Department of Information Technology

INDUSTRIAL TRAINING REPORT ON

IMAGE CLASSIFICATION FOR DOGS AND CATS USING CNN

Submitted By:

Manjeet Singh (1609113061)

JSS Academy of Technical Education, NOIDA

Dr. APJ Abdul Kalam Technical University, Lucknow, U.P

2019 - 20

DECLARATION

I the undersigned solemnly declare that the project report IMAGE CLASSIFICATION FOR DOGS AND CATS USING CNN is based on my own work carried out during the course of study under the supervision of **Mr. Vineet Pathak.**

I assert the statements made and conclusions drawn are an outcome of my research work. I further certify that -

I. The work contained in the report is original and has been done by me under the general supervision of my supervisor.
II. The work has not been submitted to any other Institution for any other degree/diploma/certificate in this university or any other University of India or abroad.
III. I have followed the guidelines provided by the university in writing the report.
IV. Whenever I have used materials (data, theoretical analysis, and text) from other sources, I have given due credit to them in the text of the report and giving their details in the references.

Manjeet Singh
(1609113061)

ACKNOWLEDGEMENT

The euphoria that accompany the successful completion of this project would be incomplete without the mention of the people who made it possible.

I would like to take this opportunity to thank my corporate mentor Mr. Vineet Pathak who in spite of busy schedule has cooperated with me and motivated me continuously. His positive and supportive attitude and continuous encouragement has made it possible for me to complete this project.

I am greatly indebted to him for providing their valuable guidance at all stages of the study. His constructive suggestions and contributions have been certainly indispensable for my project work

I would like to express my gratitude towards Mr. Atul Gupta, Training Facilitator, who made the process of working on the project well ordered with his guidance and support.

I hope that I can build upon the experience and knowledge that I have gained and make a valuable contribution towards the industry in the coming future.

ABSTRACT

In this project, I have developed a machine learning model to predict images of dogs and cats, which is the Dogs vs. Cats competition from Kaggle. I used Deep Convolutional Neural Networks (CNN) which is implemented using Keras, to learn features of images and classify them accordingly. I tried various experiments to improve the performance on the test dataset, and finally got the best accuracy of 87.20%.

For this project, I have used Wikipedia comment dataset available on Microsoft (https://www.microsoft.com/en-us/download/confirmation.aspx?id=54765)

The training archive contains 25,000 images of dogs and cats. My aim is to make the model learn the distinguishing features between the cat and dog. Once the model has learned, i.e once the model got trained, it will be able to classify the input image as either a cat or a dog.

Every image is actually a set of pixels. I converted all those pixels into an array because raw images can't be fed right through a convolutional neural network. I also converted image from RGB to grayscale. Then, I resized the image array because we have got here images of various sizes. All these operations are done using cv2. Finally, I converted the labels "dog" and "cat" to 0 and 1.

After, The dataset is preprocessed I made convolutional neural network . Firstly, I normalized the features array. Then, I made a sequential model and added three Conv2D layers to it. At the end for getting the output I added a final dense layer.Then I compiled the model.

Once the model is built, I trained it by fitting model with training data.

In the end I tested the model by giving it an image which was not in the training dataset.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 ABOUT MCN SOLUTIONS

MCN Solutions is a fifteen-year-old custom software development and outsourcing services provider. We specializes in mobile app and web development for small to mid-sized businesses. Headquartered in Delhi national capital region of India, MCN has its extended offices in USA, UK and Poland.

Whether you are a start-up or a conventional business, we will be thrilled to assist you in every phase of the software development life cycle: from business analysis, designing, and prototyping, to the development and deployment of a complete solution and testing of products.

MCN Solutions is a matchless creation of software professionals commencing several domains, with robust management, industrial experience, and a wide-ranging technical skill set. Our team is committed to delivering quality products to our customers on time.

1.2 MOTIVATION

Computers cannot "see" images like humans do. The computer reads two numeric matrices lacking semantic meaning. However, within the last decade, convolutional neural networks have demonstrated drastic improvement in image classification.
While image classification has many uses, I am particularly motivated by the application in web accessibility. For users with visual impairments, a screen reader reads aloud the alt text, which has to be manually entered. However, manually tagging images is inefficient and prone to error (i.e., skipping an image). To increase efficiency and accessibility for users, screen reader technologies should automatically detect the image's content.

The Dogs vs. Cats competition from Kaggle is trying to solve the CAPTCHA challenge, which relies on the problem of distinguishing images of dogs and cats. It is easy for humans, but evidence suggests that cats and dogs are particularly difficult to tell apart automatically. Many people have worked or are working on constructing machine learning classifiers to address this problem. A classifier based on color features got 56.9% accuracy on the Asirra dataset. An accuracy of 82.7% was achieved from an SVM classifier based on a combination of color and texture features.

In this project, I also would like to solve this problem and achieve higher performance. For instance, I tried to build a model based on convolutional neural networks (CNN) to classify the images. After testing for various nodes and Conv2D layer combinations finally achieved our best classification accuracy of 87.20% with a sequential model having 3 Conv2D layers with 32 nodes and using "relu" activation function and "adam" optimizer.

1.3 SCOPE

Today machine learning has become a driving force behind technological advancements used by people on a daily basis. Image recognition is one of the most accessible applications of it, and it's fueling a visual revolution online. This project gives a general idea of how image classification can be done efficiently.

Own image can be tested to verify the accuracy of the model.

This code can directly be can be extended as a mobile application or a site.

To extend the project to classify different entities, change the dataset accordingly and train the model.

The scope of the project can be extended to the various industries where there is a huge scope for automation, by just altering the dataset which is relevant to the problem.

The application of the project can be extended to a personal photo organization. The image recognition API integrated in the apps can categorizes images on the basis of identified patterns and groups them thematically.

CHAPTER 2
PROBLEMS AND TECHNOLOGY USED

2.1 INTRODUCTION TO THE PROBLEM STATEMENT

Our basic task is to create an algorithm to classify whether an image contains a dog or a cat. The image input given to the system will be analyzed and the predicted result will be given as output. Machine learning algorithm [Convolutional Neural Networks] is used to classify the image.

The input for this task is images of dogs or cats from training dataset, while the output is the classification accuracy on test dataset.

The dataset for this project is provided by Microsoft Research. The dataset contains 25,000 images, including 12,500 images of dogs and 12,500 images of cats.The average size for these images is around 350×500.

Our learning task is to learn a classification model to determine the decision boundary for the training dataset.

Our performance task is to apply the learned classification model to classify images from the test dataset, and then evaluate the classification accuracy.

2.2 CONVOLUTIONAL NEURAL NETWORK (CNN)

CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons are fully connected networks where each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. However, CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme.

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

CNNs use relatively little pre-processing compared to other image classification algorithms. They have applications in image and video recognition, recommender systems, image classification, medical image analysis, and natural language processing.

The basic CNN structure is as follows:
Convolution -> Pooling -> Convolution -> Pooling -> Fully Connected Layer -> Output

Convolution is the act of taking the original data, and creating feature maps from it. Pooling is down-sampling, most often in the form of "max-pooling," where we select a region, and then take the maximum value in that region, and that becomes the new value for the entire region. Fully Connected Layers are typical neural networks, where all nodes are "fully connected."

2.3. LAYERS IN CONVOLUTIONAL NEURAL NETWORK(CNN)

We use three main types of layers to build ConvNet architectures:

2.3.1 Convolutional Layer

Convolutional layer is the very first layer where we extract features from the images in our datasets. Due to the fact that pixels are only related with the adjacent and close pixels, convolution allows us to preserve the relationship between different parts of an image.

Convolution is basically filtering the image with a smaller pixel filter to decrease the size of the image without losing the relationship between pixels.

When we apply convolution to 5x5 image by using a 3x3 filter with 1x1 stride (1 pixel shift at each step). We will end up having a 3x3 output (64% decrease in complexity).

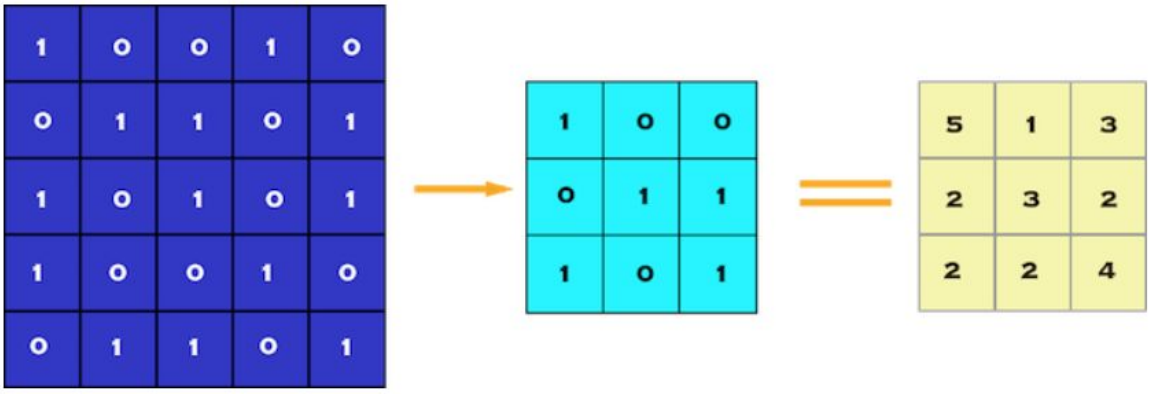

Figure 1: Convolution of 5 x 5 pixel image with 3 x 3 pixel filter (stride = 1 x 1 pixel)

Figure 2.1

2.3.2 Pooling Layer

When constructing CNNs, it is common to insert pooling layers after each convolution layer to reduce the spatial size of the representation to reduce the parameter counts which reduces the computational complexity.

In addition, pooling layers also helps with the overfitting problem. Basically we select a pooling size to reduce the amount of the parameters by selecting the maximum, average, or sum values inside these pixels.

The most common form of pooling is Max pooling where we take a filter of size F*F and apply the maximum operation over the F*F sized part of the image.

If we take the average in place of taking maximum, it will be called average pooling, but it's not very popular.

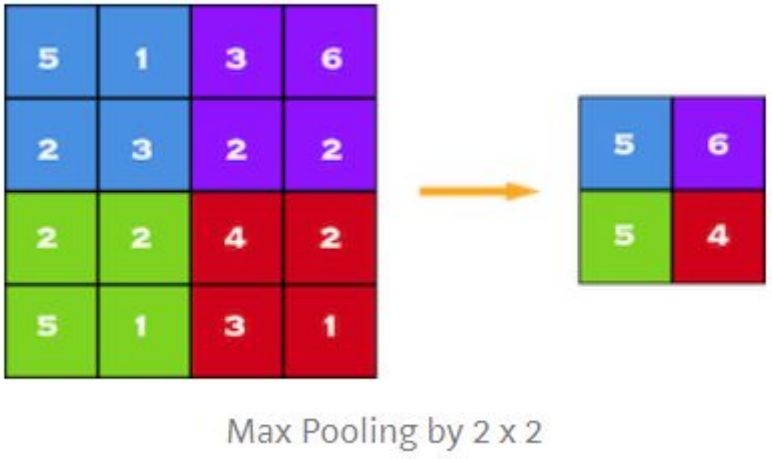

Figure 2.2

2.3.3 Fully-Connected Layers

A fully connected network is our RegularNet where each parameter is linked to one another to determine the true relation and effect of each parameter on the labels. Since our time-space complexity is vastly reduced thanks to convolution and pooling layers, we can construct a fully connected network in the end to classify our images.

If each neuron in a layer receives input from all the neurons in the previous layer, then this layer is called fully connected layer. The output of this layer is computed by matrix multiplication followed by bias offset.

A set of fully connected layers looks like this:

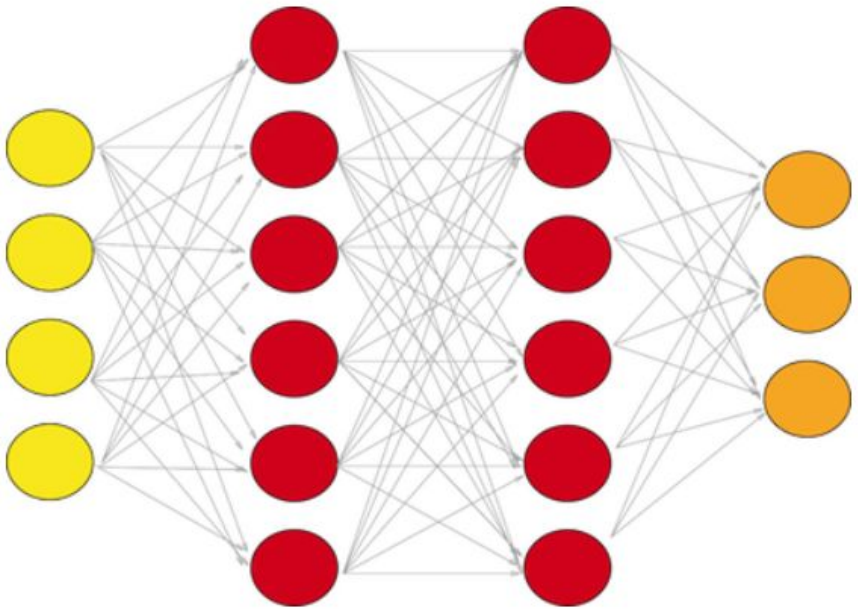

Figure 2.3

2.4 CONCEPTUAL FRAMEWORK

The project is entirely implemented using Python3. The Conceptual Framework involved is mainly:

2.4.1 Keras – Tensorflow backend

Keras is a model-level library, providing high-level building blocks for developing deep learning models. It does not handle low-level operations such as tensor products, convolutions and so on itself. Instead, it relies on a specialized, well optimized tensor manipulation library to do so, serving as the "backend engine" of Keras. Rather than picking one single tensor library and making the implementation of Keras tied to that library, Keras handles the problem in a modular way, and several different backend engines can be plugged seamlessly into Keras.

At this time, Keras has three backend implementations available: the TensorFlow backend, the Theano backend, and the CNTK backend.

- **TensorFlow** is an open-source symbolic tensor manipulation framework developed by Google.

- **Theano** is an open-source symbolic tensor manipulation framework developed by LISA Lab at Université de Montréal.

- **CNTK** is an open-source toolkit for deep learning developed by Microsoft.

Here, Keras is used to implement the CNN.

2.4.2 OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.The library is used extensively in companies, research groups and by governmental bodies.

OpenCV's deployed uses span the range from stitching street view images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

Here, OpenCV is used to handle image operations like reading the image, resizing, reshaping.

2.5 DATA STRUCTURES AND ALGORITHMS USED

2.5.1 Numpy Array

Numpy arrays are great alternatives to Python Lists. Some of the key advantages of Numpy arrays are that they are fast, easy to work with, and give users the opportunity to perform calculations across entire arrays.

Numpy Array is the most powerful and widely used data structure of python here it is used to store the pixel value of images.

Data manipulation in Python is nearly synonymous with NumPy array manipulation even newer tools like Pandas are built around the NumPy array.

A few categories of basic array manipulations here:

- **Attributes of arrays :** Determining the size, shape, memory consumption, and data types of arrays
- **Indexing of arrays :** Getting and setting the value of individual array elements
- **Slicing of arrays :** Getting and setting smaller subarrays within a larger array
- **Reshaping of arrays :** Changing the shape of a given array
- **Joining and splitting of arrays :** Combining multiple arrays into one, and splitting one array into many

Here, Image that is read will be stored in a numpy array.

2.6 TOOLS USED

2.6.1 Python 3.7

Python 3.7 is now officially released. This new Python version has been in development since September 2016, and now we all get to enjoy the results of the core developers' hard work.

There are many new features included in the new version. Some of the new features include:

- Easier access to debuggers through a new breakpoint() built-in
- Simple class creation using data classes
- Customized access to module attributes
- Improved support for type hinting
- Higher precision timing functions
- More importantly, Python 3.7 is fast.

2.6.2 Anaconda Prompt

Anaconda command prompt is just like command prompt, but it makes sure that you are able to use anaconda and conda commands from the prompt, without having to change directories or your path.

When we start Anaconda command prompt, it adds/("prepends") a bunch of locations to your PATH. These locations contain commands and scripts that we can run. So as long as we are in the Anaconda command prompt, you know you can use these commands.

2.6.3 Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

A Jupyter Notebook document is a JSON document, following a versioned schema, and containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media, usually ending with ".ipynb" extension.

A Jupyter Notebook can be converted to a number of open standard output formats (HTML, presentation slides, LaTeX, PDF, ReStructuredText, Markdown, Python).

Jupyter Notebook provides a browser-based REPL built upon a number of popular open-source libraries:

- IPython
- ØMQ
- Tornado (web server)
- jQuery
- Bootstrap (front-end framework)
- MathJax

Jupyter Notebook can connect to many kernels to allow programming in many languages. By default Jupyter Notebook ships with the IPython kernel. As of the 2.3 release (October 2014), there are currently 49 Jupyter-compatible kernels for many programming languages, including Python, R, Julia and Haskell.

2.7 TRAINING AND TESTING DATASET

The dataset for this project is provided by Microsoft Research. The dataset contains 25,000 images, including 12,500 images of dogs and 12,500 images of cats.The average size for these images is around 350×500.

The images are of non-uniform size and varied image quality. Some of them are grayscale, and some files are corrupted, making them unreadable. Additionally, the perspective can vary from headshots to full-body.

In some images, part of the animal is obstructed from view, and others contain more than one of the same animals.

Dataset : Dogs vs Cats
Provider : Microsoft Research
Description : Binary classification -- Classify dogs and cats.
Training : 19790 images (9895 per class)
Validation : 4948 images (2474 per class)
Testing : 1500 unlabeled images (Test any image which is not in training dataset)
File Format : All the image files are in ".jpg" format.

2.8 DATA PREPROCESSING

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm.

Data preprocessing is a technique, which is used to transform the raw data in a useful and efficient dataset, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. Thus, data preprocessing is applied on the data.

The images in the dataset are of non-uniform size and varied image quality. Some of them are grayscale, and some files are corrupted, making them unreadable. These raw images can't be fed right through a convolutional neural network.

So I used cv2 library to read our image into an array and imread() function in the library to read as a grayscale image to reduce the complexity.

I resized the image array using resize() function of cv2 library because we have got here images of various sizes.

Finally, I converted the labels of "dog" and "cat" to 0 and1 because we cannot do our operations on string.

CHAPTER 3
IMPLEMENTATION AND RESULT

3.1 IMPLEMENTATION OF PROJECT

1. First, we do data pre-processing.
2. Then, we define the model. Here are the steps to do define our CNN model:
    2.1. Define a Sequential model.
    2.2. Start adding layers to it.
    2.3. First we will add a Conv2D layer with 32 nodes and kernel size of (3,3).
    2.4. We have to specify input shape which is your X shape.
    2.5. For Activation we will take “relu” Activation Function.
    2.6. Now after every Conv layer we always do max pooling so we will add max pooling layer with a size of (2,2).
    2.7. We add 3 convolution layers.
    2.8. Then, we will add a flatten layer. As we have to feed our data to Dense layer later.
    2.9. In the end for getting our result we will add final Dense layer .
    2.10. For Activation I used “sigmoid” function.
    2.11. Finally we will compile the model .
    2.12. To make our model better we either minimize loss or maximize accuracy. NN always minimize loss. To measure it I have used “binary_crossentropy”.
    2.13. To minimize cost function we use optimizers. I used “adam” optimizer.
    2.14. Metrics is to denote the measure of the model. My metric is “accuracy”.

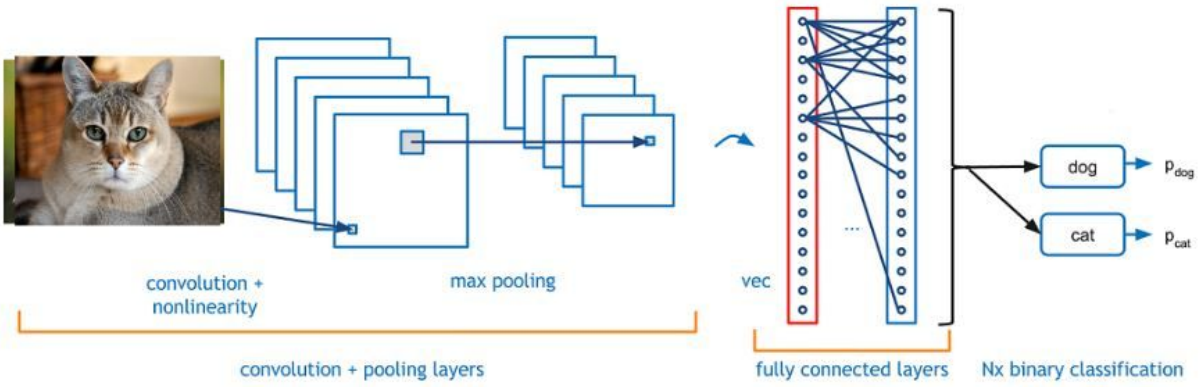

Figure 3.1(CNN Overview)

3. Then we fit our model with training data.

```
Train on 19790 samples, validate on 4948 samples
Epoch 1/10
19790/19790 - 151s - loss: 0.6206 - acc: 0.6423 - val_loss: 0.5440 - val_acc: 0.7381
Epoch 2/10
19790/19790 - 155s - loss: 0.5147 - acc: 0.7477 - val_loss: 0.5055 - val_acc: 0.7563
Epoch 3/10
19790/19790 - 156s - loss: 0.4655 - acc: 0.7780 - val_loss: 0.4683 - val_acc: 0.7848
Epoch 4/10
19790/19790 - 155s - loss: 0.4283 - acc: 0.8045 - val_loss: 0.4529 - val_acc: 0.7945
Epoch 5/10
19790/19790 - 170s - loss: 0.3968 - acc: 0.8215 - val_loss: 0.4052 - val_acc: 0.8165
Epoch 6/10
19790/19790 - 164s - loss: 0.3744 - acc: 0.8344 - val_loss: 0.4541 - val_acc: 0.7840
Epoch 7/10
19790/19790 - 156s - loss: 0.3530 - acc: 0.8446 - val_loss: 0.4681 - val_acc: 0.7811
Epoch 8/10
19790/19790 - 156s - loss: 0.3371 - acc: 0.8505 - val_loss: 0.3933 - val_acc: 0.8270
Epoch 9/10
19790/19790 - 157s - loss: 0.3155 - acc: 0.8615 - val_loss: 0.3698 - val_acc: 0.8399
Epoch 10/10
19790/19790 - 156s - loss: 0.2984 - acc: 0.8720 - val_loss: 0.3734 - val_acc: 0.8405
```

Figure 3.2(Training Screenshot)

4. Finally we feed your model with test data to predict. We use a dataset to which was not used in training so that we get the true accuracy.

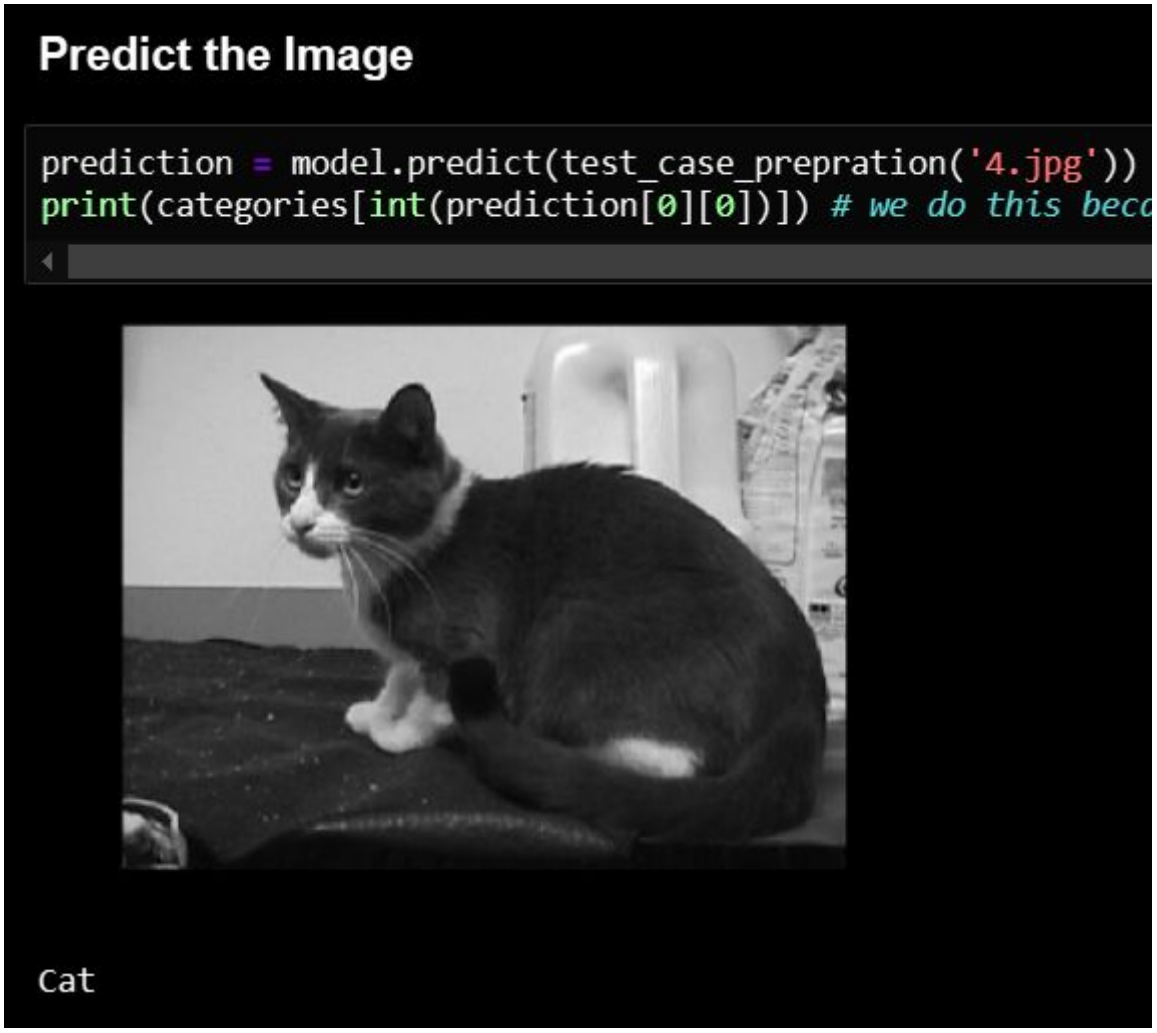

Figure 3.3(Prediction Screenshot)

3.2.RESULT AND ANALYSIS

Output contains the probabilities of the input image being a dog or a cat.

I am able to achieve a training accuracy of 87.20.% and a validation accuracy of 84.05% using CNN Algorithm.

The algorithm is also performing very good on the unseen testing image dataset.

After testing for various nodes and Conv2D layer combinations we finally achieved our best classification accuracy with a sequential model having three Conv2D layers with 32 nodes and using “relu” activation function and “adam” optimizer.

1. **Model Defining Parameters:**
    a) Type of Model = “sequential”
    b) No. of Conv2D layers = 3
    c) No. of Dense layers = 0
    d) No. nodes of Conv2D layers = 32
    e) Kernel Size = 3x3
    f) Pool Size = 2x2
    g) Activation Function of Conv2D layers = “relu”
    h) Activation Function of Output layer = “sigmoid”

2. Model Compiling parameters :

a) Optimizer = “adam”
b) Loss Function = “binary_crossentropy”
c) Metrics = “accuracy”

3. Model Training parameters :

a) Epochs = 10

b) Validation split = 0.2

c) Batch size = 32

CHAPTER 4
CONCLUSION AND FUTURE SCOPE

I've learned the underlying concepts of how computers see (Image Classification) by implementing simple yet very powerful image classification system.

I used Deep Convolutional Neural Networks (CNN) which is implemented using Keras, to learn features of images and classify them accordingly.

The CNN Algorithm outperforms the Logistic Regression Model by a huge margin. Thus, proving CNN's algorithms give a much better performance on Computer Vision tasks as compared to other machine learning algorithms.

I tried various experiments to improve the performance on the test dataset, and finally I was able to achieve a training accuracy of 87.20.% and a validation accuracy of 84.05% using CNN Algorithm.

There is vast scope of improvement for the model by using data augmentation.

In the future, I will explore more to achieve better performance. For instance, I will try to change the architecture and parameter settings of the Deep Neural Network based on the feature visualization of different layers' feature maps. Also, I will apply different parameter settings for SVMs and Deep Neural Networks. I may also try object localization to eliminate the influence of complicated backgrounds. Additionally, I would like to extract more features or try a combination of human-crafted features and learned features.

Possibilities of improvement in this field and Computer Vision in general are boundless.

The extension of the project can be used in the field of stock photography and video. Stock websites provide platforms where photographers and video makers can sell their content. Contributors need a way to tag large amounts of visual material, which is time-consuming and tedious. Image recognition provides the tools to make visual content discoverable by users via search. At the same time, is a huge relief for stock contributors. They get automatic keyword suggestions, which save them a ton of time and efforts.

The large Scale applications for the project can be done in the field of social intelligence. It involves following conversations on social media to learn more about prospects. In a sea of abundant and often irrelevant visual content, extracting useful information is possible only through machine learning – or 'visual listening.' For example, image recognition can identify visual brand mentions and expression of emotion towards a brand, as well as logo and other brand data that would be otherwise undiscoverable. On the basis of collected information from analyzing images, marketers can better target their campaigns by using customization and personalization.

REFERENCES

[1] Rajat Garg, ”Kaggle “Dogs vs. Cats” Challenge”,Medium, Feb 8, 2019
https://medium.com/@mrgarg.rajat/kaggle-dogs-vs-cats-challenge-complete-step-by-step-guide-part-2-e9ee4967b9

[2] Anchit Jain , “Creating a simple dog versus cat image classifier using Keras”, Medium, May 26, 2018
https://medium.com/@jayeshbahire/introduction-to-word-vectors-ea1d4e4b84bf

[3] Greg Surma , “Image Classifier - Cats vs Dogs”, Towards Data Science , Nov 19, 2018
https://towardsdatascience.com/image-classifier-cats-vs-dogs-with-convolutional-neural-networks-cnns-and-google-colabs-4e9af21ae7a8