[

**Image Classification Using CNN**

A thesis submitted in partial fulfillment of the requirements for

The award of the degree of

B. Tech.

In

Computer Science and Engineering

By

Satya Vishalakshi Paluri (Roll No. 17221A0585)

Bibek Lamsal (Roll No. 17221A05A7)

` Rajiv Gopal Singh (Roll No. 17221A05B0)



COMPUTER SCIENCE AND ENGINEERING BONAM VENKATA CHALAMAYYA ENGINEERING COLLEGE ODALAREVU AP 533210

July 2021

In memory of

Late Mr. Bonam Venkata Chalamayya “Founder Chairman” BVCE

Our deepest gratitude.

BONAFIDE CERTIFICATE

This is to certify that the project titled IMAGE CLASSIFICATION USING CNN is a bonafide record of the work done by

Satya Vishalakshi Paluri (Roll No. 17221A0585)

Bibek Lamsal (Roll No. 17221A05A7)

Rajiv Gopal Singh (Roll No. 17221A05B0)

in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering of the BONAM VENKATA CHALAMAYYA ENGINEERING COLLEGE, AUTONOMOUS, ODALAREVU, during the year 2020-21.



**B.V. M Subash** **Prof Gunamani Jena, ME PhD**

Guide Head of the Department

Project Viva-voce held on

Internal Examiner External Examiner

**ACKNOWLEDGEMENTS**

I would like to express our deepest gratitude to the following people for guiding us through this course and without whom this project and the results achieved from it would not have reached completion.

B Sireesha M Subash, Department of Computer Science and Engineering, for helping us and guiding us in the course of this project. Without his guidance, I would not have been able to successfully complete this project. His patience and genial attitude is and always will be a source of inspiration to me.

Prof Gunamani Jena, ME PhD, the Head of the Department, Department of Computer Science and Engineering, for allowing me to avail the facilities at the department.

I am thankful to the Principal for providing the facility in this institution.

I am also thankful to the Honorable Chairman Dr Bonam Rajesh for motivating, arranging training and internship related to the Project.

My sincere thanks to the Honorable Secretary Mr. Bonam Kanakayya for extending the excellent infrastructure.

I am also thankful to the faculty and staff members of the Department of Computer Science and Engineering, my parents and my friends for their constant support and help.

# Table of Contents

[Title 8](#_Toc77338290)

[TABLE OF CONTENTS 8](#_Toc77338291)

[CHAPTER 1 1](#_Toc77338292)

[INTRODUCTION 1](#_Toc77338293)

[1.1 Introduction 1](#_Toc77338294)

[1.2 Machine Learning 2](#_Toc77338295)

[1.3 Neural network 2](#_Toc77338296)

[1.4 Deep Learning 5](#_Toc77338297)

[1.5 Deep Learning Frameworks 8](#_Toc77338298)

[1.6 Convolutional neural networks and image classification 8](#_Toc77338299)

[1.7 PYTHON 11](#_Toc77338300)

[CHAPTER 2 12](#_Toc77338301)

[REQUIREMENT ANALYSIS 12](#_Toc77338302)

[3.1 Project Overview: - 12](#_Toc77338303)

[3.2 Functional requirements: - 13](#_Toc77338304)

[3.3 Non –Functional Requirements: - 13](#_Toc77338305)

[3.4 Hardware Requirements: - 13](#_Toc77338306)

[3.5 Software Requirements: - 14](#_Toc77338307)

[Technologies and Languages used to Develop 24](#_Toc77338308)

[CHAPTER 5 27](#_Toc77338309)

[SYSTEM STUDY 27](#_Toc77338310)

[ECONOMICAL FEASIBILITY 27](#_Toc77338311)

[TECHNICAL FEASIBILITY 27](#_Toc77338312)

[SOCIAL FEASIBILITY 27](#_Toc77338313)

[Chapter 4 28](#_Toc77338314)

[Methodology: 28](#_Toc77338315)

[4.1 CONVOLUTION NEURAL NETWORK: 28](#_Toc77338316)

[4.2 DATASET: 30](#_Toc77338317)

[4.3 INPUT DESIGN AND OUTPUT DESIGN 31](#_Toc77338318)

[CHAPTER 4 33](#_Toc77338319)

[ANALYSIS AND DESIGN PHASE 33](#_Toc77338320)

[4.1 INTRODUCTION 33](#_Toc77338321)

[4.2 USE CASE DIAGRAM 33](#_Toc77338322)

[4.3 CLASS DIAGRAM 34](#_Toc77338323)

[4.4 SEQUENCE DIAGRAM 35](#_Toc77338324)

[4.4 ACTIVITY DIAGRAM 36](#_Toc77338325)

[4.5 Object Diagram 37](#_Toc77338326)

[4.6 Component Diagram 38](#_Toc77338327)

[4.7 Deployment Diagram 39](#_Toc77338328)

[CHAPTER 5 40](#_Toc77338329)

[SYSTEM LOW LEVEL DESIGN 40](#_Toc77338330)

[5.1 Model Construction Steps: 40](#_Toc77338331)

[5.2 Application of Built in Modules Data Structure: 41](#_Toc77338332)

[CHAPTER 6 42](#_Toc77338333)

[IMPLEMENTATION 42](#_Toc77338334)

[6.1 Sample Code 42](#_Toc77338335)

[6.2 TENSORFLOW CNN 46](#_Toc77338336)

[6.3 Screen Captures 48](#_Toc77338337)

[CHAPTER 7 54](#_Toc77338338)

[SYSTEM TEST 54](#_Toc77338339)

[7.1 Software Testing 54](#_Toc77338340)

[7.2 Black box Testing 54](#_Toc77338341)

[7.3 White box Testing 54](#_Toc77338342)

[7.4 Performance Testing 54](#_Toc77338343)

[7.5 Load Testing 54](#_Toc77338344)

[7.6 Manual Testing 54](#_Toc77338345)

[CHAPTER 8 61](#_Toc77338346)

[CONCLUSION 61](#_Toc77338347)

[8.1 Conclusion 61](#_Toc77338348)

[8.2 Scope for future work 61](#_Toc77338349)

[ Create an android/iOS app instead of website which will be more convenient to user. 61](#_Toc77338350)

[ System can be implemented using cloud which can store large amount of data for comparison and provide high computing power for processing (in case of Neural Networks) 61](#_Toc77338351)

[REFERENCE 62](#_Toc77338352)

**ABSTRACT**

This project uses the CNN algorithm. Using the MNIST dataset and CNN algorithm we are able to map raw pixels correctly to images. CNN algorithm has a prediction accuracy of up to 90%, which is better than all other algorithms such as SVM etc. The CNN network must be trained to classify all possible images, and upon upload of new images, the CNN trained model will be applied to the new image in order to predict or identify the images. The algorithm (CNN) gives reasonably good classification accuracy for all possible images

# CHAPTER 1

# INTRODUCTION

### 1.1 Introduction

Machine learning has been gaining momentum over last decades: self-driving cars, efficient web search, and speech and image recognition. The successful results gradually propagate into our daily live. Machine learning is a class of artificial intelligence methods, which allows the computer to operate in a self-learning mode, without being explicitly programmed. It is a very interesting and complex topic, which could drive the future of technology.

Image Classification

The main aim of image classification in my project is to classify a given image into one of the categories using SVM and Deep Learning and understand the difference in classification. The task of image classification is to teach the computer to recognize images and classify them into one of the trained categories. To do so, we first need to teach the computer how a cat, a dog, a bird, etc. look like before it being able to recognize a new object. The more cats the computer sees, the better it gets in recognizing cats. This is known as supervised learning. We can carry this task by inputting labelled images, the computer will start recognizing patterns present in cat pictures that are absent from other ones and will start building its own cognition. We make use of Python and TensorFlow to write the program. TensorFlow is an open source deep learning framework created by Google that gives developers granular control over each neuron so we can adjust the weights and achieve optimal performance. It is an open source library for numerical computation, specializing in machine learning applications. TensorFlow has many built-in libraries (few of which we will be using for image classification) and has an amazing community, so you’ll be able to find open source implementations for virtually any deep learning topic.. Support Vector Machine: It is a supervised learning machine algorithm, mostly used for classification problems. It is obtained by plotting each data item as a point in n-dimensional space with the value of each feature being the value of a particular coordinate. Then classification is performed by finding a hyperplane that differentiates the classes very well. Deep Learning: It is a subset of Machine Learning Algorithms that is very good at recognizing patterns but typically requires a large number of data. Deep learning excels in recognizing objects in images as its implemented using 3 or more layers of artificial neural networks where each layer is responsible for extracting one or more feature of the image.

### 1.2 Machine Learning

Machine learning as a field of research is somewhat hard to describe succinctly, due to the breadth of the field, the amount of ongoing research, the interdisciplinary aspects of the field and a multitude of other factors. For the purposes of this report the following definition will be used as a starting point, as it is sufficiently precise:

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

From a mathematical standpoint the problem can be formulated as applying a machine learning algorithm to find an unknown mathematical function with a known domain, that of the input, and a known co-domain, that of the output. In more informal terms the program is supposed to find a mathematical connection between the data and the target input. The connection depends on the task at hand. A relevant example of the aforementioned classes of tasks is to find a connection between a vectorized representation of the input to an output consisting of a set of discrete categories or a probability distribution. Concrete examples of such a task can consist of developing a program for image classification and object detection, e.g. recognizing faces in pictures and finding bounding boxes for them. Other examples of types of tasks are anomaly detection, e.g. spotting credit fraud and machine translation, e.g. translation software.

The performance of an implementation of a machine learning algorithm is usually either expressed in the accuracy of the implementation, e.g. what percentage of the examples were classified correctly, or in the error rate, e.g. what the ratio of incorrectly to correctly classified examples is. The difficult part here is not so much the choice of measure rather than which aspect(s) of the output that should be measured in the first place. Should one, for example, choose the percentage of correctly transcribed whole sequences as the only measure or take the correctly transcribed parts of sequences into account when measuring the performance? Another complicating factor is that it can be hard to measure the chosen quantity due to practical difficulties, and in those cases another measure must be chosen.

There are basically two kinds of learning, or experiences, a machine learning algorithm can have during a learning or training process; unsupervised and supervised learning. A training process is based upon a dataset consisting of a given amount of examples or data points. Unsupervised learning is based upon datasets containing features where the goal is to learn useful properties about of the structure of the dataset or, more precisely, to find the underlying probability distribution of the dataset. Supervised learning, in contrast, is based upon datasets consisting of many features but with labels attached to each example. The goal in supervised learning is to learn to predict the label, the associated value, from an arbitrary example in the dataset after the training session is over. The terms supervised and unsupervised comes from the fact that the learning algorithm has a teacher that shows the algorithm what to do in the former case, but is supposed to learn from the data without a guide in the latter case.

The most challenging part of machine learning, regardless of the algorithm(s) applied in the learning process, is to make programs that generalize well; that is, programs that perform as well or almost as well on unobserved inputs as those observed during training. The main thing that separates machine learning from optimization is that the goal consists of minimizing both the training error and the test error. The aforementioned datasets are split in two predetermined sets to be used in the training process, one for training and one for testing. The goal of machine learning can now be defined more precisely; to minimize the training error and to minimize the gap between the training error and the test error. In attempting to achieve the aforementioned goal two more challenges appear; under fitting and overfitting on the training set.

Under fitting occurs when the model cannot reach a sufficiently low training error and overfitting (see figure 2.1) occurs when the gap between the training error and the test error is too large. The former problem is caused by a model with low representational capacity that cannot fit the training set properly, the latter can be caused by a model with high capacity that memorizes properties from the training set too well to perform sufficiently on the test set. Overfitting can also be caused by a training set that is too small to generalize properly from. Under fitting can be fixed rather easily with a model with sufficient capacity, but overfitting can be harder to fix due to the fact that acquiring greater amounts of data is not always feasible. A prominent example of a technique that can be introduced during the training is regularization, which is used to limit the space of potential functions (since the goal is to find the best one). There is not much more space to go into detail here, and the problems that can occur and their corresponding solutions will be discussed in later sections. The parameters of the machine learning algorithm that are not adapted or changed by the algorithm itself during training are called hyper parameters, parameters that can be tuned to change the behavior of the learning algorithm. It is fitting to introduce a further partition of the dataset here, that of the original training data into a training set and a validation set. The training set is used purely for adjusting the internal parameters, e.g. weights and biases during the training and the validation set is used to measure the current generalization error and to adjust the hyper parameters accordingly. The validation set is not used to tune the internal parameters. The important distinction between the validation set and the test set is that the latter is not used during the training process at all and does not give any input to the training process - it is simply used to measure performance.

### 1.3 Neural network

Neural Network is a machine learning algorithm, which is built on the principle of the organization and functioning of biological neural networks. This concept arose in an attempt to simulate the processes occurring in the brain by Warren McCulloch and Walter Pitts in 1943.

Neural networks consist of individual units called **neurons**. Neurons are located in a series of groups — **layers**(see figure allow). Neurons in each layer are connected to neurons of the next layer. Data comes from the input layer to the output layer along these compounds. Each individual node performs a simple mathematical calculation. Then it transmits its data to all the nodes it is connected to.



Fig-1: CNN Architecture

The last wave of neural networks came in connection with the increase in computing power and the accumulation of experience. That brought Deep learning, where technological structures of neural networks have become more complex and able to solve a wide range of tasks that could not be effectively solved before. Image classification is a prominent example.

### 1.4 Deep Learning

Although the very first step towards neural networks was taken in 1943 with the work of Warren McCulloch and Walter Pitts, the first practical application using artificial neurons came with Frank Rosenblatt’s invention, the perceptron. A perceptron is the simplest possible version of an artificial neuron, and it has the basic essential attributes as follows:

• One or more numeric inputs with corresponding weights, positive or negative, for each input.

• A bias that can be either positive or negative. Can informally be described as the neurons resistance to "firing off".

• An activation function (in the case of the perceptron, the unit step function).

• A single output value, the activation function applied to the sum of the weighted inputs and the bias.

More informally stated the perceptron outputs 1 if the sum of the weighted inputs and the bias is bigger than 0, and 0 if not. Even though perceptron’s are not used in practice they led to the next logical step, the multilayer perceptron (MLP) or the feedforward neural. A feedforward neural network is simply artificial neurons in layers, with all the outputs from each neuron in the preceding layer fed forward, not backwards, into each neuron in the following layer, the exceptions being the input layer (consisting of passive neurons that do not transform the input) and the output layer. The layers between the first and last are called the hidden layers, which gives the network depth and therefore leads to the first part of the name of this chapter, deep learning, which is also a common name for the use of deep neural networks as a whole and associated techniques. Since each hidden layer, and the output layer, consist of neurons who individually are connected with the output from each neuron in the previous layer, those layers in a feedforward network are called fully connected layers. All neurons in the network have a unique set of weights and the activation functions are non-linear functions. That latter part will be expounded upon later in the chapter.

To train a neural network we need some other kind of measure of how big the current error is outside of the training error, some kind of measure of how well off the weights and biases in the network are as whole. To solve this a cost or objective function is introduced, a function that measures the total current error. The two important properties such an objective function must have is that it is non-negative for all inputs, and that it is zero or close to zero if the error is small. The direct goal of the training is thus to minimize the objective function. A simple approach here would be to choose the mean squared error as the cost to minimize, but in practice the cross entropy function is used instead due to an intrinsically better performance.

Neural networks can contain millions, tens of millions and even billions of parameters and there is no feasible way to find the minimum with methods from ordinary calculus. Instead Plot of the rectified linear unit (Relu) function and Softmax function as our Activation function.

Even though stochastic gradient descent, as described above, is an algorithmic solution to a mathematically unsolvable problem it still needs further work to be useful in practice. Once again, neural networks have a huge amount of parameters and computing each partial derivative of the objective function as in the naive implementation above is numerically unfeasible. The solution to the problem that makes stochastic gradient descent useful in practice is backpropagation, an algorithm that, simply put, calculates the error of one layer based upon the error of the preceding layer and updates its parameters accordingly. The name of the algorithm comes from that property, that the error, and the correction of the error, propagates backwards through the entire network from the output layer and backwards. The elegance of the algorithm is that it mirrors the path the activations took forward in the network and carries roughly the same computational cost.

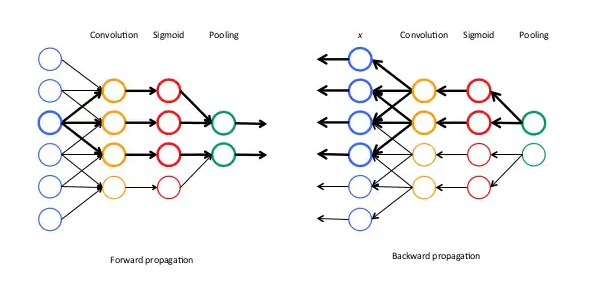


Fig-2: Forward Propagation Fig-3: Backward Propagation

As a final note on gradient descent there are more advanced variants in use today that builds upon the standard version with backpropagation, variants that adds additional elements such as dynamic learning rates. Explaining those algorithms in detail are outside of the scope of this report, but it is worth mentioning that they exist. As mentioned in the previous section there are inbuilt challenges connected to the machine learning process, and neural networks are no exception. The two problems that will be explored here are overfitting in the case of neural networks and the unstable gradient problem, a problem specific to the stochastic gradient descent algorithm applied to the training of neural networks. Overfitting in this context is remedied by methods such as gathering more and better data, something that has been mentioned earlier in this chapter, and regularization. It has been mentioned in the former section, but it is worth repeating that regularization can be described as limiting the amount of functions the machine learning algorithm can generate. In the case of neural networks there are many kinds of regularization but the discussion here will restrict itself to these following methods: L1 regularization, L2 regularization, drop out and data augmentation.

L1 and L2 regularization are two variants of the same theme and are built upon adding an extra term to the cost function, a term consisting of a weighted average of the sum of all the weights in the network. The difference between the two lies in that the sum is of the absolute values of the weights in the former case and of the squares of the weights in the latter case. The reason why this term is added is to penalize networks with too large weights, a penalty whose effect, informally stated, is to make the network generalize better by forcing it to choose less complex functions linking input to output. Dropout is a technique that is based upon dropping a random number of neurons in the hidden layers during each round of minibatches with a final adjustment of the weights in the end of a training run. The intended effect is, as with L1 and L2, to enforce better generalization. The last method, data augmentation, is built upon artificially expanding the available data by introducing random changes in the examples, e.g. flipping an image horizontally, shifting it slightly in a direction or rotating it slightly. Since the best cure for overfitting is more data this technique works in that direction to make the network generalize better.

The unstable gradient problem occurs due to the fact that the gradient, or the parameters’ rate of change, in a layer is a product of the rate change of all the layers before it. This can cause the rate of change to vanish entirely, the vanishing gradient problem, or go up drastically, the exploding gradient problem. The unwanted effects accumulate more rapidly the earlier in the network the layer in question is. This problem was, historically at least, a major hindrance to training networks beyond a certain depth. Luckily the problem seems to have been partially solved, partly due to the fact that the aforementioned rectified linear function and variants thereof have become the standard activation functions. A major cause of the problem was the use of saturating activation functions like the sigmoid and hyperbolic tangent functions, saturating in this context meaning the rate of change or derivative of the function goes to zero as the input becomes too large of a positive or negative number. In contrast the Relu function has a derivative that is either 0 or 1, which means that it does not saturate in its positive region and always propagate the gradient backwards. There have been other breakthroughs in this area, breakthroughs that will be discussed later. During this exposition the terms neural networks and feedforward neural networks have been used synonymously, which is not surprising since it is the latter that has been discussed and explained during the majority of this chapter. The terms are not entirely synonymous though, since there are neural networks with loops and feedback connections. That subclass of neural networks is called recurrent neural networks, and makes use of more complex layers than the ones mentioned so far, layers such as long short-term memory units (LSTMs). How recurrent neural networks work in detail is outside the scope of this report, but they are important and worth mentioning due to the fact that they are behind some of the latest breakthroughs in text and speech processing. There are other kinds of neural networks, but the focus of this work lies upon feedforward neural networks and their derivatives.

### 1.5 Deep Learning Frameworks

This chapter introduces the reader to the deep learning frameworks used in this study, as well as some other deep learning frameworks, but firstly: a short introduction to the purposes and functions of deep learning frameworks in general.

The algorithms and functions used in machine learning, and especially deep learning, involves a lot of mathematics; it can therefore be difficult and time consuming to implement the neural network from the ground up. Deep learning frameworks provide a high level API in order to make the implementation of neural networks simpler and more efficient. The frameworks make the implementation simpler and more efficient by abstracting away the underlying mathematics and by providing premade modules and code. By abstracting away the mathematical implementation the frameworks remove the requirement for the programmer to have an extensive mathematical background, thus making deep learning easier to work with and more available.

### 1.6 Convolutional neural networks and image classification

Convolutional neural networks (CNN) is a special architecture of artificial neural networks, proposed by Yann LeCun in 1988. CNN uses some features of the visual cortex. One of the most popular uses of this architecture is image classification. For example Facebook uses CNN for automatic tagging algorithms, Amazon — for generating product recommendations and Google — for search through among users’ photos.

Let us consider the use of CNN for image classification in more detail. The main task of image classification is acceptance of the input image and the following definition of its class. This is a skill that people learn from their birth and are able to easily determine that the image in the picture is an elephant. But the computer sees the pictures quite differently:

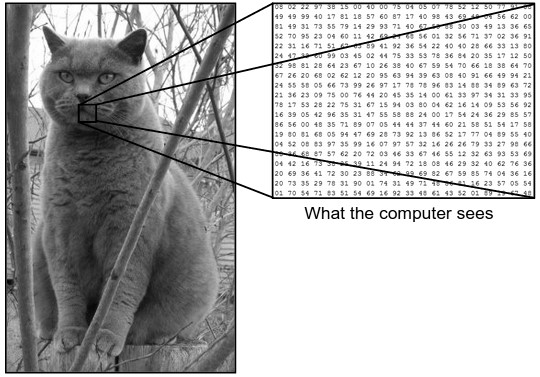
What we see vs. What computers see

Fig-4: Recognition of Image between Human vs Computer

Instead of the image, the computer sees an array of pixels. For example, if image size is 300 x 300. In this case, the size of the array will be 300x300x3. Where 300 is width, next 300 is height and 3 is RGB channel values. The computer is assigned a value from 0 to 255 to each of these numbers. This value describes the intensity of the pixel at each point.

To solve this problem the computer looks for the characteristics of the base level. In human understanding such characteristics are for example the trunk or large ears. For the computer, these characteristics are boundaries or curvatures. And then through the groups of convolutional layers the computer constructs more abstract concepts.

Based on our images dataset trained model we are going to classify the images like in the given images.Fig-5: Label Dataset

In more detail: the image is passed through a series of convolutional, nonlinear, pooling layers and fully connected layers, and then generates the output.

**The** **Convolution layer** is always the first. The image (matrix with pixel values) is entered into it. Imagine that the reading of the input matrix begins at the top left of image. Next the software selects a smaller matrix there, which is called a **filter**(or neuron, or core). Then the filter produces convolution, i.e. moves along the input image. The filter’s task is to multiply its values by the original pixel values. All these multiplications are summed up. One number is obtained in the end. Since the filter has read the image only in the upper left corner, it moves further and further right by 1 unit performing a similar operation. After passing the filter across all positions, a matrix is obtained, but smaller then a input matrix.

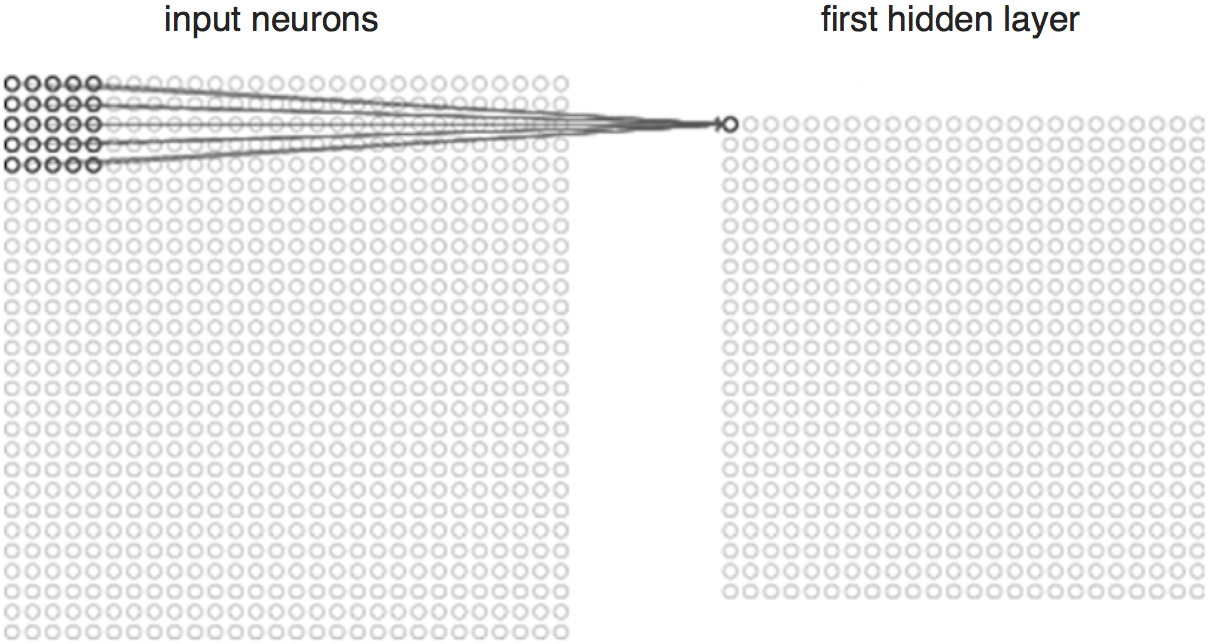


Fig-6: Mapping input pixel to the first hidden layer

This operation, from a human perspective, is analogous to identifying boundaries and simple colors on the image. But in order to recognize the properties of a higher level such as the trunk or large ears the whole network is needed.

The network will consist of several convolutional networks mixed with nonlinear and pooling layers. When the image passes through one convolution layer, the output of the first layer becomes the input for the second layer. And this happens with every further convolutional layer.

The nonlinear layeris added after each convolution operation. It has an activation function, which brings nonlinear property. Without this property a network would not be sufficiently intense and will not be able to model the response variable (as a class label).

The pooling layer follows the nonlinear layer. It works with width and height of the image and performs a down sampling operation on them. As a result the image volume is reduced. This means that if some features (as for example boundaries) have already been identified in the previous convolution operation, than a detailed image is no longer needed for further processing, and it is compressed to less detailed pictures.

After completion of series of convolutional, nonlinear and pooling layers, it is necessary to attach a fully connected layer. This layer takes the output information from convolutional networks. Attaching a fully connected layer to the end of the network results in an N dimensional vector, where N is the amount of classes from which the model selects the desired class.

A fragment of the code of this model written in Python will be considered further in the practical part.

### 1.7 PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using [whitespace](https://en.wikipedia.org/wiki/Whitespace_character) indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as [C++](https://en.wikipedia.org/wiki/C%2B%2B)or [Java](https://en.wikipedia.org/wiki/Java_(programming_language)). It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many [operating systems](https://en.wikipedia.org/wiki/Operating_system). [CPython](https://en.wikipedia.org/wiki/CPython), the [reference implementation](https://en.wikipedia.org/wiki/Reference_implementation) of Python, is [open source](https://en.wikipedia.org/wiki/Open_source) software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit [Python Software Foundation](https://en.wikipedia.org/wiki/Python_Software_Foundation). Python features a [dynamic type](https://en.wikipedia.org/wiki/Dynamic_type) system and automatic [memory management](https://en.wikipedia.org/wiki/Memory_management). It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigm), including [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming), [imperative](https://en.wikipedia.org/wiki/Imperative_programming), [functional](https://en.wikipedia.org/wiki/Functional_programming) and [procedural](https://en.wikipedia.org/wiki/Procedural_programming), and has a large and comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library)

# CHAPTER 2

# REQUIREMENT ANALYSIS

This chapter provides the details of the project’s need-based survey, system requirements, Hardware Requirements, Software Requirements.

### 2.1 Project Overview: -

In this paper author is describing concept to classify the images by using python TENSORFLOW and Deep Learning algorithm.

**Existing System: -**

* In this paper, instead of recognizing a large number of disparate categories, the problem of recognizing a large number of classes within one category is investigated that of objects. Classifying objects pose an extra challenge over categories, because of the large similarity between classes. Previous work on image classification has deal with a small number of classes.
* To implement this technique, we need to train some objects and generate a model and then by uploading any image deep learning algorithm will convert uploaded image into gray scale format and apply that image on train model to predict best match species name for uploaded image.
* SVMs are a class of supervised learning models that have been widely used for classification and regression SVMs are based on statistical learning theory and are better able to avoid local optima than other classification algorithms. An SVM is a kernel-based learning algorithm that seeks the optimal hyper plane. The kernel learning process maps the input patterns into a higher-dimensional feature space in which linear separation is feasible. The existing kernel functions can be classified as either local or global kernel functions. Local kernel functions have a good learning ability but do not have good generalization ability. By contrast, global kernel functions have good generalization ability but a poor learning ability. For example, the radial kernel function is known to be a local function, whereas the polynomial kernel function is a global kernel function. The main challenge lies in determining which kernel function should be used for the current problem instance or the current decision point. This is because the kernel selection process strongly depends on the distribution of the input vectors and the relationship between the input vector and the output vector (predicted variables). However, the feature space distribution is not known in advance and may change during the course of the solution process, especially in big data cyber security. Consequently, different kernel functions may work well for different instances or in different stages of the solution process and kernel selection may thus have a crucial impact on SVM performance. To address this issue, in this work, we use multiple kernel functions to improve the accuracy of our algorithm and avoid the shortcomings of using a single kernel function.

**Proposed System: -**

* Represents the actual flow of the proposed system. To develop such system a trained dataset is required to classify an image. Trained dataset consists of two parts trained result and test result. The dataset has to be retrained to achieve higher accuracy in identification using imageclassification.py in local system. The training dataset is made using 10000 steps taking into consideration that higher the number of steps higher is its accuracy. The accuracy of training dataset is 93%. The testing dataset consists of nearly 1000 images with an accuracy of 90%. Dataset is validated with an accuracy of 75% to increase the performance of system. Whenever a user will upload an input file on website, the image is temporarily stored in primary physical memory. This input file is then feed to system and given to CNN where CNN is coupled with trained dataset. The CNN consists of convolutional layers which contain various alignments/features such as head, body, color, and shape, entire image of objects is considered for classification to yield maximum accuracy.

**Advantages: -**

* Dataset is validated with an accuracy of 93% to increase the performance of system.

### 2.2 Functional requirements: -

In software engineering, a functional requirement defines a system or its component. It describes the functions a software must perform. A function is nothing but inputs, its behavior, and outputs. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform.

Functional software requirements help you to capture the intended behavior of the system. This behavior may be expressed as functions, services or tasks or which system is required to perform

.

### 2.3 Non –Functional Requirements: -

A non-functional requirement defines the quality attribute of a software system. They represent a set of standards used to judge the specific operation of a system. Example, how fast does the website load?

A non-functional requirement is essential to ensure the usability and effectiveness of the entire software system. Failing to meet non-functional requirements can result in systems that fail to satisfy user needs.

### 2.4 Hardware Requirements: -

Since the computational aspect of the project is of importance to Deep Learning it is important to know the hardware that was used in the evaluation process. The training and evaluation of the neural network model has been done on a Windows 10 computer using a quad-core CPU at 3.4 GHz. Comparing the results from home user-level of hardware to the kind of hardware that should be available to a particle detector could give us an inkling of how effective this kind of CNN could be. Let us also note that the final experimental setup will be generating events at frequencies of 2 or 20 MHz, which is relevant in the sense of data points to process per second.

* Operating System supported by

1. Windows 8

2. Windows 10

* Processor – Intel i5 @ 2.50 or Higher
* RAM -- 8 GB
* Space on Hard Disk -- Minimum 512 GB

### 2.5 Software Requirements: -

* For developing the Application

1. PyCharm
2. Anaconda Navigator

#### **2.5.1 Anaconda**

**Download and Install Anaconda**

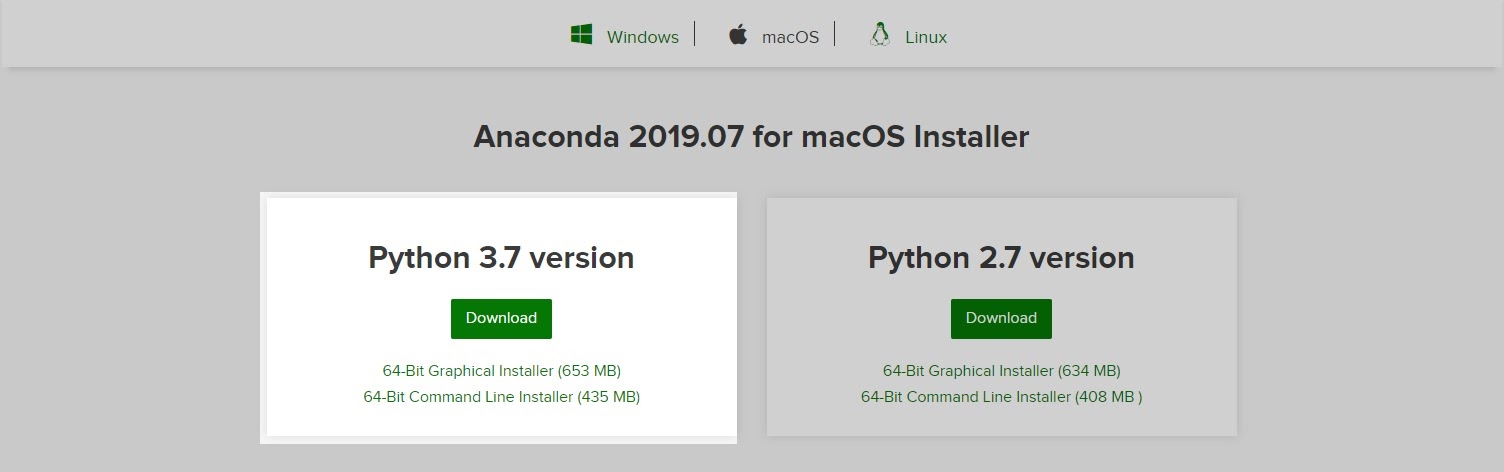
##### ***Step #1: Go To Anaconda.com***

Go to [Anaconda.com](https://www.anaconda.com/distribution/#windows), and download the Anaconda version for Windows.

##### **Step #2: Download the Python 3 version for Windows.**

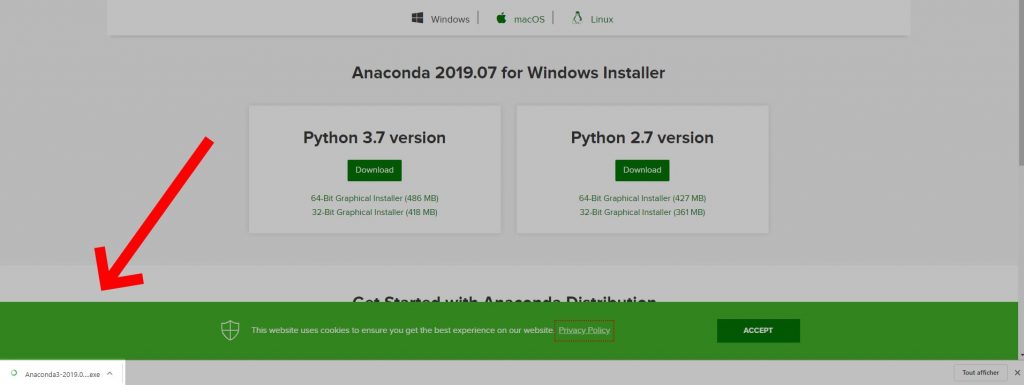
Version 2 will not be updated past 2020, so do yourself a favor and start using V3.

[You Might Also Like Python Script Automation Using Task Scheduler (Windows)](https://www.jcchouinard.com/python-automation-using-task-scheduler/" \t "_self)

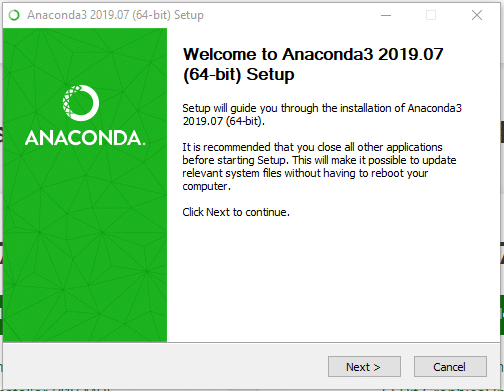


##### **Step #3: Double-click on the executable file.**

To get the installation of Anaconda started on your operating system open the executable file in your Download folder.



##### **Step #4: Click Next**



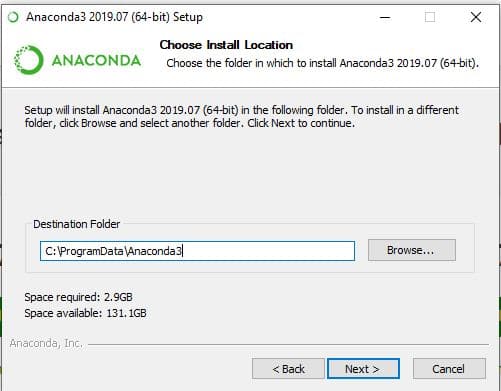
##### **Step #5: Click I agree to the terms and conditions**

##### **Step #6: Select Who You Want To Give Anaconda To**https://www.jcchouinard.com/wp-content/uploads/2019/08/users-to-install.jpg

This step will ask you if you want to install Anaconda just for you or for all the users using this PC. Click “Just-Me”, or “All users”, depending on your preference. Both options will do but to select “all users” you will need admin privileges.

##### ***Step #7: Select the installation location***

If you have selected “All users”, by default, Anaconda will get installed in the *C:\ProgramData\Anaconda3* folder. So make sure that you have at least the right amount of space available to install the subdirectory comparing it the space required.

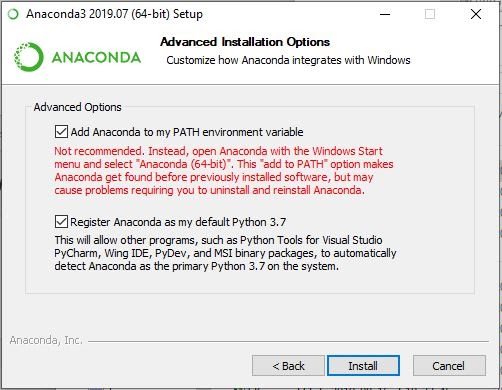


##### **Step #8: Select the environment variables**

Depending on if you have any version of Python already installed on your operating system, or not, to do different set-up.

##### **If You Are Installing Python for the First Time**

Check the Add Anaconda to my PATH environment variable. This will let you use Anaconda in your command prompt.

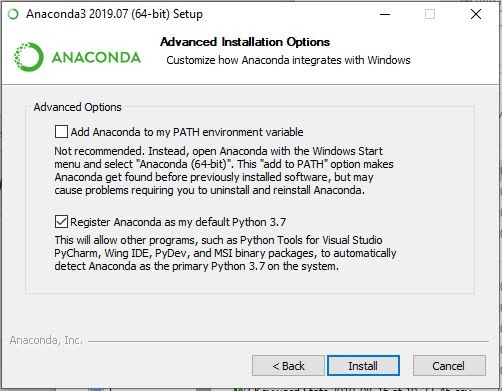


##### **If You Already Have Python Installed**

Leave Add Anaconda to my PATH environment variable unchecked.

Leaving it unchecked means that you will have to use Anaconda Command Prompt in order to use Anaconda.

So, unless you add the PATH later, you will not be able to use Python from your command prompt.

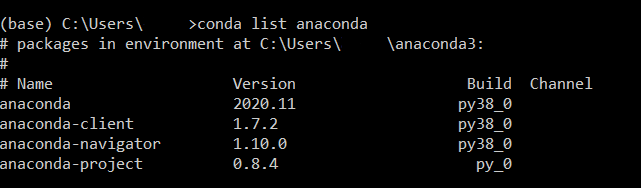


Python is not usually included by default on Windows, however we can check if any version exists on the system.

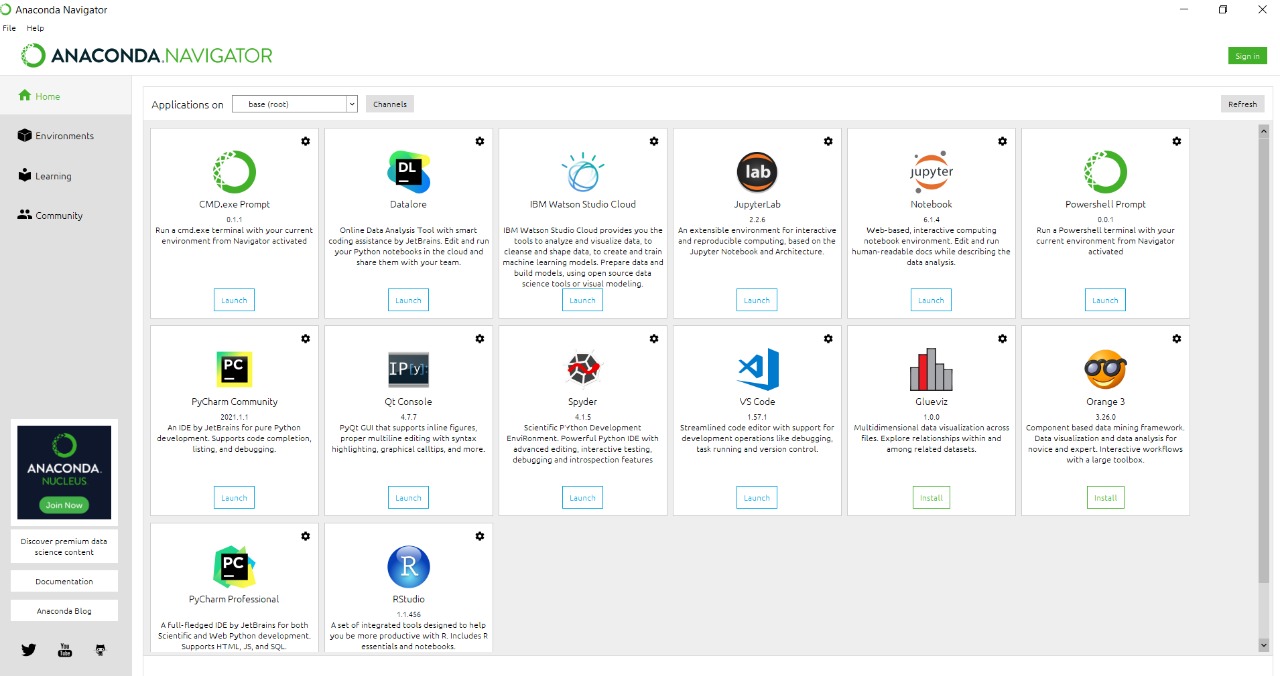
##### **To know if you have Python Installed.**

1. Go to Start Menu and type “Anaconda Prompt” to open it.
2. If nothing is showing means you don’t have Anaconda Navigator Install.
3. Type the following command and hit the Enter key “conda list anaconda”
4. This result.

*$* conda list anaconda



##### **Step #9: Click next and then “Finish” and it display as**



#### **2.5.2 PyCharm**

[](https://www.guru99.com/images/1/090719_0649_11BESTPytho1.png)

PyCharm is a cross-platform IDE used for Python programming. It is one of the best Python IDE editor that can be used on Windows, macOS, and Linux. This software contains API that can be used by the developers to write their own Python plugins so that they can extend the basic functionalities.

**Price:** Free

**Features:**

* It is an intelligent Python code editor supports for CoffeeScript, JavaScript, CSS, and TypeScript.
* Provides smart search to jump to any file, symbol, or class.
* Smart Code Navigation
* This Python editor offers quick and safe refactoring of code.
* It allows you to access PostgreSQL, Oracle, MySQL, SQL Server, and many other databases from the IDE.

**Download Link:**<https://www.jetbrains.com/pycharm/>

#### **2.5.3 Jupyter**

[](https://www.guru99.com/images/1/090719_0649_11BESTPytho7.png)

Jupyter is a tool for people who have just started with data science. It is easy to use, interactive data science IDE across many programming languages that just not work as an editor, but also as an educational tool or presentation.

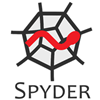
**Price:** Free

**Features:**

* It is one of the best Python IDE that supports for Numerical simulation, data cleaning machine learning data visualization, and statistical modeling.
* Combine code, text, and images.
* Support for many programming languages.
* Integrated data science libraries (Matplotlib, NumPy, Pandas).

**Download Link:** <https://jupyter.org/install.html>

#### **2.5.4 Spyder**

[](https://www.guru99.com/images/1/090719_0649_11BESTPytho2.png)

Spyder is a scientific integrated development environment written in Python. This software is designed for and by scientists who can integrate with Matplotlib, SciPy, NumPy, Pandas, Cython, IPython, SymPy, and other open-source software. Spyder is available through Anaconda (open-source distribution system) distribution on Windows, macOS, and Linux.

**Price**: Free

**Features:**

* It is one of the best Python IDE for Windows which allows you to run Python code by cell, line, or file.
* Plot a histogram or time-series, make changes in dateframe or numpy array.
* It offers automatic code completion and horizontal/vertical splitting.
* Find and eliminate bottlenecks
* An interactive way to trace each step of Python code execution.

**Download Link:** <https://www.spyder-ide.org>

### Technologies and Languages used to Develop

-- Python Environment

**GUI Tkinter Toolkit:**

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

Create a tkinter app:

Importing the module – tkinter

Create the main window (container)

Add any number of widgets to the main window

Apply the event Trigger on the widgets.

Importing tkinter is same as importing any other module in the Python code. Note that the name of the module in Python 2.x is ‘Tkinter’ and in Python 3.x it is ‘tkinter’.

**import tkinter**

#### Used Libraries:

1. Numpy

2. Pandas

3. OpenCV

4. Imutils

5. Tensorflow

##### Numpy

NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

**Numeric**, the ancestor of NumPy, was developed by Jim Hugunin. Another package Num array was also developed, having some additional functionalities. In 2005, Travis Oliphant created NumPy package by incorporating the features of Num array into Numeric package. There are many contributors to this open source project.

Operations using NumPy

Using NumPy, a developer can perform the following operations −

* Mathematical and logical operations on arrays.
* Fourier transforms and routines for shape manipulation.
* Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

##### Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures.

Key Features of Pandas

* Fast and efficient Data Frame object with default and customized indexing.
* Tools for loading data into in-memory data objects from different file formats.
* Data alignment and integrated handling of missing data.
* Reshaping and pivoting of date sets.
* Label-based slicing, indexing and sub setting of large data sets.
* Columns from a data structure can be deleted or inserted.
* Group by data for aggregation and transformations.
* High performance merging and joining of data.
* Time Series functionality.

##### OpenCV

OpenCV is a cross-platform library using which we can develop real-time **computer vision applications**. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.

Let’s start the chapter by defining the term "Computer Vision".

Computer Vision can be defined as a discipline that explains how to reconstruct, interrupt, and understand a 3D scene from its 2D images, in terms of the properties of the structure present in the scene. It deals with modeling and replicating human vision using computer software and hardware.

Computer Vision overlaps significantly with the following fields −

* **Image Processing** − It focuses on image manipulation.
* **Pattern Recognition** − It explains various techniques to classify patterns.

In our project it is used for image processing as well as putting the text in the images.

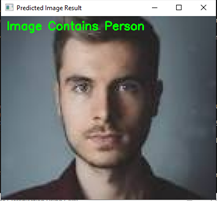


Fig-1: Used for testing

In this image “Image Contains Person can be written with the help of OpenCV.

In other words, we are doing the image Manipulation concept using OpenCV.

Imutils:

A series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, displaying Matplotlib images, sorting contours, detecting edges, and much more easier with OpenCV and both Python 2.7 and Python 3.

TensorFlow:

TensorFlow is an open-source library for Deep Learning and Machine Learning. Developed by the google brain team and released in November 2015. TensorFlow is mainly used for: Classification, Perception, Understanding, Discovering, TensorFlow can be describe as Tensor + Flow = Data + Flow. TensorFlow uses a dataflow graph to represent your computation.

* Parallelism. (it is easy for the system to identify operations that can execute in parallel )
* Distributed execution (it is possible for TensorFlow to partition your program across multiple devices CPUs, GPUs, and TPUs)
* Compilation (generate faster code)
* Portability

# Chapter 3

# Methodology:

### 3.1 CONVOLUTION NEURAL NETWORK:

CNN consists of four layers: convolutional layer, activation layer, pooling layer and fully connected. Convolutional layer allows extracting visual features from an image in small amounts. Pooling is used to reduce the number of neurons from previous convolutional layer but maintaining the important information. Activation layer passes a value through a function which compresses values into range. Fully connected layer connects a neuron from one layer to every neuron in another layer. As CNN classifies each neuron in depth, so it provides more accuracy.

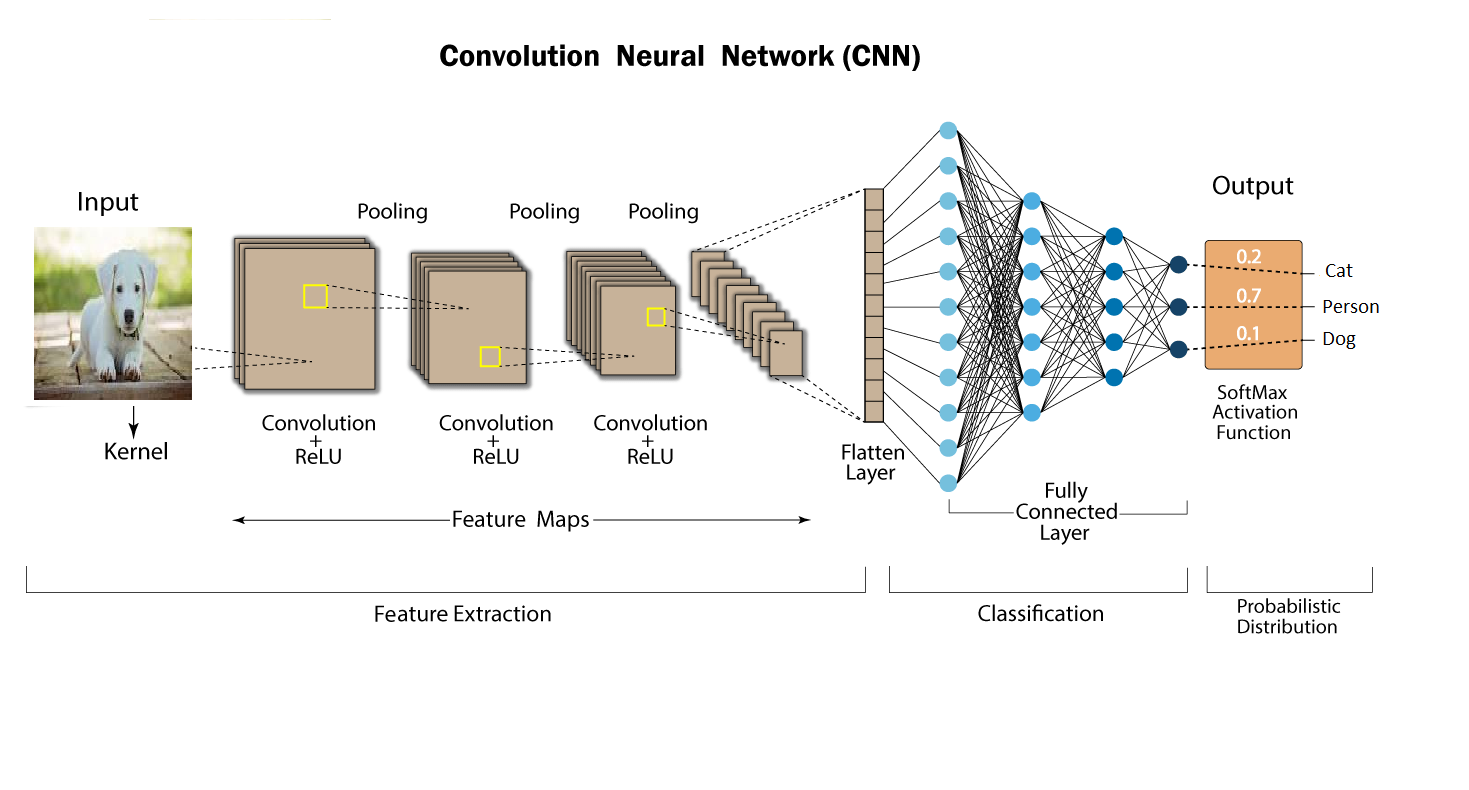


Fig 1: CNN Architecture

The CNN have two components:

1) Feature extraction part: Features are detected when network performs a series of convolutional and pooling operation.

2) Classification part: Extracted features are given to fully connected layer which acts as classifier.

Feature Extraction Features are detected when network performs a series of convolutional and pooling operation.

Convolutional Layer: Convolutional neural network layer types mainly include three types, namely convolutional layer, pooling layer and fully-connected layer Convolutional layers apply a convolution operation to the input, passing the result to the next layer. The convolution emulates the response of an individual neuron to visual stimuli. Each convolutional neuron processes data only for its receptive field. Although fully connected feed forward neural networks can be used to learn features as well as classify data, it is not practical to apply this architecture to images. A very high number of neurons would be necessary, even in a shallow (opposite of deep) architecture, due to the very large input sizes associated with images, where each pixel is a relevant variable. For instance, a fully connected layer for a (small) image of size 100 x 100 has 10000 weights for each neuron in the second layer. The convolution operation brings a solution to this problem as it reduces the number of free parameters, allowing the network to be deeper with fewer parameters.

For instance, regardless of image size, tiling regions of size 5 x 5, each with the same shared weights, requires only 25 learnable parameters. In this way, it resolves the vanishing or exploding gradients problem in training traditional multi- layer neural networks with many layers by using back propagation. The aim of Convolutional layer is to learn feature representations of the inputs. As shown in above, Convolutional layer is consisting of several feature maps. Each neuron of the same feature map is used to extract local characteristics of different positions in the former layer, but for single neurons, its extraction is local characteristics of same positions in former different feature map. In order to obtain a new feature, the input feature maps are first convolved with a learned kernel and then the results are passed into a nonlinear activation function.

Pooling Layers: Convolutional networks may include local or global pooling layers, which combine the outputs of neuron clusters at one layer into a single neuron in the next layer. For example, max pooling uses the maximum value from each of a cluster of neurons at the prior layer another example is average pooling, which uses the average value from each of a cluster of neurons at the prior layer. The sampling process is equivalent to fuzzy filtering. The pooling layer has the effect of the secondary feature extraction, it can reduce the dimensions of the feature maps and increase the robustness of feature extraction. It is usually placed between two Convolutional layers. The size of feature maps in pooling layer is determined according to the moving step of kernels. The typical pooling operations are average pooling and max pooling. We can extract the high-level characteristics of inputs by stacking several Convolutional layer and pooling layer.

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional multi-layer perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images. In general, the classifier of Convolutional neural network is one or more fully-connected layers. They take all neurons in the previous layer and connect them to every single neuron of current layer. There is no spatial information preserved in fully-connected layers. The last fully-connected layer is followed by an output layer. For classification tasks, softmax regression is commonly used because of it generating a well-performed probability distribution of the outputs. Another commonly used method is SVM, which can be combined with CNNs to solve different classification tasks.

Receptive field: In neural networks, each neuron receives input from some number of locations in the previous layer. In a fully connected layer, each neuron receives input from every element of the previous layer. In a convolutional layer, neurons receive input from only a restricted subarea of the previous layer. Typically the subarea is of a square shape (e.g., size 5 by 5). The input area of a neuron is called its receptive field. So, in a fully connected layer, the receptive field is the entire previous layer. In a convolutional layer, the receptive area is smaller than the entire previous layer.

Weights: Each neuron in a neural network computes an output value by applying some function to the input values coming from the receptive field in the previous layer. The function that is 21 applied to the input values is specified by a vector of weights and a bias (typically real numbers). Learning in a neural network progresses by making incremental adjustments to the biases and weights. The vector of weights and the bias are called a filter and represents some feature of the input. A distinguishing feature of CNNs is that many neurons share the same filter. This reduces memory footprint because a single bias and a single vector of weights is used across all receptive fields sharing that filter, rather than each receptive field having its own bias and vector of weights.

### 3.2 DATASET:

A dataset is a collection of data. For performing action related to objects a dataset named train and test dataset. The detailed information about the dataset is as follows: Number of categories: 5 with number of files 400, Number dataset is validated with an accuracy of 93% to increase the performance of system.

**Implementation:**

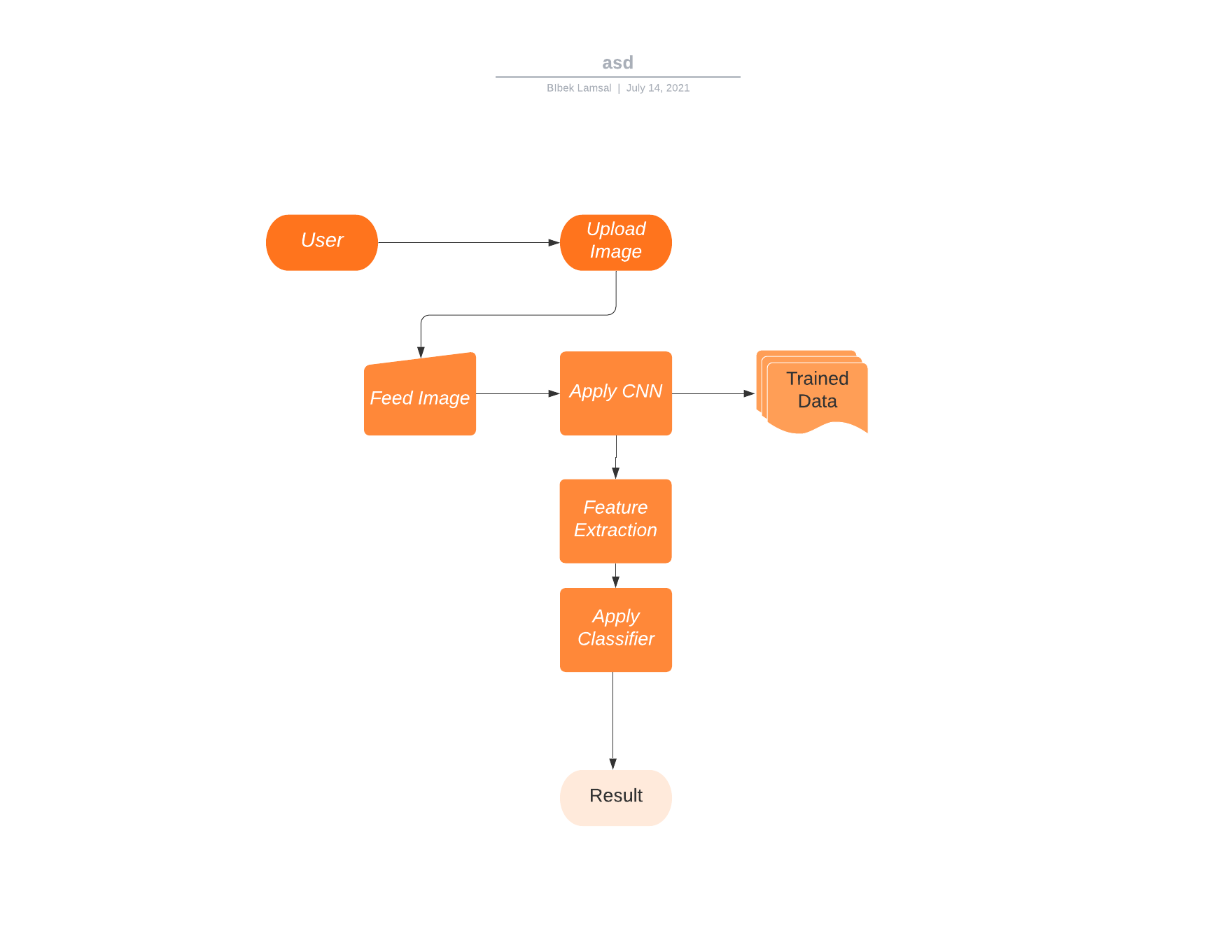


Fig 2: Proposed Architecture

The above figure no.2 represents the actual flow of the proposed system. To develop such system a trained dataset is required to classify an image. Trained dataset consists of two parts trained result and test result.

Whenever a user will upload an input file on website, the image is temporarily stored in database. This input file is then feed to system and given to CNN where CNN is coupled with trained dataset. A CNN consists of various convolutional layers. Various alignments/features such as head, body, color, shape, entire image of object is considered for classification to yield maximum accuracy. Each alignment is given through deep convocational network to extract features out from multiple layers of network. Then an unsupervised algorithm called deep learning using CNN is used to classify that image. By using the Tkinter application this will be deployed.

### 3.3 INPUT DESIGN AND OUTPUT DESIGN

##### 3.3.1 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

* What data should be given as input?
* How the data should be arranged or coded?
* The dialog to guide the operating personnel in providing input.
* Methods for preparing input validations and steps to follow when error occur.

##### 3.3.2 OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

##### 3.3.3 OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

* Convey information about past activities, current status or projections of the
* Future.
* Signal important events, opportunities, problems, or warnings.
* Trigger an action.
* Confirm an action.

# CHAPTER 4

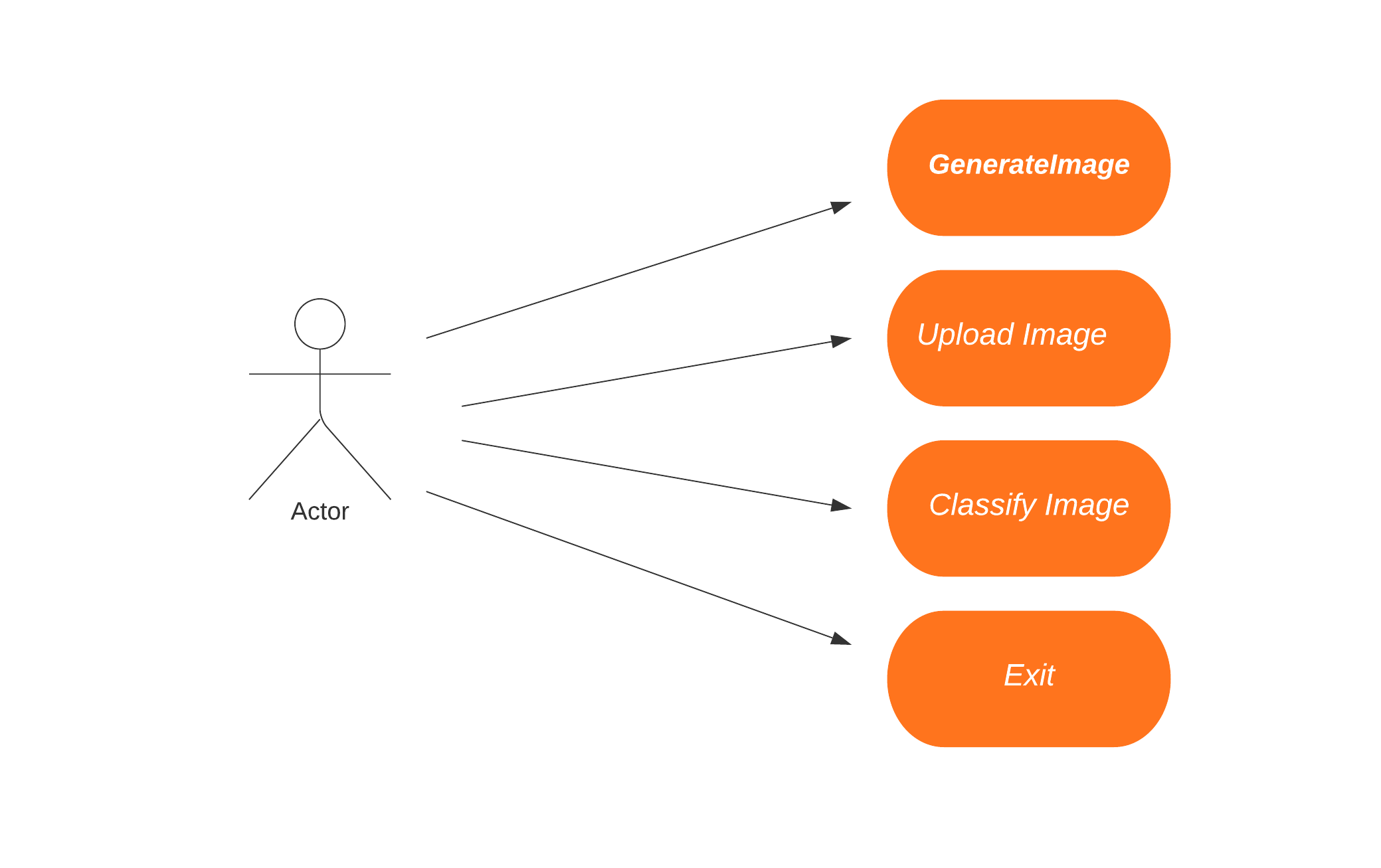
# ANALYSIS AND DESIGN PHASE

### 4.1 INTRODUCTION

This chapter provides the design phase of the Application. To design the project, we use the UML diagrams. The Unified Modelling Language (UML) is a general- purpose, developmental, modelling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system.

### 4.2 USE CASE DIAGRAM

g



**Fig Use case Diagram**

The use case diagram is used to represent all the functional use cases that are involved in the project.

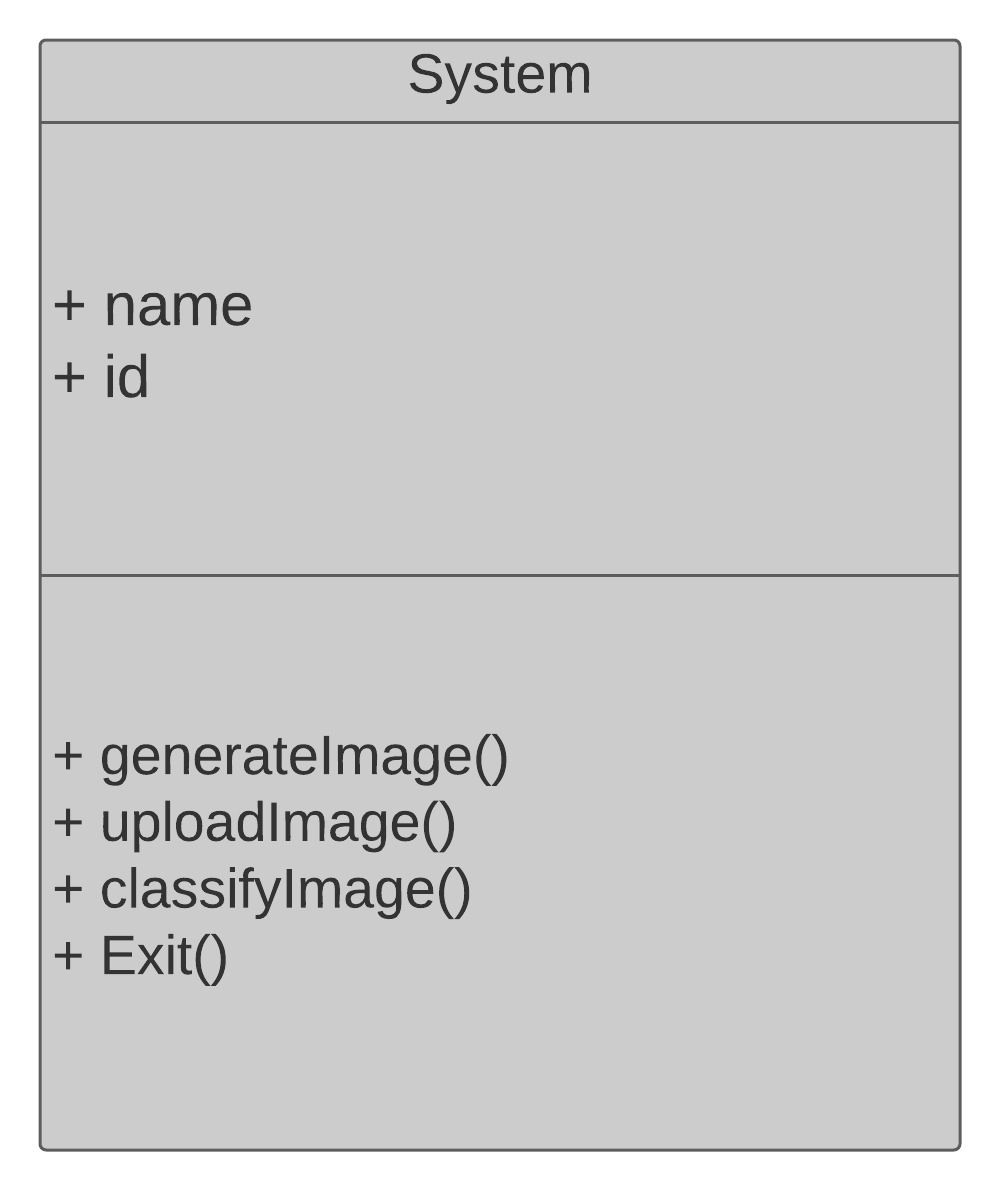
The above diagram represents the main two **actors** in the project, they are

* + - User

### CLASS DIAGRAM

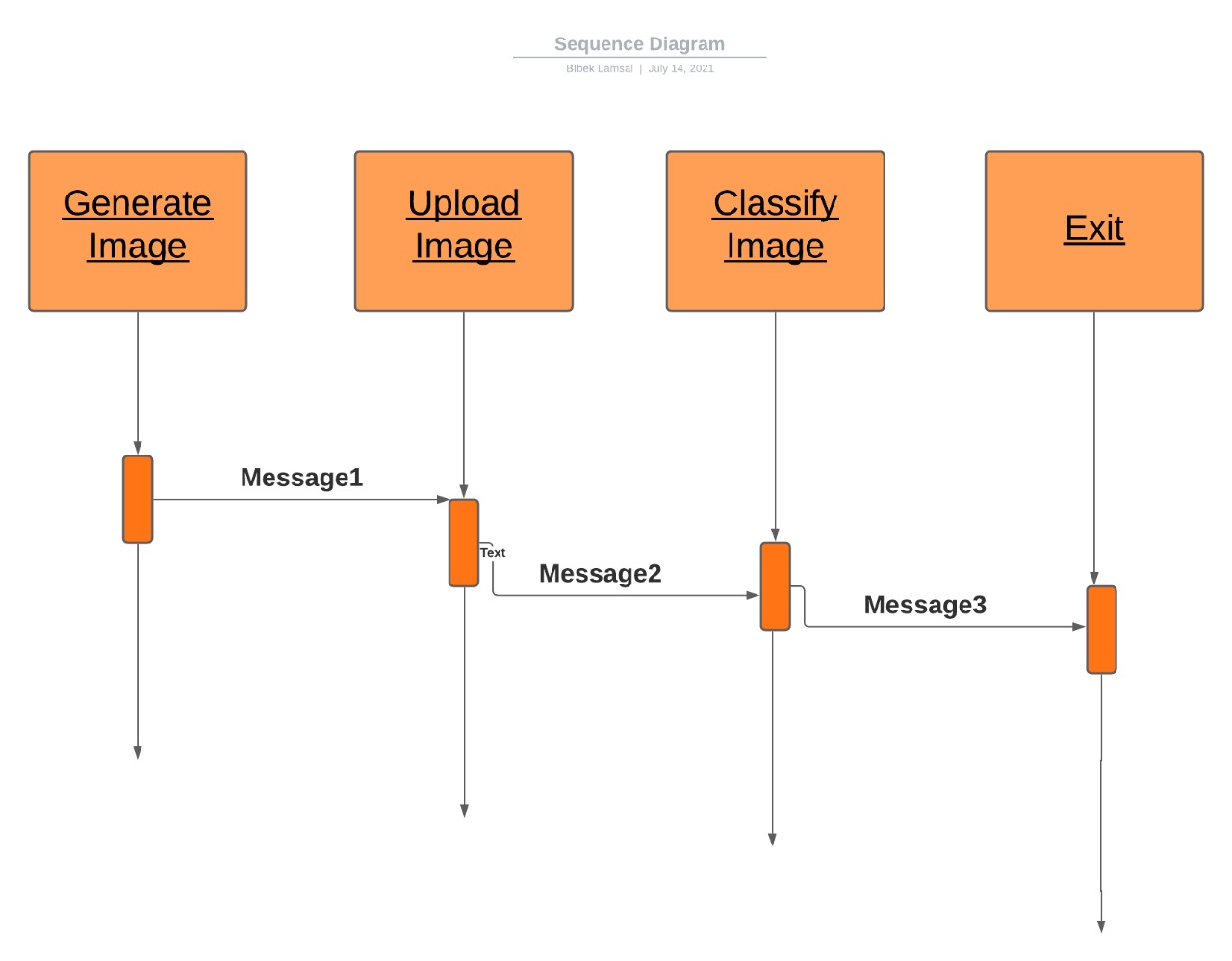
The above-mentioned class diagram represents the Chabot system workflow model. This diagram has class models with class names as

* + - User
    - Home screen



**Fig: class diagram**

### 4.4 SEQUENCE DIAGRAM

****

**Fig: sequence diagram**

The above diagram represents the sequence of flow of actions in the system.

The sequence diagram captures the time sequence of the message flow from one object to another and the collaboration diagram describes the organization of objects in a system taking part in the message flow.

### ACTIVITY DIAGRAM

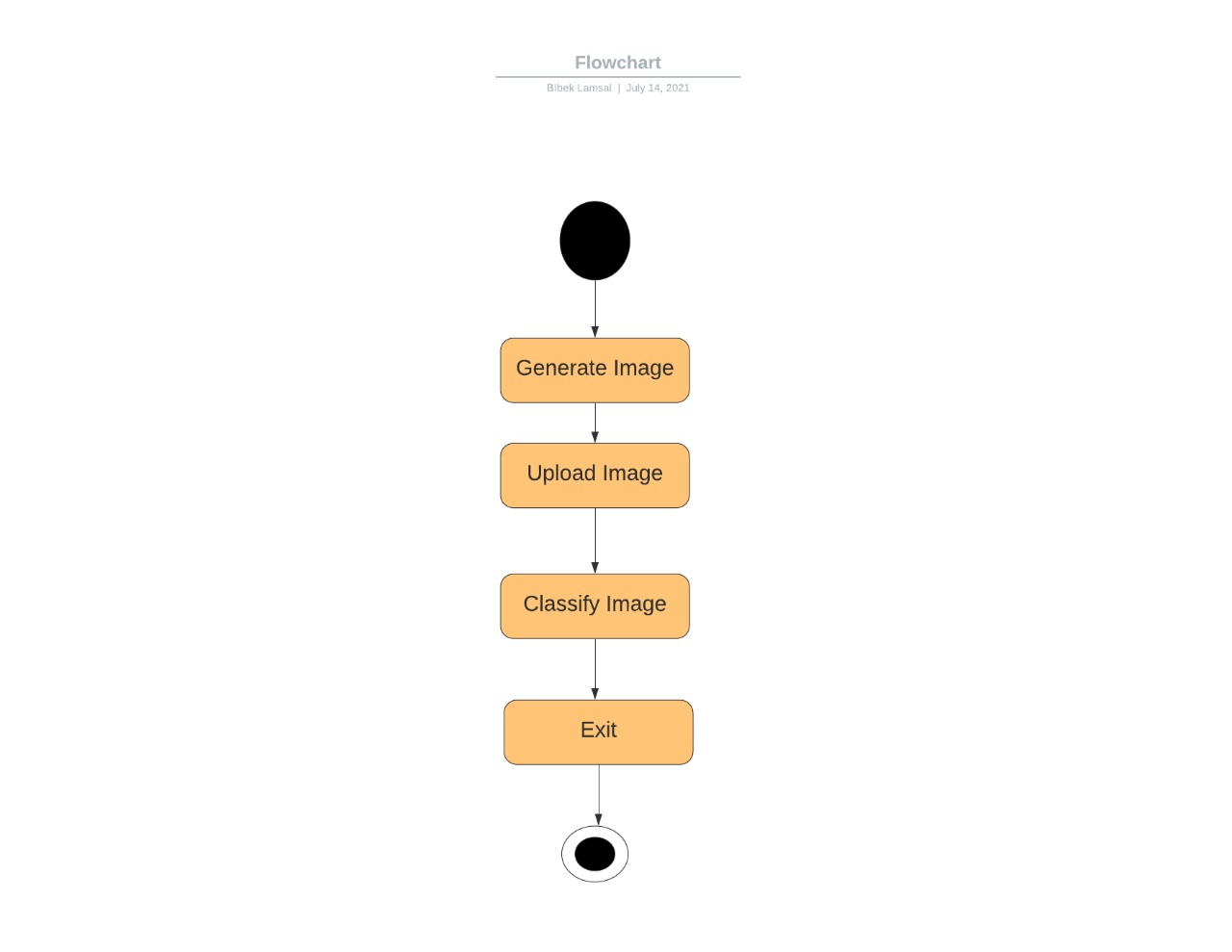
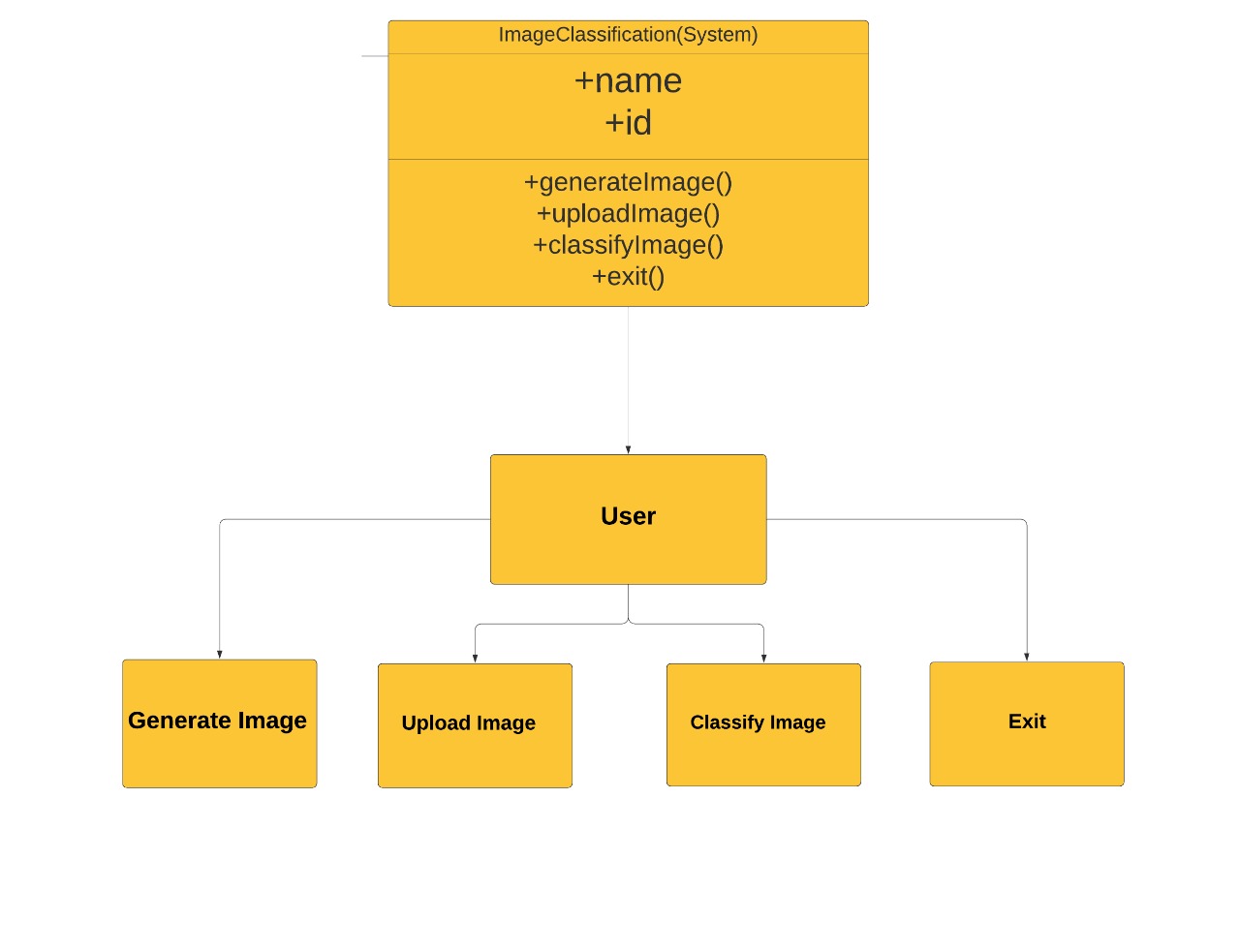
****

Fig: Activity Diagram

### 4.6 OBJECT DIAGRAM

Fig- : Object Diagram



To capture a particular system, numbers of class diagrams are limited. However, if we consider object diagrams then we can have unlimited number of instances, which are unique in nature. Only those instances are considered, which have an impact on the system.

The following diagram is an example of an object diagram. It represents the Order management system which we have discussed in the chapter Class Diagram. The following diagram is an instance of the system at a particular time of purchase. It has the following objects.

* Generate Image
* Upload Image
* Classify Image
* Exit

### 4.7 COMPONENT DIAGRAM

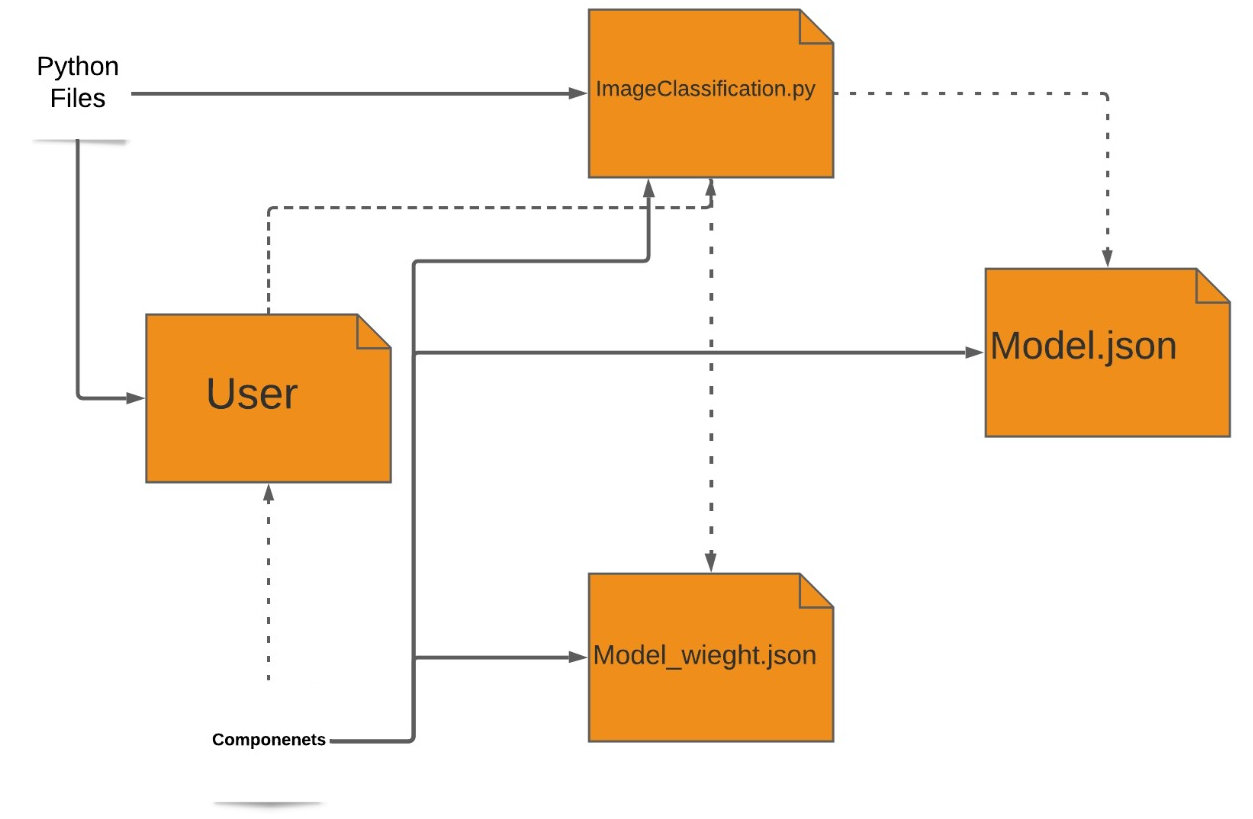


Fig- : Component Diagram

These diagrams show the physical components of a system. To clarify it, we can say that component diagrams describe the organization of the components in a system.

Organization can be further described as the location of the components in a system. These components are organized in a special way to meet the system requirements

### 4.8 DEPLOYMENT DIAGRAM

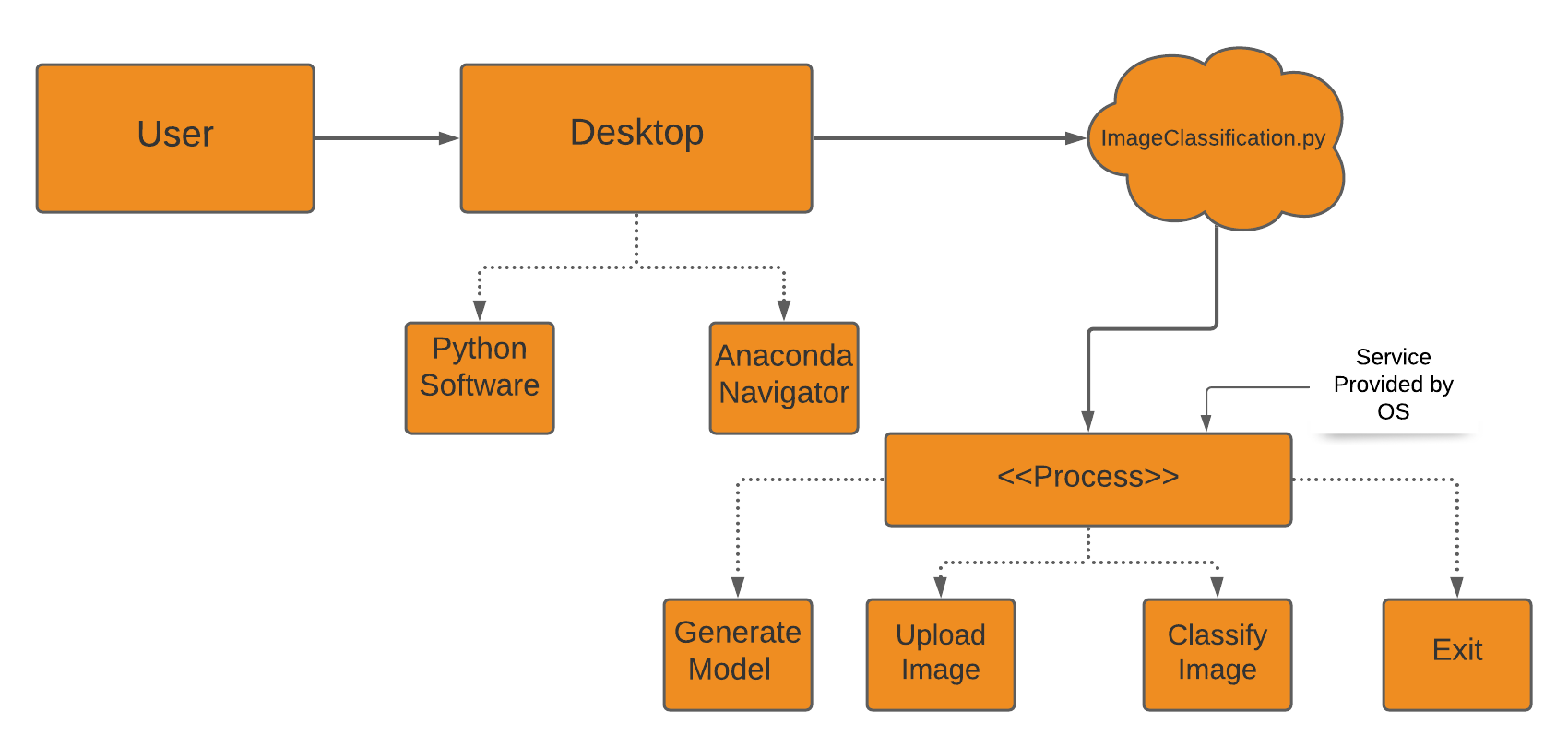


Fig- : Deployment Diagram

These diagrams are used to describe the physical components (hardware), their distribution, and association.

Deployment diagrams can be visualized as the hardware components/nodes on which the software components reside.

Software applications are developed to model complex business processes. Efficient software applications are not sufficient to meet the business requirements. Business requirements can be described as the need to support the increasing number of users, quick response time, etc.

# CHAPTER 5

## SYSTEM LOW LEVEL DESIGN

This chapter mainly provides the overview on modules of the application, objectives of the project and a detailed project overview.

### 5.1 Model Construction Steps:

1. Model construction

2. Model training

3. Model testing

4. Model evaluation

**Model construction** depends on machine learning algorithms. In this projects case, it was neural networks.

Such an algorithm looks like:

1. begin with its object: model = Sequential()
2. then consist of layers with their types: model.add(type\_of\_layer())
3. After adding a sufficient number of layers the model is compiled. At this moment Keras communicates with Tensor Flow for construction of the model. During model compilation it is important to write a loss function and an optimizer algorithm. It looks like: model.comile (loss= ‘name\_of\_loss\_function’, optimizer= ‘name\_of\_opimazer\_alg’) the loss function shows the accuracy of each prediction made by the model.

Before model training it is important to scale data for their further use.

After model construction it is time for **model training.**In this phase, the model is trained using training data and expected output for this data.

Its look this way: model.fit (training\_data, expected\_output).

Progress is visible on the console when the script runs. At the end it will report the final accuracy of the model.

Once the model has been trained it is possible to carry out **model testing.**During this phase a second set of data is loaded. This data set has never been seen by the model and therefore its true accuracy will be verified.

After the model training is complete, and it is understood that the model shows the right result, it can be saved by: model.save (“name\_of\_file.h5”).

Finally, the saved model can be used in the real world. The name of this phase is **model evaluation**. This means that the model can be used to evaluate new data.

### 5.2 Application of Built in Modules Data Structure:

* To implement this technique, we need to train objects and generate a model and then by uploading any image deep learning algorithm will convert uploaded image into gray scale format and apply that image on train model to predict best match species name for uploaded image.
* In this process we are using TensorFlow, Keras, Numpy, Pandas, OpenCV etc. which helps us to build project fast and efficient way .

# CHAPTER 6

# SYSTEM STUDY

**FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

**Three key considerations involved in the feasibility analysis are,**

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

### 6.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### 6.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

### 6.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

# CHAPTER 7

# IMPLEMENTATION

This chapter mainly provides the sample code and implementation of the project.

### 7.1 Sample Code

|  |  |  |  |
| --- | --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161 | # {'cars': 0, 'cats': 1, 'dogs': 2, 'person': 3, 'planes': 4}  **from** **tkinter** **import** \*  **import** **tkinter**  **from** **tkinter** **import** filedialog  **import** **numpy** **as** **np**  **from** **tkinter.filedialog** **import** askopenfilename  **import** **pandas** **as** **pd**  **from** **tensorflow.keras.optimizers** **import** Adam  **from** **keras.models** **import** model\_from\_json  **from** **tkinter** **import** simpledialog  **from** **keras.models** **import** Sequential  **from** **keras.layers** **import** Convolution2D  **from** **keras.layers** **import** MaxPooling2D  **from** **keras.layers** **import** Flatten  **from** **keras.layers** **import** Dense  **import** **os**  **from** **keras.preprocessing** **import** image  **from** **keras.preprocessing.image** **import** ImageDataGenerator  **from** **tkinter** **import** messagebox  **import** **cv2**  **from** **imutils** **import** paths  **import** **imutils**  #Required lib. for printing the predicted result in the output anaconda prompt  **from** **colorama** **import** init  **from** **colorama** **import** Fore, Back, Style  **global** filename  **global** loaded\_model  dic\_labels = { **0** : 'CAR', **1** : 'CAT', **2** : 'DOG', **3** : 'PERSON', **4** : 'PLANES'}  **class** **ImageClassificationSystem**:  **def** **\_\_init\_\_**(self):  self.main = tkinter.Tk()  self.main.title("Image Classification Using CNN")  self.main.geometry("600x500")  **def** **upload**(self):  **global** filename  filename = filedialog.askopenfilename(initialdir="testimages")  messagebox.showinfo("File Information", "image file loaded")  **def** **generateModel**(self):  **global** loaded\_model  **if** os.path.exists('model.json'):  **with** open('model.json', "r") **as** json\_file:  loaded\_model\_json = json\_file.read()  loaded\_model = model\_from\_json(loaded\_model\_json)  loaded\_model.load\_weights("model\_weights.h5")  loaded\_model.make\_predict\_function()  print(loaded\_model.summary)  messagebox.showinfo("Model Generated",  "CNN Model Generated on Train & Test Data. See black console for details")  **else**:  classifier = Sequential()  classifier.add(Convolution2D(**32**, **3**, **3**, input\_shape=(**48**, **48**, **3**), activation='relu'))  classifier.add(MaxPooling2D(pool\_size=(**2**, **2**)))  classifier.add(Convolution2D(**32**, **3**, **3**, activation='relu'))  classifier.add(MaxPooling2D(pool\_size=(**2**, **2**)))  classifier.add(Flatten())  classifier.add(Dense(units=**128**, activation='relu'))  classifier.add(Dense(units=**5**, activation='softmax'))  classifier.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])  train\_datagen = ImageDataGenerator()  test\_datagen = ImageDataGenerator()  training\_set = train\_datagen.flow\_from\_directory('data/train',  target\_size=(**48**, **48**),  batch\_size=**32**,  class\_mode='categorical',  shuffle=**True**)  test\_set = test\_datagen.flow\_from\_directory('data/validation',  target\_size=(**48**, **48**),  batch\_size=**32**,  class\_mode='categorical',  shuffle=**False**)  classifier.fit\_generator(training\_set,  steps\_per\_epoch=**8000**,  epochs=**5**, # i have changed nb\_epoch 1 to 5  validation\_data=test\_set,  validation\_steps=**2000**)  classifier.save\_weights('model\_weights.h5')  model\_json = classifier.to\_json()  **with** open("model.json", "w") **as** json\_file:  json\_file.write(model\_json)  print(training\_set.class\_indices)  print(classifier.summary)  messagebox.showinfo("Model Generated",  "CNN Model Generated on Train & Test Data. See black console for details")  **def** **classify**(self):  **global** filename  imagetest = image.load\_img(filename, target\_size=(**48**, **48**))  imagetest = image.img\_to\_array(imagetest)  imagetest = np.expand\_dims(imagetest, axis=**0**)  preds = loaded\_model.predict(imagetest)  print(Back.BLACK+str(preds) + " " + str(np.argmax(preds)))  predict = np.argmax(preds)  msg = ""  **if** predict == **0**:  msg = "Image Contains Car"  **if** predict == **1**:  msg = "Image Contains Cat"  **if** predict == **2**:  msg = "Image Contains Dog"  **if** predict == **3**:  msg = "Image Contains Person"  **if** predict == **4**:  msg = "Image Contains Plane"  imagedisplay = cv2.imread(filename)  orig = imagedisplay.copy()    output = imutils.resize(orig, width=**400**)  cv2.putText(output, msg, (**10**, **25**), cv2.FONT\_HERSHEY\_SIMPLEX, **0.7**, (**0**, **255**, **0**), **2**)    cv2.imshow("Predicted Image Result", output)  init()  print(Back.BLUE+str(dic\_labels))  print(Back.GREEN +"Probability of "+dic\_labels[predict]+" is high so we can say that : "+msg)  cv2.waitKey(**0**)  **def** **exit**(self):  self.main.destroy()  **def** **start\_program**(self):  font = ('times', **16**, 'bold')  title = Label(self.main, text='Image Classification Using Deep Learning CNN Algorithm', justify=LEFT)  title.config(bg='lavender blush', fg='DarkOrchid1')  title.config(font=font)  title.config(height=**3**, width=**120**)  title.place(x=**100**, y=**5**)  title.pack()  font1 = ('times', **14**, 'bold')  model = Button(self.main, text="Generate CNN Train & Test Model", command=self.generateModel)  model.place(x=**200**, y=**100**)  model.config(font=font1)  uploadimage = Button(self.main, text="Upload Test Image", command=self.upload)  uploadimage.place(x=**200**, y=**150**)  uploadimage.config(font=font1)  classifyimage = Button(self.main, text="Classify Picture In Image", command=self.classify)  classifyimage.place(x=**200**, y=**200**)  classifyimage.config(font=font1)  exitapp = Button(self.main, text="Exit", command=self.exit)  exitapp.place(x=**200**, y=**250**)  exitapp.config(font=font1)  self.main.config(bg='light coral')  self.main.mainloop()  **Run.py**   |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9 | **from ImageClassification import** **ImageClassificationSystem**  **class** **RunProgram**:  **def** **\_\_init\_\_**(self):  user = ImageClassificationSystem()  user.start\_program()  run = RunProgram() | |
|  |  |

### 7.2 TENSORFLOW CNN

In this project we are using python KERAS and Google TENSORFLOW CNN algorithm to classify images, CNN algorithm can predict images correctly up to 90% which is better prediction accuracy compare to all other algorithms such as SVM, KNN etc. To classify images using CNN we need to train CNN network with all possible images and when new images uploaded then CNN train model will be applied on this new image to predict or identify image.

To demonstrate how to build a convolutional neural network based image classifier, we shall build a 6 layer neural network that will identify and separate one image from other. This network that we shall build is a very small network that we can run on a CPU as well. Traditional neural networks that are very good at doing image classification have many more parameters and take a lot of time if trained on normal CPU. However, our objective is to show how to build a real-world convolutional neural network using TENSORFLOW.

Neural Networks are essentially mathematical models to solve an optimization problem. They are made of neurons, the basic computation unit of neural networks. A neuron takes an input (say x), do some computation on it (say: multiply it with a variable w and adds another variable b) to produce a value (say; z= wx + b). This value is passed to a non-linear function called activation function (f) to produce the final output (activation) of a neuron. There are many kinds of activation functions. One of the popular activation function is Sigmoid. The neuron which uses sigmoid function as an activation function will be called sigmoid neuron. Depending on the activation functions, neurons are named and there are many kinds of them like RELU, TanH.

#### 7.2.1 Relu Activation Function

An activation function is basically just a simple function that transforms its inputs into outputs that have a certain range. There are various types of activation functions that perform this task in a different manner, For example, the sigmoid activation function takes input and maps the resulting values in between 0 to 1.

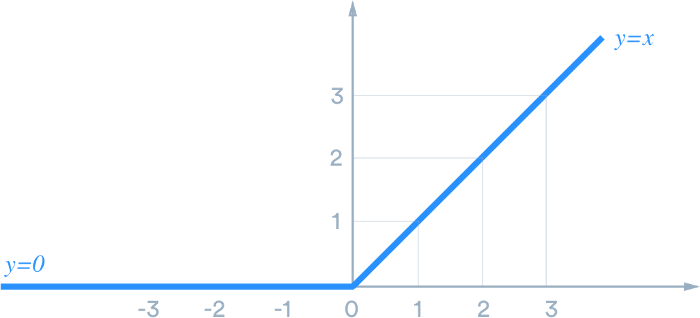
ReLU stands for rectified linear activation unit and is considered one of the few milestones in the deep learning revolution. It is simple yet really better than its predecessor activation functions such as sigmoid or tanh.

ReLU activation function formula

Now how does ReLU transform its input? It uses this simple formula:

F(x) = max (0, x)

ReLU function is its derivative both are monotonic. The function returns 0 if it receives any negative input, but for any positive value x, it returns that value back. Thus it gives an output that has a range from 0 to infinity.

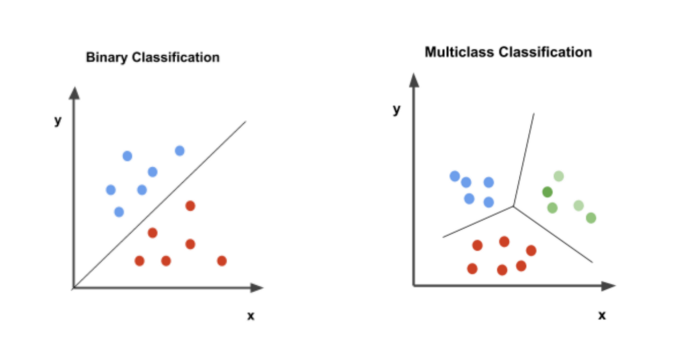


Now let us give some inputs to the ReLU activation function and see how it transforms them and then we will plot them also. ReLU is used as a default activation function and nowadays and it is the most commonly used activation function in neural networks, especially in CNNs.

#### 7.2.2 Softmax Function

The softmax function is an activation function that turns real values into probabilities. A softmax function is a generalization of the logistic function that can be used to classify multiple kinds of data. The softmax function takes in real values of different classes and returns a probability distribution.

Where the standard logistical function is capable of binary classification, the softmax function is able to do multiclass classification.



Fi: Binary Classification vs Multiclass Classification

To predict image class multiple layers operate on each other to get best match layer and this process continues till no more improvement left.

First to go the Folder in which the application located and then run the manage.py file in to run the application.

### 7.3 Screen Captures

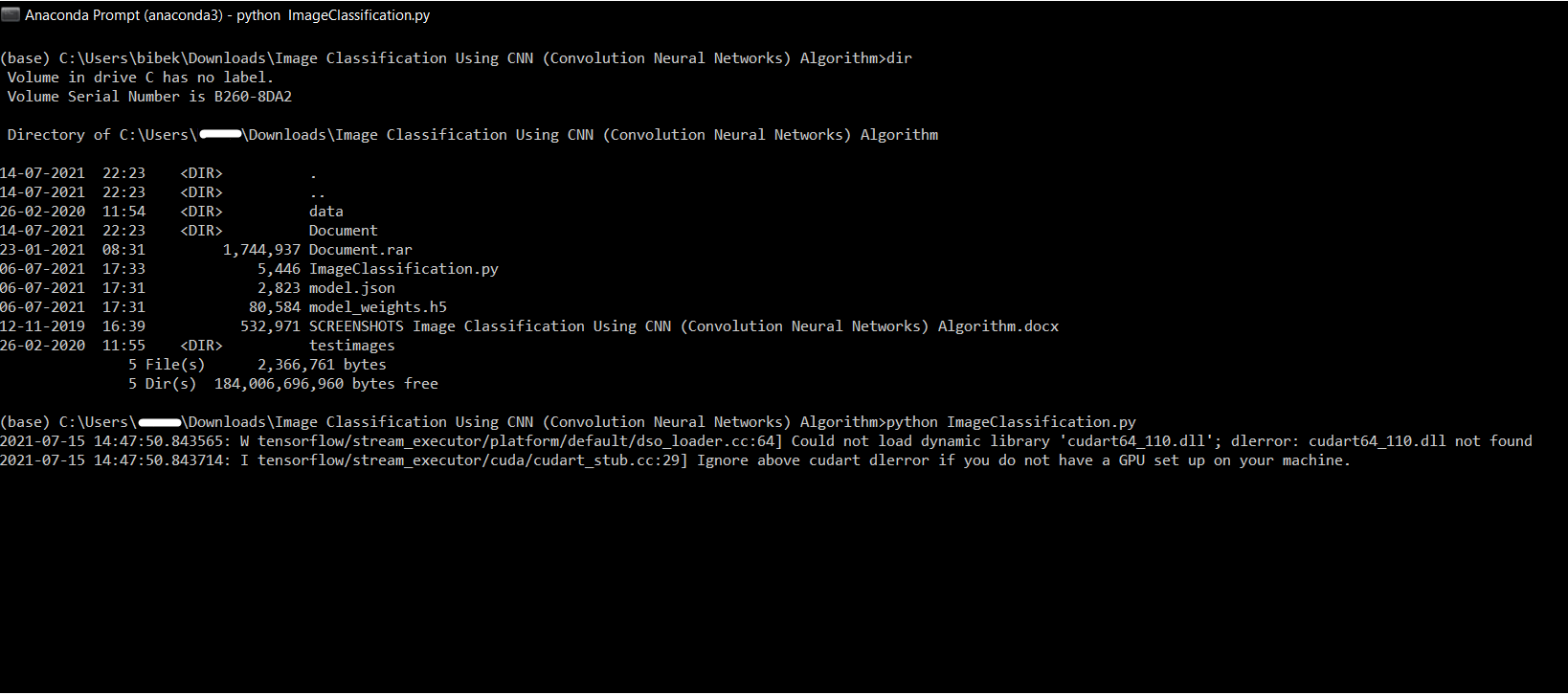


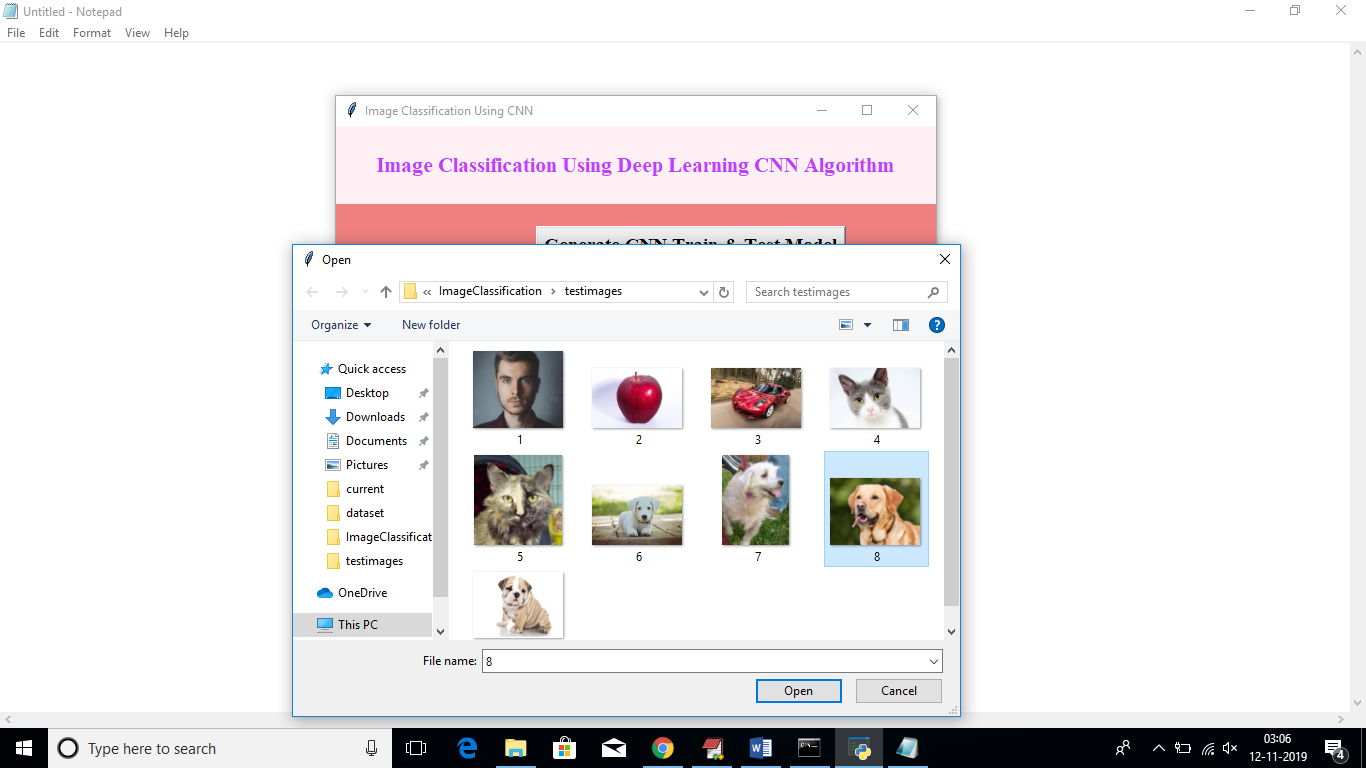
Fig- : Running ImageClassification.py program

The running procedure for the program in the anaconda command prompt. The python filename ImageClassification.py will be open from the given directory in the prompt.



Fig: GUI of Image Classification Using Deep Learning CNN Algorithm

In above screen click on ‘Generate CNN Train & Test Model’ to generate CNN model with all images given inside data folder. In this application I am building CNN model with CAR, CAT, DOG, PLANE and PERSON images. This application can identify or predict images up to 90% and this CNN can work up to 100% also but we need to train it with all possible images and high processing CPU. So in this normal CPU my generated model can identity above images with 90% accuracy. Now click on ‘Upload Test Image’ button to upload image



Fig; uploading the test images for the Image Classification Process

The some test image for the image classification to practice the accuracy for of the CNN algorithm.

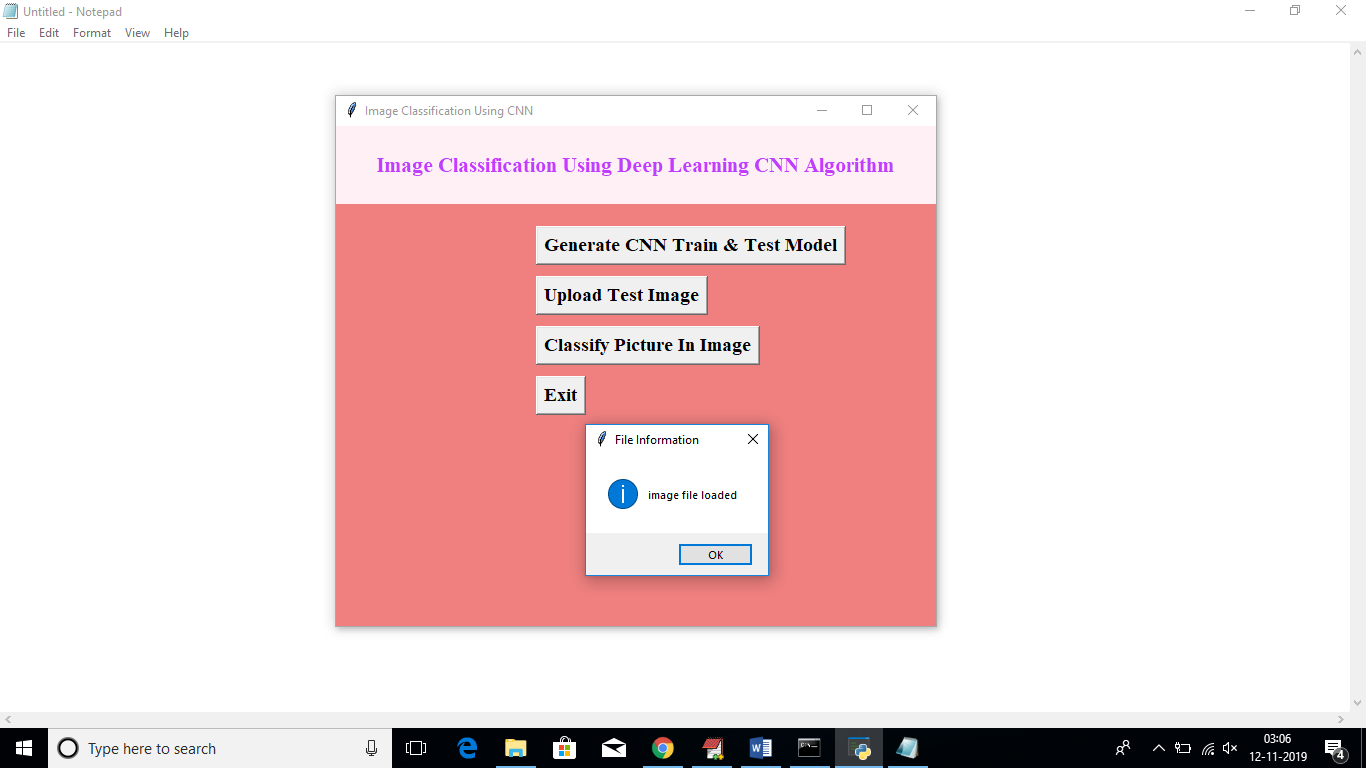


Fig: Successfully Image File Uploaded

Upload Test Image button will upload the test images for the classification of the image. The classification of image detail will be shown in below picture.

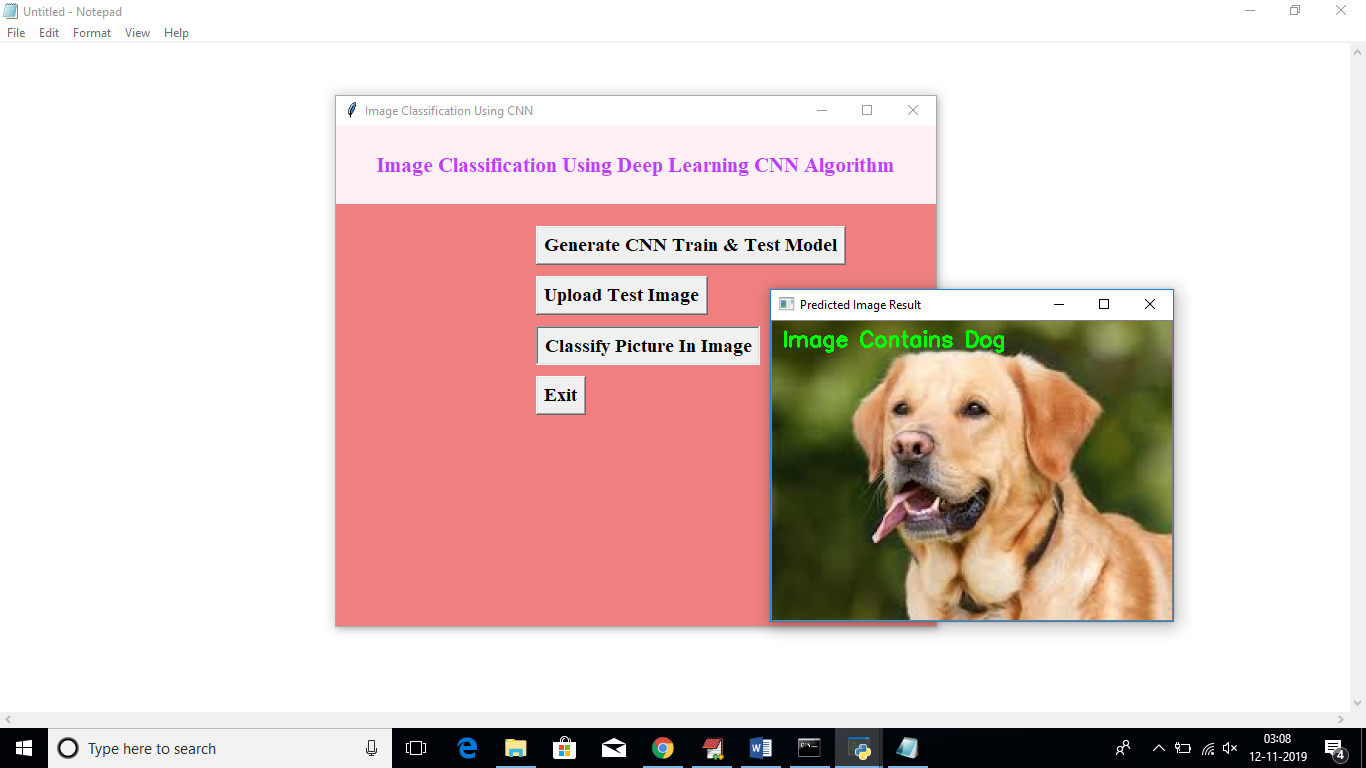


Fig: Successfully classify the image

After uploading image click on ‘Classify Picture in Image’ button to predict photo in uploaded image

In uploaded image in above screen we can see application classify image as Dog. Similarly u can test with other image

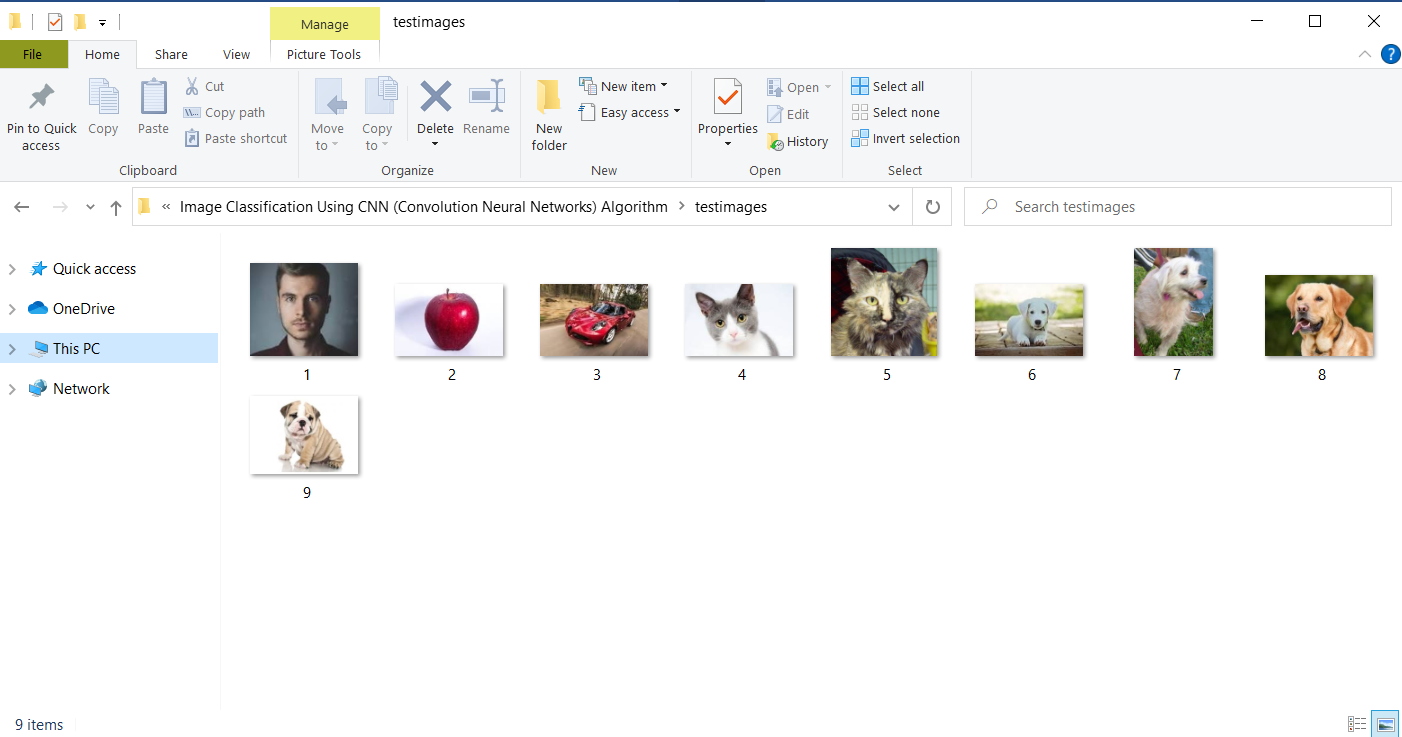


Fig- : Test Images

In above screen some object images are there but we don’t know its name or object name. So by uploading this image to application we can get their objects name.  So, by uploading any image we can know the name of bird. You can upload any image and get it name and uploading image name should be as integer value

Accuracy value of this algorithm you can see in below screen

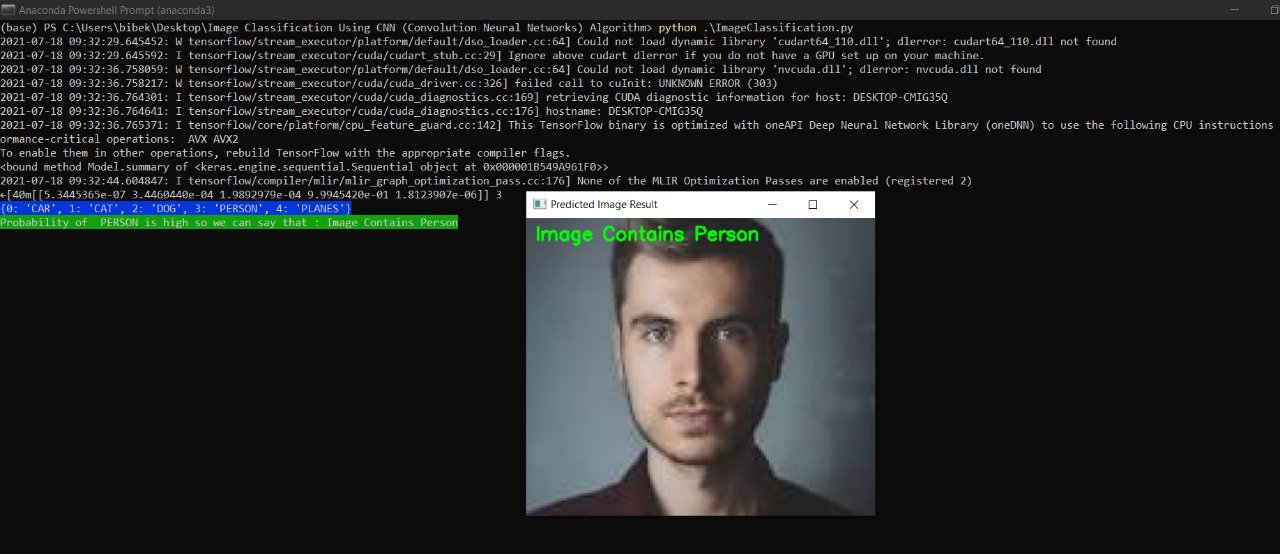


Fig- : Accuracy of the program

In above screen in selected text, you can see Accuracy value.

# CHAPTER 8

# SYSTEM TEST

### 8.1 Software Testing

Software testing is the process of validating and verifying that a software application meets the technical requirements which are involved in its design and development. It is also used to uncover any defects/bugs that exist in the application. It assures the quality of the software. There are many types of testing software viz., manual testing, unit testing, black box testing, performance testing, stress testing, regression testing, white box testing etc. Among these performance testing and load testing are the most important one for an android application and next sections deal with some of these types.

### 8.2 Black box Testing

Black box testing treats the software as a "black box"—without any knowledge of internal implementation. Black box testing methods include equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix, exploratory testing, and specification-based testing.

### 8.3 White box Testing

White box testing is when the tester has access to the internal data structures and algorithms including the code that implement these.

### 8.4 Performance Testing

Performance testing is executed to determine how fast a system or sub-system performs under a particular workload. It can also serve to validate and verify other quality attributes of the system such as scalability, reliability and resource usage.

### 8.5 Load Testing

Load testing is primarily concerned with testing that can continue to operate under specific load, whether that is large quantities of data or a large number of users.

### 8.6 Manual Testing

Manual Testing is the process of manually testing software for defects. Functionality of this application is manually tested to ensure the correctness. Few examples of test case for Manual Testing are discussed later in this chapter.

# CHAPTER 9

# CONCLUSION

### 9.1 Conclusion

In this work, I figured out what is deep learning. I assembled and trained the CNN model to classify photographs of cars and elephants. I have tested that this model works really well with a small number of photos. I measured how the accuracy depends on the number of epochs in order to detect potential overfitting problem. I determined that 10 epochs are enough for a successful training of the model.

My next step would be to try this model on more data sets and try to apply it to practical tasks. I would also like to experiment with the neural network design in order to see how a higher efficiency can be achieved in various problems.

### 9.2 Scope for future work

* Create an android/iOS app instead of website which will be more convenient to user.
* System can be implemented using cloud which can store large amount of data for comparison and provide high computing power for processing (in case of Neural Networks)

# REFERENCE

Code snippets for any errors <http://stackoverflow.com/>

Software testing <http://en.wikipedia.org/wiki/Software_testing>

Manual Testing <http://en.wikipedia.org/wiki/Manual_testing>

Performance testing <http://en.wikipedia.org/wiki/Software_performance_testing>

**Anaconda Installation**

<https://www.jcchouinard.com/install-python-with-anaconda-on-windows/>

Anaconda Version check

<https://mauridb.medium.com/how-to-check-your-anaconda-version-c092400c9978>

Relu activation

<https://www.mygreatlearning.com/blog/relu-activation-function/>

Keras

<https://keras.io/why_keras/>

Softmax

<https://towardsdatascience.com/what-is-the-softmax-function-teenager-explains-65495eb64338>

Pandas

<https://www.tutorialspoint.com/python_pandas/python_pandas_introduction.html>