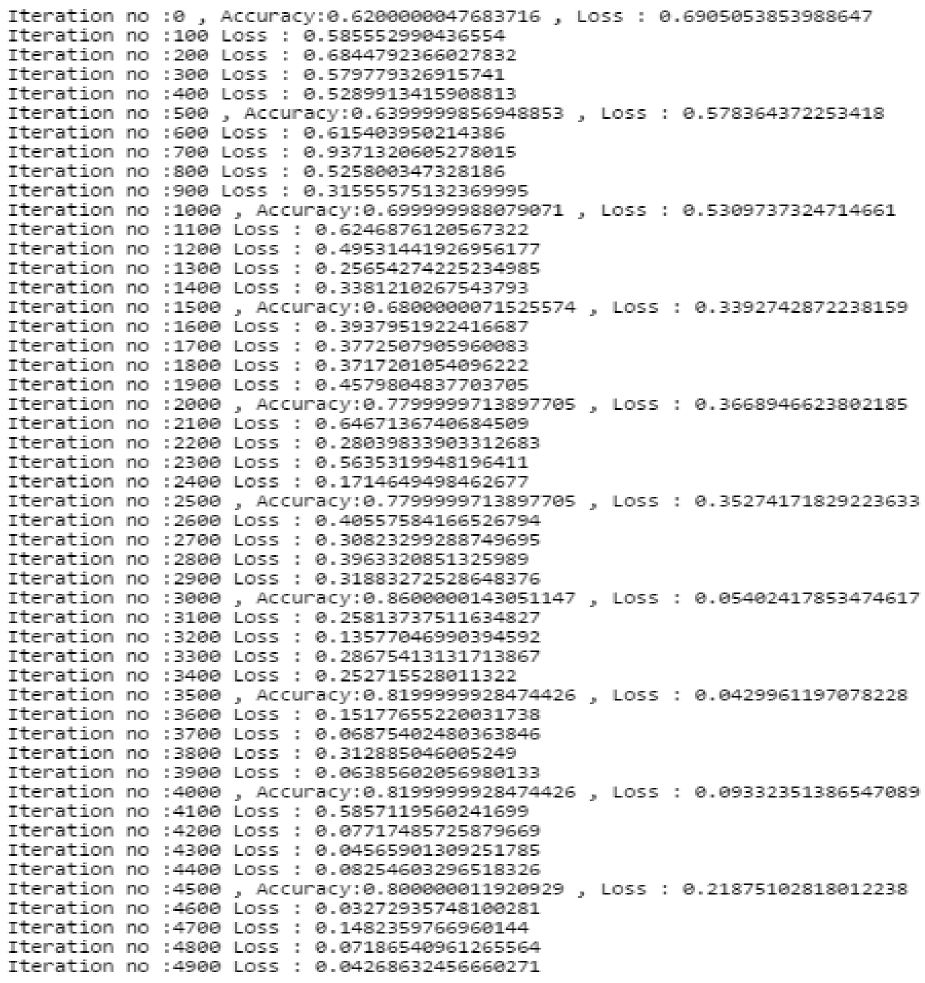
For the Evaluation of the Model (Algorithm), the Model is now fed with the Testing Data to check the accuracy of the Model.

For this task first the Epoch size and the Batch size are set:
The Model is not fed with the whole Testing data at once rather it is done in terms of small batches known as EPOCHS, it can take thousands of epochs for your backpropagation algorithm to converge on a combination of weights with an acceptable level of accuracy.

- Epochs = 5000 (This can be changed according to the hardware)
- Batch_size = 20 (For the GPU it is in power of 2 and can be changed)

Now the OPTIMIZATION of the algorithm is done by RESHAPING the inputs of the

Training data and the Accuracy is printed after every 500 iterations to check if the Accuracy is Increasing or not.
Also, the Loss is printed after every 100 Iterations:

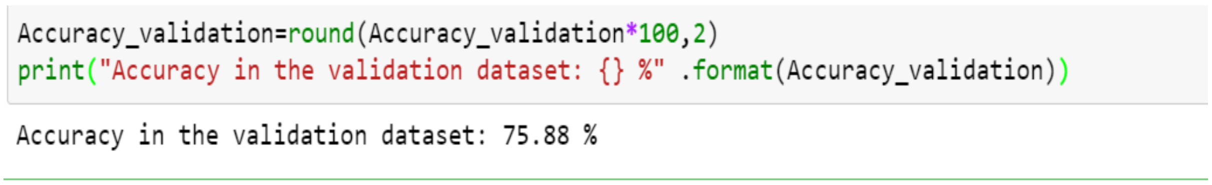


```
Iteration no :0 , Accuracy:0.6200000047683716 , Loss : 0.6905053853988647
Iteration no :100 Loss : 0.585552990436554
Iteration no :200 Loss : 0.6844792366027832
Iteration no :300 Loss : 0.579779326915741
Iteration no :400 Loss : 0.5289913415908813
Iteration no :500 , Accuracy:0.6399999856948853 , Loss : 0.578364372253418
Iteration no :600 Loss : 0.615403950214386
Iteration no :700 Loss : 0.9371320605278015
Iteration no :800 Loss : 0.525800347328186
Iteration no :900 Loss : 0.31555575132369995
Iteration no :1000 , Accuracy:0.699999988079071 , Loss : 0.5309737324714661
Iteration no :1100 Loss : 0.6246876120567322
Iteration no :1200 Loss : 0.49531441926956177
Iteration no :1300 Loss : 0.25654274225234985
Iteration no :1400 Loss : 0.3381210267543793
Iteration no :1500 , Accuracy:0.6800000071525574 , Loss : 0.3392742872238159
Iteration no :1600 Loss : 0.3937951922416687
Iteration no :1700 Loss : 0.3772507905960083
Iteration no :1800 Loss : 0.3717201054096222
Iteration no :1900 Loss : 0.4579804837703705
Iteration no :2000 , Accuracy:0.7799999713897705 , Loss : 0.3668946623802185
Iteration no :2100 Loss : 0.6467136740684509
Iteration no :2200 Loss : 0.28039833903312683
Iteration no :2300 Loss : 0.5635319948196411
Iteration no :2400 Loss : 0.1714649498462677
Iteration no :2500 , Accuracy:0.7799999713897705 , Loss : 0.35274171829223633
Iteration no :2600 Loss : 0.40557584166526794
Iteration no :2700 Loss : 0.30823299288749695
Iteration no :2800 Loss : 0.3963320851325989
Iteration no :2900 Loss : 0.31883272528648376
Iteration no :3000 , Accuracy:0.8600000143051147 , Loss : 0.05402417853474617
Iteration no :3100 Loss : 0.25813737511634827
Iteration no :3200 Loss : 0.13577046990394592
Iteration no :3300 Loss : 0.28675413131713867
Iteration no :3400 Loss : 0.252715528011322
Iteration no :3500 , Accuracy:0.8199999928474426 , Loss : 0.0429961197078228
Iteration no :3600 Loss : 0.15177655220031738
Iteration no :3700 Loss : 0.06875402480363846
Iteration no :3800 Loss : 0.312885046005249
Iteration no :3900 Loss : 0.06385602056980133
Iteration no :4000 , Accuracy:0.8199999928474426 , Loss : 0.09332351386547089
Iteration no :4100 Loss : 0.5857119560241699
Iteration no :4200 Loss : 0.07717485725879669
Iteration no :4300 Loss : 0.04565901309251785
Iteration no :4400 Loss : 0.08254603296518326
Iteration no :4500 , Accuracy:0.800000011920929 , Loss : 0.21875102818012238
Iteration no :4600 Loss : 0.03272935748100281
Iteration no :4700 Loss : 0.1482359766960144
Iteration no :4800 Loss : 0.07186540961265564
Iteration no :4900 Loss : 0.04268632456660271
```

The above image shows that the Model has reached the accuracy of nearly **80%** with the Loss Minimized to **0.21875**

6.2 Validation Data: 2500 Images

Again, the Optimization of the Validation Images is done reshaping the Images to 56*56*3 , Labelling the Inputs to 2 Values(cats and dogs) and then feeding the data to the model.
Accuracy of the Validation Data is: **75.88%**



```
Accuracy_validation=round(Accuracy_validation*100,2)
print("Accuracy in the validation dataset: {} %" .format(Accuracy_validation))
```

```
Accuracy in the validation dataset: 75.88 %
```

7 DEPLOYMENT:

Finally, the Model is deployed by creating a function, to test the Test Data with 12500 Images which predicts if the input image is a CAT or a DOG.
If the Class predicted is **1**: The Image is of a **Dog**
If the Class predicted is **0**: The image is of a **Cat.**

8 CONCLUSION:

In this report, the whole Development process the Machine learning project was discussed. First the familiarity of the Image Recognition is provided in the Deep Learning and Computer Vision fields, then the Drawbacks of the traditional approaches were highlighted. Furthermore, the use of Convolutional neural Networks when classifying the images is shown. Subsequently, The Model architecture and the hyperparameters were depicted which are used in this model. The Accuracy of both the Training and the Test data is recorded and the model is deployed successfully. This Project work on the Cat vs Dog dataset is the first step towards the more complex Face Tagging and Computer Vision techniques.

9 REFERENCES:

- https://www.analyticsvidhya.com/blog/2016/04/deep-learning-computer-vision-introduction-convolution-neural-networks/
- http://homepages.inf.ed.ac.uk/rbf/HIPR2/classify.htm
- Datacamp – Deep learning with Python
- http://www.subsubroutine.com/sub-subroutine/2016/9/30/cats-and-dogs-and-convolutional-neural-networks
- https://medium.com/@ageitgey/machine-learning-is-fun-part-3-deep-learning-and-convolutional-neural-networks-f40359318721