VISVESVARAYA TECHNOLOGICAL UNIVERSITY
JNANASANGAMA, BELAGAVI - 590018

" IMAGE RECOGNITION USING NEURAL NETWORK & DEEP LEARNING"

Thesis submitted in partial fulfillment of the curriculum prescribed for the award of the degree of Bachelor of Engineering in Computer Science & Engineering by

1CR14CS138	Siddharth
1CR14CS163	Vivekanand Vivek
1CR14CS007	Akarsh Ramesh Khatagalli
1CR14CS134	Shreyans Maitrey

Under the Guidance of

Mr. Kartheek GCR
Assistant Professor
Department of CSE, CMRIT, Bengaluru

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
#132, AECS LAYOUT, IT PARK ROAD, BENGALURU - 560037

2017-18

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
JNANASANGAMA, BELAGAVI - 590018

Certificate

This is to certify that the project entitled **" IMAGE RECOGNITION USING NEURAL NETWORK & DEEP LEARNING"** is a bonafide work carried out by **Siddharth** , **Vivekanand Vivek** , **Akarsh Ramesh Khatagalli** and **Shreyans Maitrey** in partial fulfillment of the award of the degree of Bachelor of Engineering in Computer Science & Engineering of Visvesvaraya Technological University, Belgaum, during the year 2017-18. It is certified that all corrections / suggestions indicated during reviews have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the Bachelor of Engineering Degree.

----------	----------	----------
Signature of Guide	Signature of HoD	Signature of Principal
Mr. Kartheek GCR	**Dr. Jhansi Rani P**	**Dr. Sanjay Jain**
Assistant Professor	Professor & Head	Principal
Department of CSE	Department of CSE	CMRIT,
CMRIT, Bengaluru - 37	CMRIT, Bengaluru - 37	Bengaluru - 37

External Viva

	Name of the Examiners	Institution	Signature with Date
1.	----------	----------	----------
2.	----------	----------	----------

Acknowledgement

We take this opportunity to thank all of those who have generously helped us to give a proper shape to our work and complete our BE project successfully. A successful project is fruitful culmination efforts by many people, some directly involved and some others indirectly, by providing support and encouragement.

We would like to thank **Dr. SANJAY JAIN** , Principal , CMRIT , for providing excellent academic environment in the college.

We would like to express our gratitude towards **Dr. JHANSI RANI** , Professor & HOD , Dept of CSE , CMRIT , who provided guidance and gave valuable suggestions regarding the project.

We consider it a privilege and honour to express our sincere gratitude to our Internal Guide **Mr. KARTHEEK GCR** , Asst. Professor , Department of Computer Science & Engineering , CMRIT , for his valuable guidance throughout the tenure of this project work.

Siddharth
Vivekanand Vivek
Akarsh Ramesh Khatagalli
Shreyans Matriey

Table of Contents

List of Figures

List of Tables

Abstract

The main idea of the project is to be able to classify product category by looking only at the image, we think that it make sense to use Deep Convolutional Neural Networks. To begin with, we are planning to use transfer learning and fine-tuning on ResNet50, VGG-19, and some other pre-trained on ImageNet data networks that are accessible in Keras. For this step we are planning to use only small subset of the whole training data it should also represent only small subset of the whole classes that we need to classify data into (50-100 classes out of total 5000 classes).

We are planning to rely on transfer learning, because it seems to be very time consuming to train the Deep Convolutional Neural Network on such a large dataset from scratch, we will need tens of training epochs, each of them should run through at least some big subset of the initial train set and then update the weights of the network. Each of these epochs will take few days even on GPU, then to train from scratch we need tens of days on GPU which we dont have. Thus we decided to stop on transfer learning which have been reported as a very powerful way to deal with image classification problems. Because on the top level most object on images are built from the same lines, and corners we then just need to specify combinations of these lines and corners and their connection with the classes in the last layers of the pretrained networks. Next steps would involve training on a cluster because of the reasons stated in the background section. For this task some Keras functions would be very useful to us and would save some time from reimplementing functions for dealing with big training sets of data , because in this project we should be concerned about RAM usage.

As a final step we are planning to use the hierarchical structure of the target classification labels. Because we have 3 level tree of categories with roughly 50 nodes at the first layer, 50-100 children per node at 2nd and 3rd levels we can use it in a way that we first classify the image to one (or top 5) of the upper level categories, then based on this we can do 2nd level classification and 3rd (level of interest) classification and the idea is that because we are fixing some class on the upper level the lower level should have smaller number of choices and be able to better learn from smaller amounts of training data.

Chapter 1

PREAMBLE

1.1 INTRODUCTION

As the necessity for higher levels of security rises, technology is bound to swell to fulfill these needs. Any new creation, enterprise, or development should be uncomplicated and acceptable for end users in order to spread worldwide. This strong demand for user-friendly systems which can secure our assets and protect our privacy without losing our identity in a sea of numbers, grabbed the attention and studies of scientists toward whats called biometrics. There is a more scientific Mathematical Introduction For Face Recognition: Pixel Arithmetic for readers who are interested in the mathematical perspective and representation of pixels in face recognition applications. The link also contains a VB.NET implementation of the Pixel class.

Biometrics is the emerging area of bioengineering; it is the automated method of recognizing person based on a physiological or behavioral characteristic. There exist several biometric systems such as signature, finger prints, voice, iris, retina, hand geometry, ear geometry, and face. Among these systems, facial recognition appears to be one of the most universal, collectable, and accessible systems.

Biometric face recognition is a particularly attractive biometric approach, since it focuses on the same identifier that humans use primarily to distinguish one person from another: their faces. One of its main goals is the understanding of the complex human visual system and the knowledge of how humans represent faces in order to discriminate different identities with high accuracy.

The face recognition problem can be divided into two main stages: face verification (or authentication), and face identification (or recognition).

1.2 MOTIVATION

Face recognition has been a sought after problem of biometrics and it has a variety of applications in modern life. The problems of face recognition attracts researchers working in biometrics, patternrecognition eld and computer vision . Several face recognition algorithms are also used in many dierent applications apart from biometrics , such as video compressions , indexings etc. They can also be used to classify multimedia content, to allow fast and ecient searching for material that is of interest to the user. An ecent face recognition system can be of great help in forensic sciences, identication for law enforcement, surveillance , authentication for banking and security system, and giving preferential access to authorised users i.e. access control for secured areas etc. The problem of face recognition has gained even more importance after the recent increase in the terrorism related incidents. Use of face recognition for authentication also reduces the need of remembering passwords and can provide a much greater security if face recognition is used in combination with other security measures for access control. The cost of the license for an ecient commercial Face recognition system ranges from 30,000 $ to 150,000 $ which shows the signicant value of t he problem. Though face recognition is considered to be a very crucial authentication system but even after two decades continuous research and evolution of many face recognition algorithms , a truely robust and ecient system that can produce good results in realtime and normal conditions is still not available. The Face Recognition Vendor Test (FRVT) that has been conducted by the National Institute of Standards and Technology (NIST), USA, has shown that the commercial face recognition systems do not perform well under the normal daily conditions. Some of the latest face recognition algorithm involving machine learning tools perform well but sadly the training period and processing time is large enough to limit its use in practical applications. Hence there is a continuous strife to propose an eective face recognition system with high accuracy and acceptable processing time.

1.3 FACE DETECTION APPROACHES

Some of the main face detection methods are discussed here. 1) Knowledge based methods are developed on the rules derived from the researchers knowledge of human faces. Problem in this approach is the diculty in translating human knowledge into welldened rules. 2) Featured-based methods: Invariant features of faces are used for detecting texture, skin color. But features from such algorithm can be severely corrupted due to illumination, noise and occlusion. 3) Template matching: Input image is compared with predened face template. But the performance here suers due to variations in scale, pose and shape. 4) Appearance-based method: In template

matching methods, the templates are predened by experts. Whereas, the templates in appearance based methods are learned from examples in images. Statistical analysis and machine learning techniques can be used to nd the relevant characteristics of face and non-face images.

Chapter 2

LITERATURE SURVEY

2.1 INTRODUCTION

Literature survey is mainly carried out in order to analyze the background of the current project which helps to find out flaws in the existing system and guides on which unsolved problems we can work out. So, the following topics not only illustrate the background of the project but also uncover the problems and flaws which motivated to propose solutions and work on this project.

2.2 LITERATURE SURVEY

Literature survey is the documentation of a comprehensive review of the published and unpublished work from secondary sources data in the areas of specific interest to the researcher. The library is a rich storage base for secondary data and researchers used to spend several weeks and sometimes months going through books, journals, newspapers, magazines, conference proceedings, doctoral dissertations, master's theses, government publications and financial reports to find information on their research topic. Reviewing the literature on the topic area at this time helps the research er to focus further interviews more meaningfully on certain aspects found to be important is the published studies even if these had not surfaced during the earlier questioning .So the literature survey is important for gathering the secondary data for the research which might be proved very helpful in the research. The literature survey can be conducted for several reasons. The literature review can be in any area of the business.

2.3 PAPER 1

In FFNN, the neurons are connected in a directed way having clear start and stop place i.e., the input layer and the output layer. The layer between these two layers, are called as the hidden layers. Learning occurs through adjustment of weights and the aim is to try and minimize error between the output obtained from the output layer and the input that goes into the input layer. The weights are adjusted by process of back propagation (in which the partial derivative of the error with respect to last layer of weights is calculated). The process of weight adjustment is repeated in a recursive manner until weight layer connected to input layer is updated.

2.4 PAPER 2

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such asthe brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well.

2.5 PAPER 3

We propose a deep convolutional neural network architecture codenamed Inception that achieves the new state of the art for classification and detection in the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14). The main hallmark of this architecture is the improved utilization of the computing resources inside the network. By a carefully crafted design, we increased the depth and width of the network while keeping the computational budget constant. To optimize quality, the architectural decisions were based on the Hebbian principle and the intuition of multi-scale processing. One particular incarnation used in our submission for ILSVRC14 is called GoogLeNet, a 22 layers deep network, the quality of which is assessed in the context of classification and detection.

Chapter 3

THEORETICAL BACKGROUND

3.1 INTRODUCTION

The process of image classification involves two steps, training of the system followed by testing. The training process means, to take the characteristic properties of the images and form a unique description for a particular class. The process is done for all classes depending on the type of classification problem; binary classification or multi-class classification. The testing step means to categorize the test images under various classes for which system was trained. This assigning of class is done based on the partitioning between classes based on the training features. Since 2006, deep structured learning, or more commonly called deep learning or hierarchical learning, has emerged as a new area of machine learning research . Several definitions are available for Deep Learning; coating one of the many definitions from Deep Learning is defined as: A class of machine learning techniques that exploit many layers of nonlinear information processing for supervised or unsupervised feature extraction and transformation and for pattern analysis and classification. This work aims at the application of Convolutional Neural Network or CNN for image classification. The image data used for testing the algorithm includes remote sensing data of aerial images and scene data from SUN database. The rest of the paper is organized as follows. Section 2 deals with the working of the network followed by section 2.1 with theoretical background. The working of CNN gives the experimental procedure in detail.

3.2 PERCEPTION

Perceptrons were developed in the 1950s and 1960s by the scientist Frank Rosenblatt, inspired by earlier work by Warren McCulloch and Walter Pitts. Today, it's more common to use other models of artificial neurons - in this book, and in much modern work on neural networks, the main neuron model used is one called the sigmoid neuron. We'll get to sigmoid neurons shortly. But to understand why sigmoid neurons are defined the way they are, it's worth taking the time to first understand perceptrons. So how do perceptrons work? A perceptron takes several binary inputs, x1,x2,, and produces a single binary output:

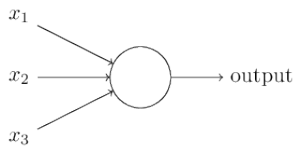

Figure 3.1: Neurons Output

3.3 SIGMOID NEURONS

Suppose we have a network of perceptrons that we'd like to use to learn to solve some problem. For example, the inputs to the network might be the raw pixel data from a dataset, image of a face. And we'd like the network to learn weights and biases so that the output from the network correctly classifies the Face. To see how learning might work, suppose we make a small change in some weight (or bias) in the network. What we'd like is for this small change in weight to cause only a small corresponding change in the output from the network. As we'll see in a moment, this property will make learning possible. Schematically, here's what we want :

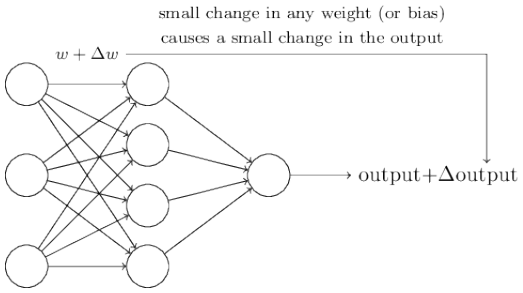

Figure 3.2: Sigmoid neurons

In fact, the exact form of isn't so important - what really matters is the shape of the function when plotted. Here's the shape:

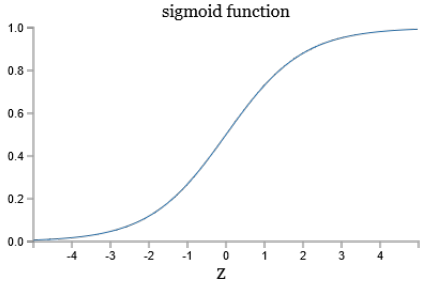

Figure 3.3: Sigmoid Fuction

This shape is a smoothed out version of a step function:

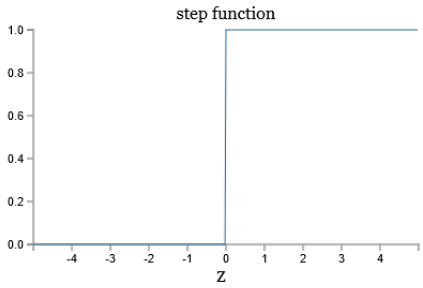

Figure 3.4: Step Fuction

3.4 THE ARCHITECTURE OF NEURAL NETWORKS

In preparation for that, it helps to explain some terminology that lets us name different parts of a network. Suppose we have the network:

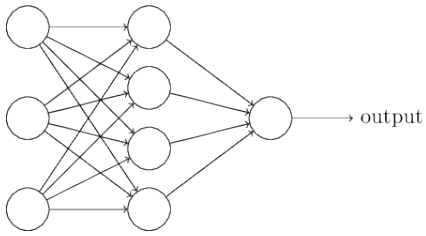

Figure 3.5: Nuron

As mentioned earlier, the leftmost layer in this network is called the input layer, and the neurons within the layer are called input neurons. The rightmost or output layer contains the output neurons, or, as in this case, a single output neuron. The middle layer is called a hidden layer, since the neurons in this layer are neither inputs nor outputs. The term "hidden" perhaps sounds a little mysterious - the first time I heard the term I thought it must have some deep philosophical or mathematical significance - but it really means nothing more than "not an input or an output". The network above has just a single hidden layer, but some networks have multiple hidden layers. For example, the following four-layer network has two hidden layers:

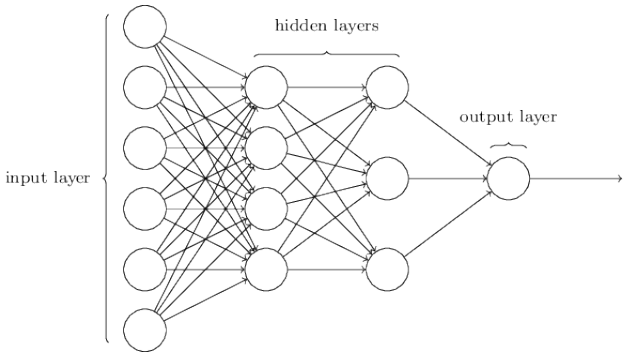

Figure 3.6: Nuron

3.5 A SIMPLE NETWORK TO CLASSIFY FACES

We'll focus on writing a program to solve the problem, that is, classifying individual Faces. We do this because it turns out that the segmentation problem is not so difficult to solve, once you have a good way of classifying Faces. There are many approaches to solving the segmentation problem. One approach is to trial many different ways of segmenting the image, using the individual Face classifier to score each trial segmentation. A trial segmentation gets a high score if the individual Face classifier is confident of its classification in all segments, and a low score if the classifier is having a lot of trouble in one or more segments. The idea is that if the classifier is having trouble somewhere, then it's probably having trouble because the segmentation has been chosen incorrectly. This idea and other variations can be used to solve the segmentation problem quite well. So instead of worrying about segmentation we'll concentrate on developing a neural network which can solve the more interesting and difficult problem, namely, recognizing individual Faces. To recognize individual Faces we will use a three-layer neural network:

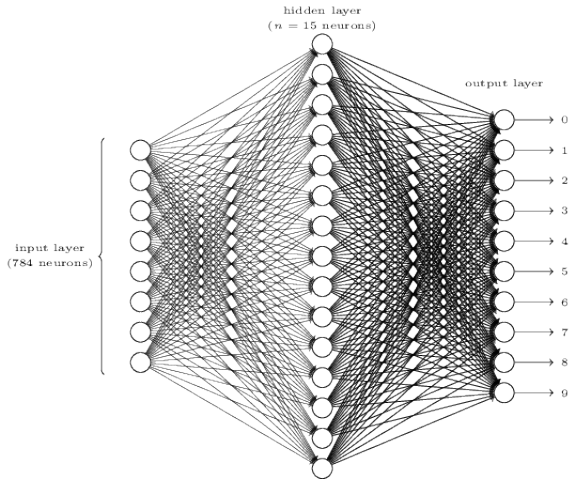

Figure 3.7: Used Network

3.6 DEEP NEURAL NETWORK

Deep Neural Networks are distinguished from the more commonplace single-hidden-layer neural networks by their depth; that is, the number of node layers through which data passes in a multistep process of pattern recognition.

Earlier versions of neural networks such as the first perceptrons were shallow, composed of one input and one output layer, and at most one hidden layer in between. More than three layers (including input and output) qualifies as deep learning. So deep is a strictly defined, technical term that means more than one hidden layer.

In deep neural networks, each layer of nodes trains on a distinct set of features based on the previous layers output. The further you advance into the neural net, the more complex the features your nodes can recognize, since they aggregate and recombine features from the previous layer.

This is known as feature hierarchy, and it is a hierarchy of increasing complexity and abstraction. It makes deep-learning networks capable of handling very large, high-dimensional data sets with billions of parameters that pass through nonlinear functions.

Above all, these nets are capable of discovering latent structures within unlabeled, unstructured data, which is the vast majority of data in the world. Another word for unstructured data is raw media; i.e. pictures, texts, video and audio recordings. Therefore, one of the problems deep learning solves best is in processing and clustering the worlds raw, unlabeled media, discerning similarities and anomalies in data that no human has organized in a relational database or ever put a name to.

3.7 CONVOLUTIONAL NEURAL NETWORK

Computational models of neural networks have been around for a long time, first model proposed was by McCulloch and Pitts. Neural networks are made up of a number of layers with each layer connected to the other layers forming the network. A feed-forward neural network or FFNN can be thought of in terms of neural activation and the strength of the connections between each pair of neurons.

In FFNN, the neurons are connected in a directed way having clear start and stop place i.e., the input layer and the output layer. The layer between these two layers, are called as the hidden layers. Learning occurs through adjustment of weights and the aim is to try and minimize error between the output obtained from the output layer and the input that goes into the input layer. The weights are adjusted by process of back propagation.

The process of weight adjustment is repeated in a recursive manner until weight layer connected to input layer is updated.

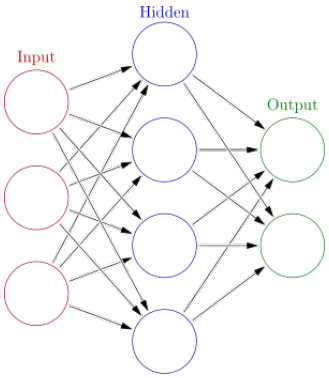

Figure 3.8: Typical Neural Network

3.7.1 WHY ARE WE USING CONVOLUTIONAL NEURAL NETWORK ?

Convolutional Neural Networks (CNN) is variants of MultiLayer Perceptron which are inspired from biology. These filters are local in input space and are thus better suited to exploit the strong spatially local correlation present in natural images. Convolutional neural networks are designed to process two dimensional image . A CNN architecture used in this project is that defined in . The network consists of three types of layers namely convolution layer, sub sampling layer and the output layer.

Chapter 4

SYSTEM REQUIREMENT SPECIFICATION

4.1 INTRODUCTION

This chapter describes about the requirements. It specifies the hardware and software requirements that are required in order to run the application properly. The Software Requirement Specification (SRS) is explained in detail, which includes overview of dissertation as well as the functional and non-functional requirement of this dissertation.

A SRS document describes all data, functional and behavioral requirements of the software under production or development. SRS is a fundamental document, which forms the foundation of the software development process. Its the complete description of the behavior of a system to be developed. It not only lists the requirements of a system but also has a description of its major feature. Requirement Analysis in system engineering and software engineering encompasses those tasks that go into determining the need or conditions to meet for a new or altered product, taking account of the possibly conflicting requirements of the various stakeholders, such as beneficiaries or users. Requirement Analysis is critical to the success to a development project. Requirement must be documented, measurable, testable, related to in identified business needs or opportunities, and defined to a level of detail sufficient for system design.

The SRS functions as a blueprint for completing a project. The SRS is often referred to as the "parent" document because all subsequent project management documents, such as design specifications, statements of work, software architecture specification, testing and validation plans, and documentation plans, are related to it. It is important to note that an SRS contains functional and non-functional requirements only.

Thus the goal of preparing the SRS document is to

- To facilitate communication between the customer, analyst, system developers, maintainers.
- To serve as a contrast between purchaser and supplier.
- To firm foundation for the design phase.
- Support system testing facilities.
- Support project management and control.
- Controlling the evolution of the system.

4.2 FUNCTIONAL REQUIREMENTS

Functional Requirement defines a function of a software system and how the system must behave when presented with specific inputs or conditions. These may include calculations, data manipulation and processing and other specific functionality. In this system following are the functional requirements:-

- Input test case must not have compilation and runtime errors.
- The application must not stop working when kept running for even a long time.
- The application must function as expected for every set of test cases provided.
- The application should generate the output for given input test case and input parameters.
- The application should generate on-demand services.

4.3 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements are the requirements which are not directly concerned with the specific function delivered by the system. They specify the criteria that can be used to judge the operation of a system rather than specific behaviors. They may relate to emergent system properties such as reliability, response time and store occupancy. Non-functional requirements arise through the user needs, because of budget constraints, organizational policies, the need for interoperability with other software and hardware systems or because of external factors such as:-

- Product Requirements
- Organizational Requirements
- User Requirements
- Basic Operational Requirements

In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours. This should be contrasted with functional requirements that define specific behaviour or functions. The plan for implementing non-functional requirements is detailed in the system architecture. Broadly, functional requirements define what a system is supposed to do and non- functional requirements define how a system is supposed to be. Functional requirements are usually in the

form of system shall do requirement, an individual action of part of the system, perhaps explicitly in the sense of a mathematical function, a black box description input, output, process and control functional model or IPO Model. In contrast, non-functional requirements are in the form of system shall be requirement, an overall property of the system as a whole or of a particular aspect and not a specific function. The systems' overall properties commonly mark the difference between whether the development project has succeeded or failed. Non-functional requirements of our project include:

- Response time The time the system takes to load and the time for responses on any action the user does.
- Processing time - How long is acceptable to perform key functions or export / import data?
- Throughput The number of transactions the system needs to handle must be kept in mind.
- Storage - The amount of data to be stored for the system to function.
- Growth Requirements As the system grows it will need more storage space to keep up with the efficiency.
- Locations of operation - Geographic location, connection requirements and the restrictions of a local network prevail.
- Architectural Standards The standards needed for the system to work and sustain.

4.3.1 PRODUCT REQUIREMENTS

- Portability: Since the SLR system is designed to run using Python (whose library is written in C), the system is portable.
- Correctness: It follows a well-defined set of procedures and rules to compute and also rigorous testing is performed to confirm the correctness of the data.
- Ease of Use: The front end is designed in such a way that it provides an interface which allows the user to interact in an easy manner.
- Modularity: The complete product is broken up into many modules and well-defined interfaces are developed to explore the benefit of flexibility of the product.

- Robustness: This software is being developed in such a way that the overall performance is optimized and the user can expect the results within a limited time with utmost relevancy and correctness.

whereas evolution quality involves testability, maintainability, extensibility or scalability.

4.3.1.1 ORGANIZATIONAL REQUIREMENTS

Process Standards: IEEE standards are used to develop the application which is the standard used by the most of the standard software developers all over the world. Design Methods: Design is one of the important stages in the software engineering process. This stage is the first step in moving from problem to the solution domain. In other words, starting with what is needed design takes us to work how to satisfy the needs.

4.3.1.2 USER REQUIREMENTS

The user requirements document (URD) or user requirements specification is a document usually used to software engineering that specifies the requirements the user expects from software to be constructed in a software project. Once the required information is completely gathered it is documented in a URD, which is meant to spell out exactly what the software must do and becomes part of the contractual agreement. A customer cannot demand feature not in the URD, whilst the developer cannot claim the product is ready if it does not meet an item of the URD. The URD can be used as a guide to planning cost, timetables, milestones, testing etc. The explicit nature of the URD allows customers to show it to various stakeholders to make sure all necessary features are described. Formulating a URD requires negotiation to determine what is technically and economically feasible. Preparing a URD is one of those skills that lies between a science and economically feasible. Preparing a URD is one of those skills that lies between a science and an art, requiring both software technical skills and interpersonal skills.

4.3.1.3 BASIC OPERATIONAL REQUIREMENTS

Operational requirement is the process of linking strategic goals and objectives to tactic goals and objectives. It describes milestones, conditions for success and explains how, or what portion of, a strategic plan will be put into operation during a given operational period, in the case of, a strategic plan will be put into operation during a given operational period, in the case of commercial application, a fiscal year or another given budgetary term. An operational plan is the basis for, and justification

of an annual operating budget request. Therefore, a five-year strategic plan would typically require five operational plans funded by five operating budgets. Operational plans should establish the activities and budgets for each part of the organization for the next 1-3 years. They link the strategic plan with the activities the organization will deliver and the resources required to deliver them. An operational plan draws directly from agency and program strategic plans to describe agency and program missions and goals, program objectives, and program activities. Like a strategic plan, an operational plan addresses four questions:

- Where are we now?
- Where do we want to be?
- How do we get there?

The customers are those that perform the eight primary functions of systems engineering, with special emphasis on the operator as the key customer. Operational requirements will define the basic need and, at a minimum, will be related to these following points:

- Mission profile or scenario: It describes about the procedures used to accomplish mission objective. It also finds out the effectiveness or efficiency of the system.
- Performance and related parameters: It points out the critical system parameters to accomplish the mission
- Utilization environments: It gives a brief outline of system usage. Finds out appropriate environments for effective system operation.
- Operational life cycle: It defines the system lifetime

4.4 HARDWARE REQUIREMENTS

System Requirement for training the Model:

- Operating System : Linux or MacOS
- RAM : 16GB Minimum.
- Internal storage : 500 MB.
- GPU : Nvidia
- VRAM : 6GB

System Requirement for Recognition using Trained Model:

- Operating System : Linux or MacOS
- RAM : 8GB Minimum.
- Internal storage : 500 MB.
- GPU : IntelHD, Nvidia, AMD
- VRAM : 1.5GB

4.5 SOFTWARE REQUIREMENTS

System Requirement for training the Model:

- Operating System : Linux or MacOS
- Coding Language : Python 2.7
- Tools : PyCharm
- Library : DLib, Keras, Open CV, numpy, Tinker.

System Requirement for Recognition using Trained Model:

- Operating System : Linux or MacOS
- Coding Language : Python 2.7
- Tools : PyCharm
- Library : Keras,Open CV,numpy,Tinker.

4.6 SOFTWARE QUALITY ATTRIBUTES

- **Functionality:** the capability of the software to provide functions which meet stated and implied needs when the software is used under specified conditions.
- **Reliability:** the capability of the software to maintain its level of performance when used under specified conditions.
- **Usability:** the capability of the software to be understood, learned, used and liked by the user, when used under specified conditions.
- **Efficiency:** the capability of the software to provide the required performance, relative to the amount of resources used, under stated conditions.

- **Maintainability:** the capability of the software to be modified. Modifications may include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specifications.
- **Portability:** the capability of software to be transferred from one environment to another.

Chapter 5

SYSTEM ANALYSIS

5.1 INTRODUCTION

Design is a meaningful engineering representation of something that is to be built. It is the most crucial phase in the developments of a system. Software design is a process through which the requirements are translated into a representation of software. Design is a place where design is fostered in software Engineering. Based on the user requirements and the detailed analysis of the existing system, the new system must be designed. This is the phase of system designing. Design is the perfect way to accurately translate a customers requirement in the finished software product. Design creates a representation or model, provides details about software data structure, architecture, interfaces and components that are necessary to implement a system. The logical system design arrived at as a result of systems analysis is converted into physical system design.

5.2 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

- Operational Feasibility
- Economical Feasibility
- Technical Feasibility
- Social Feasibility

Chapter 6

SYSTEM DESIGN

6.1 INTRODUCTION

Design is a meaningful engineering representation of something that is to be built. It is the most crucial phase in the developments of a system. Software design is a process through which the requirements are translated into a representation of software. Design is a place where design is fostered in software Engineering. Based on the user requirements and the detailed analysis of the existing system, the new system must be designed. This is the phase of system designing. Design is the perfect way to accurately translate a customers requirement in the finished software product. Design creates a representation or model, provides details about software data structure, architecture, interfaces and components that are necessary to implement a system. The logical system design arrived at as a result of systems analysis is converted into physical system design.

6.2 SYSTEM DEVELOPMENT METHODOLOGY

System development method is a process through which a product will get completed or a product gets rid from any problem. Software development process is described as a number of phases, procedures and steps that gives the complete software. It follows series of steps which is used for product progress. The development method followed in this project is waterfall model.

6.2.1 MODEL PHASES

The waterfall model is a sequential software development process, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Requirement initiation, Analysis, Design, Implementation, Testing and maintenance.

- **Requirement Analysis:** This phase is concerned about collection of requirement of the system. This process involves generating document and requirement review.
- **System Design:** Keeping the requirements in mind the system specifications are translated in to a software representation. In this phase the designer emphasizes on:-algorithm, data structure, software architecture etc.
- **Coding:** In this phase programmer starts his coding in order to give a full sketch of product. In other words system specifications are only converted in to machine readable compute code.

- **Implementation:** The implementation phase involves the actual coding or programming of the software. The output of this phase is typically the library, executables, user manuals and additional software documentation
- **Testing:** In this phase all programs (models) are integrated and tested to ensure that the complete system meets the software requirements. The testing is concerned with verification and validation.
- **Maintenance:** The maintenance phase is the longest phase in which the software is updated to fulfill the changing customer need, adapt to accommodate change in the external environment, correct errors and oversights previously undetected in the testing phase, enhance the efficiency of the software.

6.2.2 REASON FOR CHOOSING WATERFALL MODEL AS DEVELOPMENT METHOD

- Clear project objectives.
- Stable project requirements.
- Progress of system is measurable.
- Strict sign-off requirements.
- Helps you to be perfect.
- Logic of software development is clearly understood.
- Production of a formal specification
- Better resource allocation.
- Improves quality. The emphasis on requirements and design before writing a single line of code ensures minimal wastage of time and effort and reduces the risk of schedule slippage.
- Less human resources required as once one phase is finished those people can start working on to the next phase.

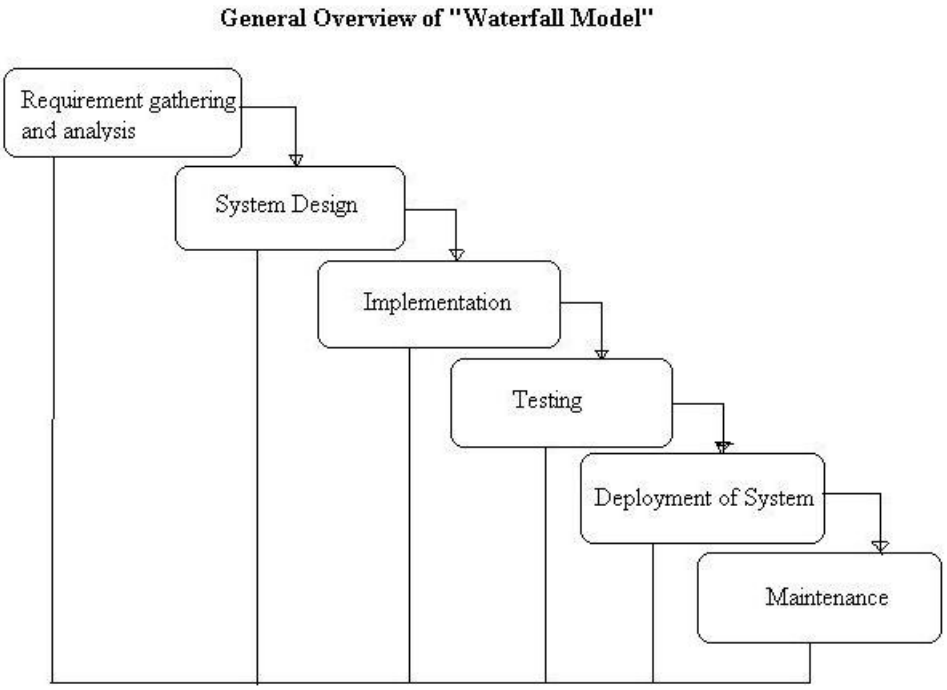

Figure 6.1: Waterfall Model

6.3 DESIGN USING UML

Designing UML diagram specifies, how the process within the system communicates along with how the objects with in the process collaborate using both static as well as dynamic UML diagrams since in this ever-changing world of Object Oriented application development, it has been getting harder and harder to develop and manage high quality applications in reasonable amount of time. As a result of this challenge and the need for a universal object modeling language every one could use, the Unified Modeling Language (UML) is the Information industries version of blue print. It is a method for describing the systems architecture in detail. Easier to build or maintains system, and to ensure that the system will hold up to the requirement changes.

Line	Symbol
Association	AssociationName
Aggregation	
Generalization	
Dependency	
Activity edge	
Event, transition	event[guard]/action
Link	LinkName
Composition	
Realization	
Assembly connection	
Message	some code
Control flow	

Table 6.1: Symbols Used In UML

6.4 DATA FLOW DIAGRAM

A data flow diagram (DFD) is graphic representation of the "flow" of data through an information system. A data flow diagram can also be used for the visualization of data processing (structured design). It is common practice for a designer to draw a context-level DFD first which shows the interaction between the system and outside entities. DFDs show the flow of data from external entities into the system, how the data moves from one process to another, as well as its logical storage. There are only four symbols: 1. Squares representing external entities, which are sources and destinations of information entering and leaving the system. 2. Rounded rectangles representing processes, in other methodologies, may be called 'Activities', 'Actions', 'Procedures', 'Subsystems' etc. which take data as input, do processing to it, and output it. 3. Arrows representing the data flows, which can either, be electronic data or physical items. It is impossible for data to flow from data store to data store except via a process, and external entities are not allowed to access data stores directly. 4. The flat three-sided rectangle is representing data stores should both receive information for storing and provide it for further processing.

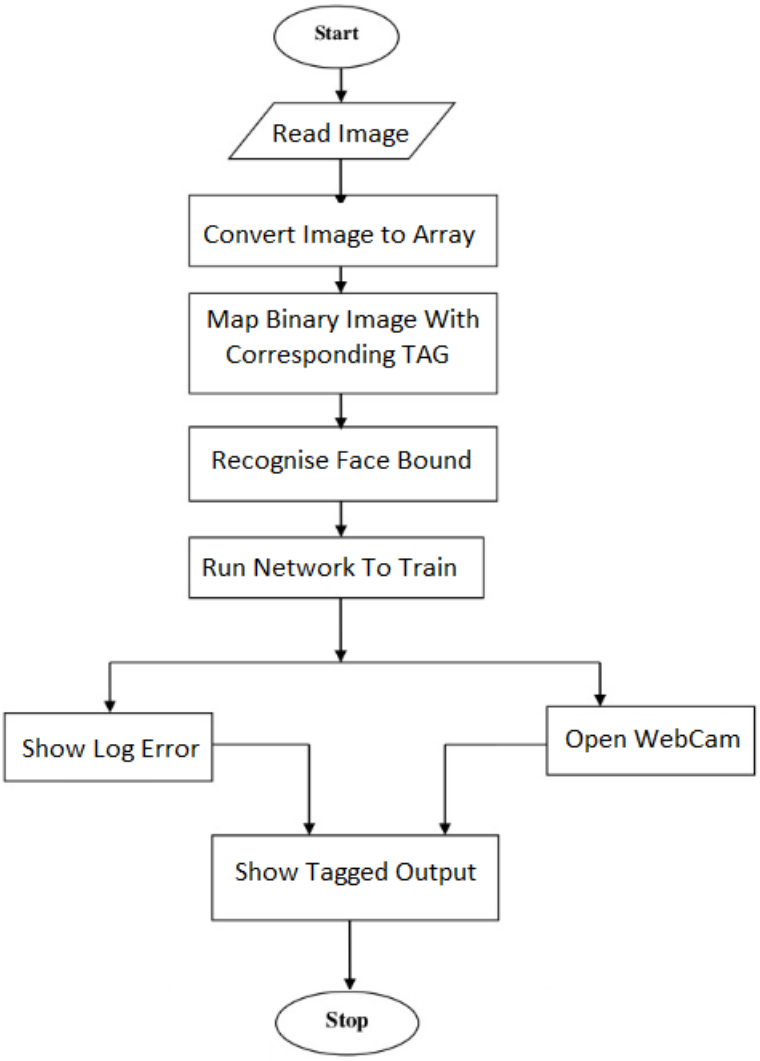

Figure 6.2: Dataflow Diagram

6.5 USE CASE DIAGRAM

A use case defines a goal-oriented set of interactions between external entities and the system under consideration. The external entities which interact with the system are its actors. A set of use cases describe the complete functionality of the system at a particular level of detail and it can be graphically denoted by the use case diagram.

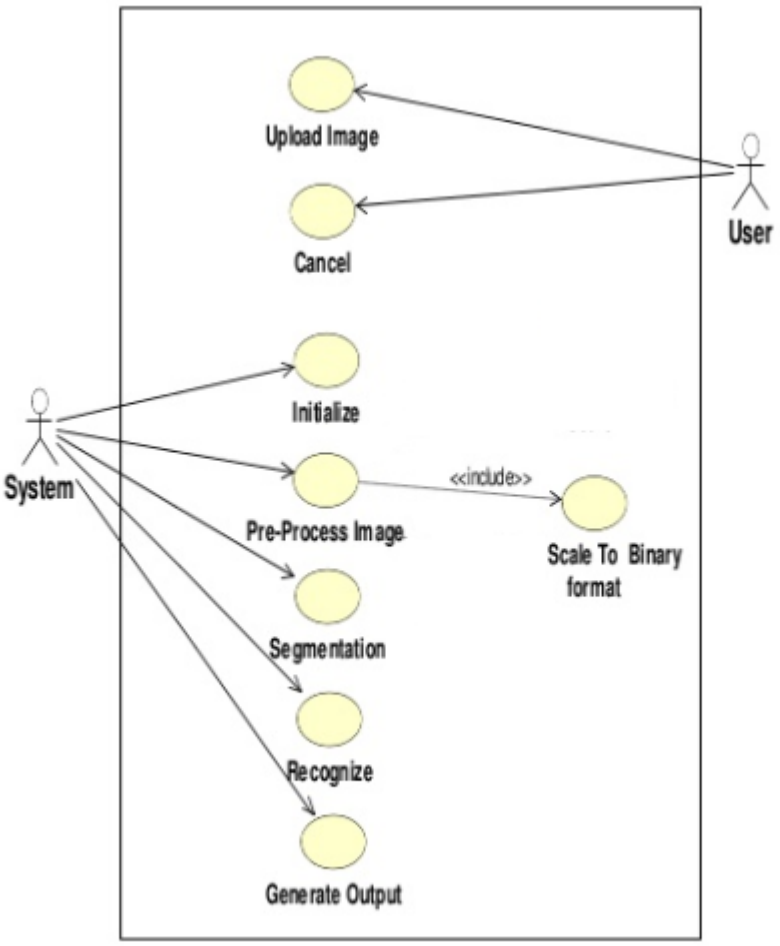

Figure 6.3: Usecase Diagram

6.6 ACTIVITY DIAGRAM

An activity diagram shows the sequence of steps that make up a complex process. An activity is shown as a round box containing the name of the operation. An outgoing solid arrow attached to the end of the activity symbol indicates a transition triggered by the completion.

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system.

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part.

It does not show any message flow from one activity to another. Activity diagram is sometimes considered as the flowchart. Although the diagrams look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single.

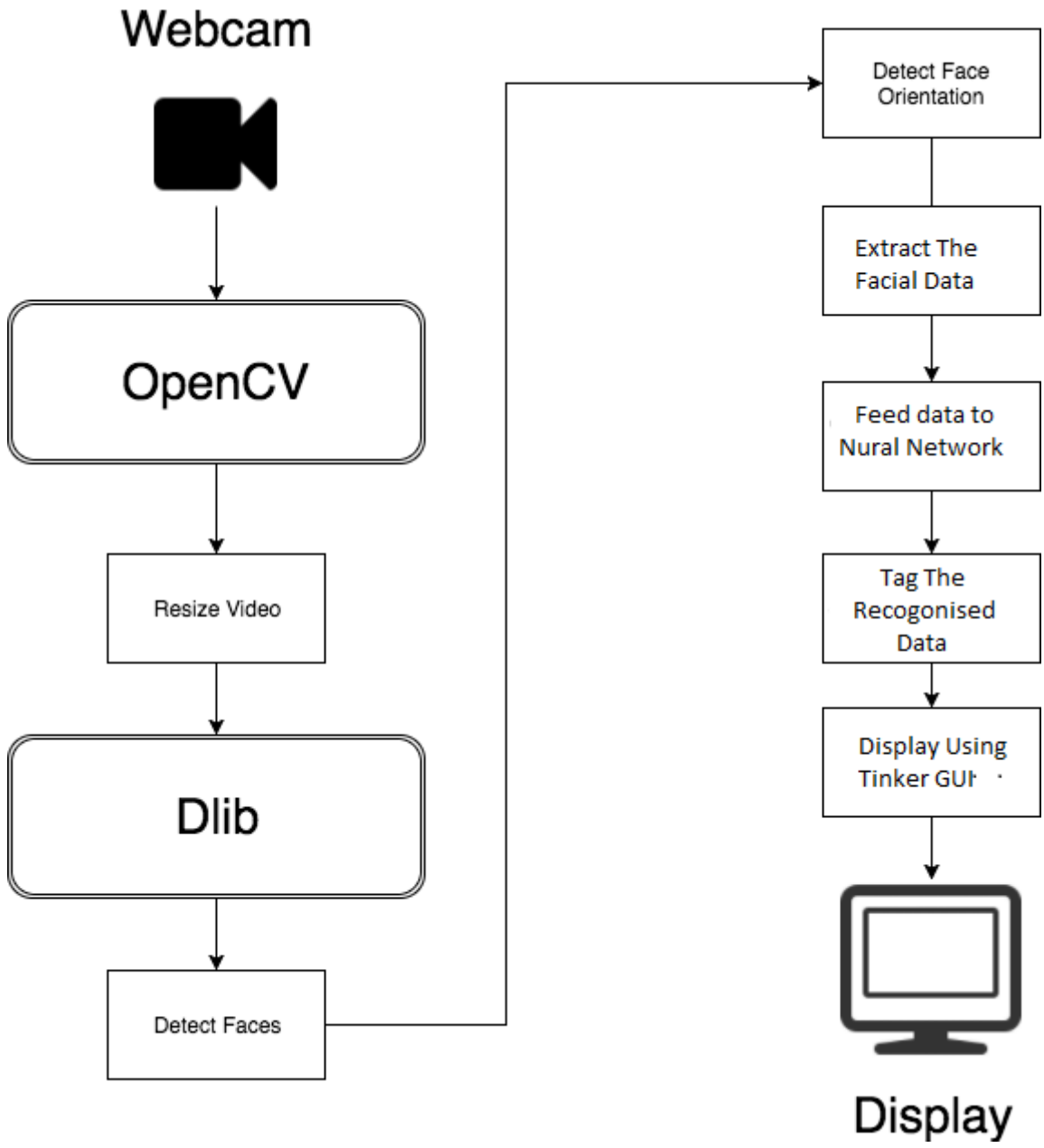

Figure 6.4: Activity Diagram

6.7 SEQUENCE DIAGRAM

Sequence diagram are an easy and intuitive way of describing the behavior of a system by viewing the interaction between the system and the environment. A sequence diagram shows an interaction arranged in a time sequence. A sequence diagram has two dimensions: vertical dimension represents time, the horizontal dimension represents the objects existence during the interaction.

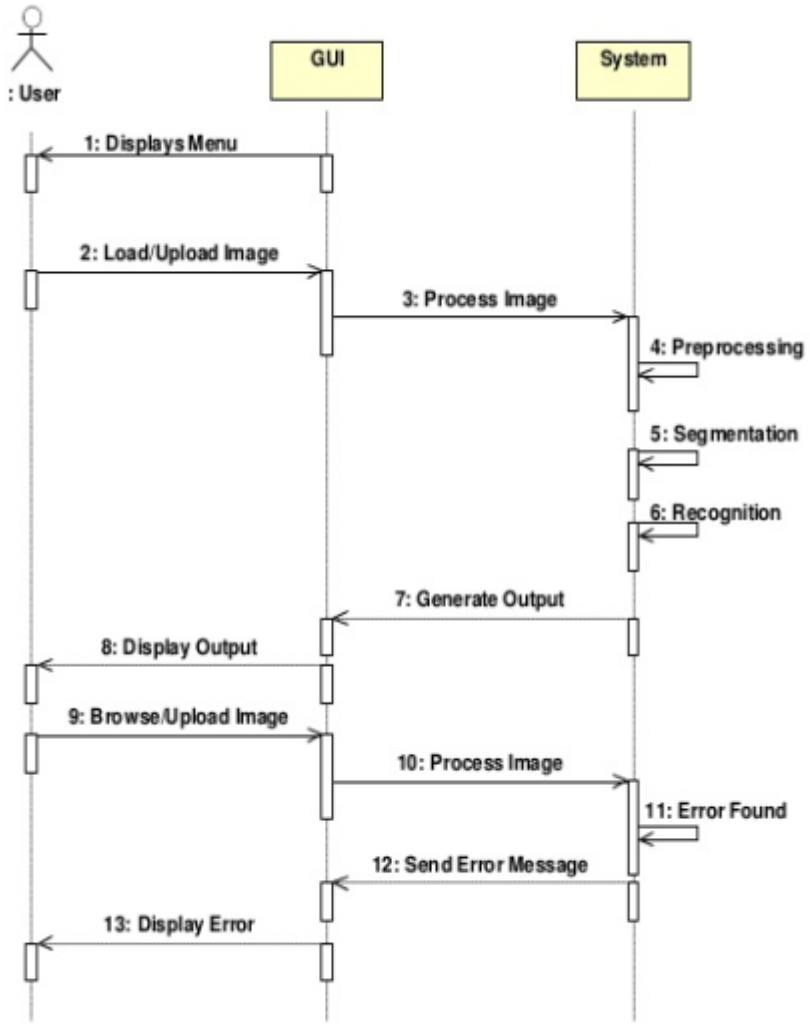

Figure 6.5: Sequence Diagram

Basic elements:

- **Vertical rectangle:** Represent the object is active (method is being performed).
- **Vertical dashed line:** Represent the life of the object.
- **X:** represent the life end of an object. (Being destroyed from memory)
- **Horizontal line with arrows:** Messages from one object to another.

Chapter 7

IMPLEMENTATION

7.1 INTRODUCTION

The implementation phase of the project is where the detailed design is actually transformed into working code. Aim of the phase is to translate the design into a best possible solution in a suitable programming language. This chapter covers the implementation aspects of the project, giving details of the programming language and development environment used. It also gives an overview of the core modules of the project with their step by step flow. The implementation stage requires the following tasks:

- Careful planning.
- Investigation of system and constraints.
- Design of methods to achieve the changeover.
- Evaluation of the changeover method.
- Correct decisions regarding selection of the platform.
- Appropriate selection of the language for application development.

7.2 TRAINING MODULE CODE

```
import math
from sklearn import neighbors
import os
import os.path
import pickle
from PIL import Image, ImageDraw
import face_recognition
from face_recognition.face_recognition_cli
import image_files_in_folder

ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg'}

def train(train_dir, model_save_path=None, n_neighbors=None,
 knn_algo='ball_tree', verbose=False):
    X = []
    y = []
    for class_dir in os.listdir(train_dir):
```

```
            if not os.path.isdir(os.path.join
                    (train_dir, class_dir)):
                continue

        for img_path in image_files_in_folder
          (os.path.join(train_dir, class_dir)):
              image = face_recognition.load_image_file(img_path)
          face_bounding_boxes = face_recognition.face_locations
                                    (image)
              if len(face_bounding_boxes) != 1:
                  if verbose:
                    print("Image {} not suitable for training: {}".
                     format(img_path, "Didn't find a face" if len
          (face_bounding_boxes) < 1 else "Found more
 ..........than one face"))
              else:
                  X.append(face_recognition.face_encodings
          (image, known_face_locations=face_bounding_boxes)[0])
                  y.append(class_dir)

      if n_neighbors is None:
          n_neighbors = int(round(math.sqrt(len(X))))
          if verbose:
              print("Chose n_neighbors automatically:", n_neighbors)

      knn_clf = neighbors.KNeighborsClassifier(n_neighbors=n_neighbors,
           algorithm=knn_algo, weights='distance')
      knn_clf.fit(X, y)

      if model_save_path is not None:
          with open(model_save_path, 'wb') as f:
              pickle.dump(knn_clf, f)

      return knn_clf


def predict(X_img_path, knn_clf=None,
          model_path=None, distance_threshold=0.6):
```

```
    if not os.path.isfile(X_img_path) or
        os.path.splitext(X_img_path)
        [1][1:] not in ALLOWED_EXTENSIONS:

        raise Exception("Invalid image path: {}"
                .format(X_img_path))

    if knn_clf is None and model_path is None:
        raise Exception("Must supply classifier either
        thourgh knn_clf or model_path")

    if knn_clf is None:
        with open(model_path, 'rb') as f:
            knn_clf = pickle.load(f)


    X_img = face_recognition.load_image_file(X_img_path)
    X_face_locations = face_recognition.face_locations(X_img)

    if len(X_face_locations) == 0:
        return []

    faces_encodings = face_recognition.face_encodings
        (X_img, known_face_locations=X_face_locations)

    closest_distances = knn_clf.kneighbors(faces_encodings,
        n_neighbors=1)
    are_matches = [closest_distances[0][i][0] <=
        distance_threshold for i in range(len
                    (X_face_locations))]

    return [(pred, loc) if rec else ("unknown", loc) for pred,
        loc, rec in zip(knn_clf.predict(faces_encodings)
        , X_face_locations,are_matches)]


def show_prediction_labels_on_image(img_path, predictions):
    pil_image = Image.open(img_path).convert("RGB")
    draw = ImageDraw.Draw(pil_image)
```

```
    for name, (top, right, bottom, left) in predictions:
        draw.rectangle(((left, top), (right, bottom))
                       , outline=(0, 0, 255))

        name = name.encode("UTF-8")
        text_width, text_height = draw.textsize(name)
        draw.rectangle(((left, bottom - text_height - 10)
                       , (right, bottom))
        , fill=(0, 0, 255), outline=(0, 0, 255))
        draw.text((left + 6, bottom - text_height - 5),
         name, fill=(255, 255, 255, 255))
    del draw

    pil_image.show()


if __name__ == "__main__":
    print("Training classifier...")
    classifier = train("train", model_save_path=
        "trained_model.dat", n_neighbors=2)
    print("Training complete!")

    for image_file in os.listdir("test"):
        full_file_path = os.path.join("test", image_file)

        print("Looking for faces in {}".format(image_file))

        predictions = predict(full_file_path, model_path=
                                        "trained_model.dat")

        for name, (top, right, bottom, left) in predictions:
            print("- Found {} at ({}, {})".format
                        (name, left, top))

        show_prediction_labels_on_image(os.path.join
                        ("test", image_file), predictions)
```

7.2.1 TRAINING MODULE SET,INPUT AND OUTPUT SCREENSSHOTS

Figure 7.1: Training Data Set

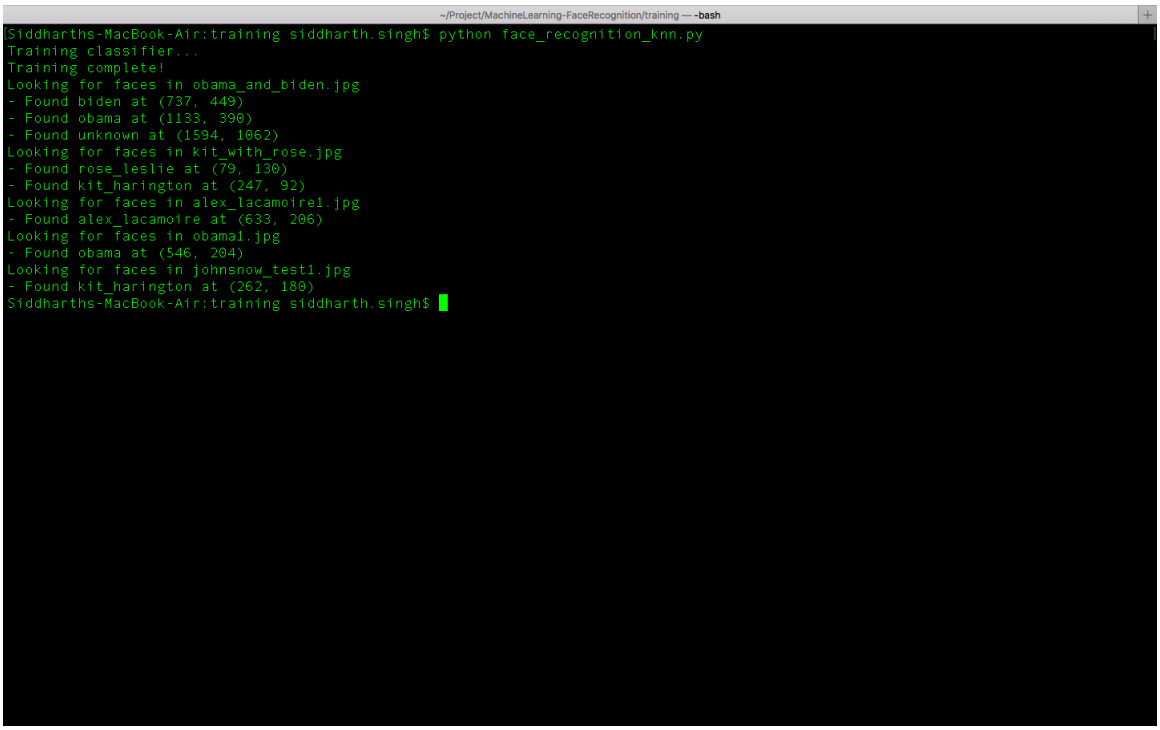

Figure 7.2: Training Console Input

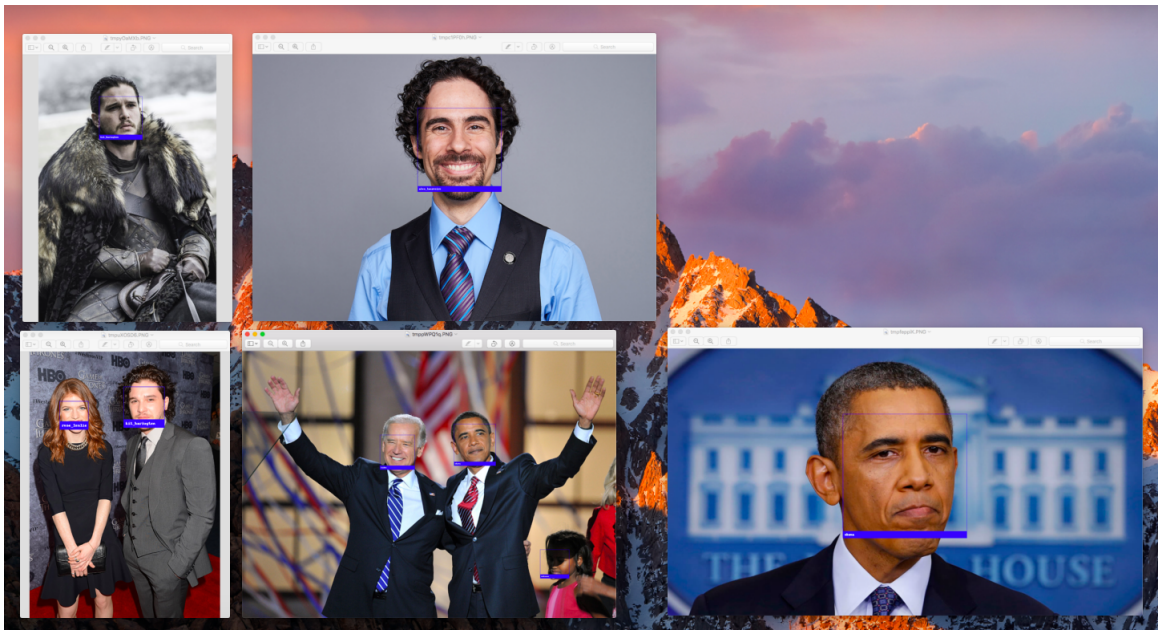

Figure 7.3: Training Data Output

7.3 IMAGE RECOGNITION CODE

```
import face_recognition as face_recognition
from PIL import Image , ImageTk
import Tkinter
import cv2
import constants as CONSTANTS
import sys

videoCApture = cv2.VideoCapture(0)

ret , image = videoCApture.read()

print "Enter File NAme of Known People (Single Line Seperated
By Spaces)"
knownImages = raw_input().split(' ')
print "Enter Original NAme OF Unknown Peoples (Single Line
Seperated By Spaces)"
knownNames =raw_input().split(' ')

knownImagesPreEncoded = []
```

```
print knownNames," And ",knownImages

#Create Encoding For All The Known Images
for knownImage in knownImages:
    tempImageBinary = face_recognition.load_image_file(knownImage)
    image2Learn =(face_recognition.face_encodings(tempImageBinary))
    if(len(image2Learn)!=1):
        if image2Learn == 0:
            print ("Unable To Recogonize Image", knownImage,
                "COntains",len(image2Learn), "FAces")
            exit()
        else:
            print ("Make Sure There is Just One Face", knownImage,
                "COntains",len(image2Learn), "FAces")
            exit()
    knownImagesPreEncoded.append((face_recognition.face_encodings
            (tempImageBinary))[0])

del knownImages


def updateImage(root,canvas):
    try:
        #Get the Image From VIdeo Camera
        ret, frame = videoCApture.read()
        #Scale It DOwn For Speed
        small_frame = cv2.resize(frame, (0, 0),
                fx=CONSTANTS.SCALING_X,
                fy=CONSTANTS.SCALING_Y)
        #set this Image as MAin IMage
        image = cv2.cvtColor(small_frame,
         cv2.COLOR_BGR2RGB)

        #identify Image
        face_locations = face_recognition.face_locations(image)

        face_encodings = face_recognition.face_encodings
                            (image, face_locations)
```

```
        tempReco = []

        for i in range(len(face_encodings)):
            peopleBools = face_recognition.compare_faces
                        (knownImagesPreEncoded, face_encodings[i],
                        CONSTANTS.COMPARISON_TOLERENCE)
            tempReco.append("Unknown")
            print peopleBools
            for j in range(len(peopleBools)):
                if peopleBools[j] and tempReco[i]=="Unknown":
                    tempReco[i]=knownNames[j]
                elif peopleBools[j]:
                    tempReco[i]=tempReco[i] + "_OR_" + knownNames[j]


    for fset in range(len(face_locations)):
        image[face_locations[fset][0]][min
         (face_locations[fset][1],
         face_locations[fset][3]):max(face_locations[fset][1],
        face_locations[fset][3])]=
        CONSTANTS.BOUNDING_BOX_COLOR
        image[face_locations[fset][2]][min(face_locations[fset][1],
       face_locations[fset][3]):
       max(face_locations[fset][1],
      face_locations[fset][3])]=
       CONSTANTS.BOUNDING_BOX_COLOR
          for i in range(min(face_locations[fset][0]
              ,face_locations[fset][2])
         ,max(face_locations[fset][0],face_locations[fset][2])):
            image[i][face_locations[fset][1]]
                    =CONSTANTS.BOUNDING_BOX_COLOR
            image[i][face_locations[fset][3]]
                  =CONSTANTS.BOUNDING_BOX_COLOR
            im = Image.fromarray(image, 'RGB')
        canvas.image = ImageTk.PhotoImage(im)
        canvas.create_image(0, 0, image=canvas.image, anchor='nw')
        for fset in range(len(face_locations)):
            canvas.create_text((face_locations[fset][1]+
            face_locations[fset][3])/2
```

```
            ,min(face_locations[fset][0],face_locations[fset][2])
            ,fill=CONSTANTS.LABEL_TEXT_COLOR
            ,font=CONSTANTS.LABEL_FONT,text=tempReco[fset])

            canvas.pack()

            print("Updating...")
        except IndexError:
            print "Error :",sys.exc_info()
        canvas.after(1, updateImage, root,canvas)

#Create A WIndow
top = Tkinter.Tk()
#Add CAnvas To Window To DIsplay RGB Image
canvas = Tkinter.Canvas(top,height=image.shape[0]*
            CONSTANTS.SCALING_X
            , width=image.shape[1]*CONSTANTS.SCALING_Y)
#Update CAnvas As Fast As Possible
updateImage( top,canvas)
#LIsten For Evwnts
top.mainloop()

#Release Video Permission
videoCApture.release()
```

Chapter 8

TESTING AND RESULTS

8.1 INTRODUCTION

Testing is an important phase in the development life cycle of the product this was the phase where the error remaining from all the phases was detected. Hence testing performs a very critical role for quality assurance and ensuring the reliability of the software. Once the implementation is done, a test plan should be developed and run on a given set of test data. Each test has a different purpose, all work to verify that all the system elements have been properly integrated and perform allocated functions. The testing process is actually carried out to make sure that the product exactly does the same thing what is suppose to do. Testing is the final verification and validation activity within the organization itself. In the testing stage following goals are tried to achieve:-

- To affirm the quality of the project.
- To find and eliminate any residual errors from previous stages.
- To validate the software as the solution to the original problem.
- To provide operational reliability of the system.

During testing the major activities are concentrated on the examination and modification of the source code. The test cases executed for this project are listed below. Description of the test case, steps to be followed; expected result, status and screenshots are explained with each of the test cases.

8.2 TESTING METHODOLOGIES

There are many different types of testing methods or techniques used as part of the software testing methodology. Some of the important types of testing are:

8.2.1 WHITE BOX TESTING

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level. Using white box testing we can derive test cases that:

- Guarantee that all independent paths within a module have been exercised at least once.
- Exercise all logical decisions on their true and false sides.

- Execute all loops at their boundaries and within their operational bounds.
- Execute internal data structure to assure their validity.

8.2.2 BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot see into it. The test provides inputs and responds to outputs without considering how the software works. It uncovers a different class of errors in the following categories:

- Incorrect or missing function.
- Performance errors.
- Initialization and termination errors.
- Errors in objects.

8.2.3 UNIT TESTING

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases. **Test strategy and approach**
Field testing will be performed manually and functional tests will be written in detail.

8.3 TEST CASE

Figure 8.1: Input Images

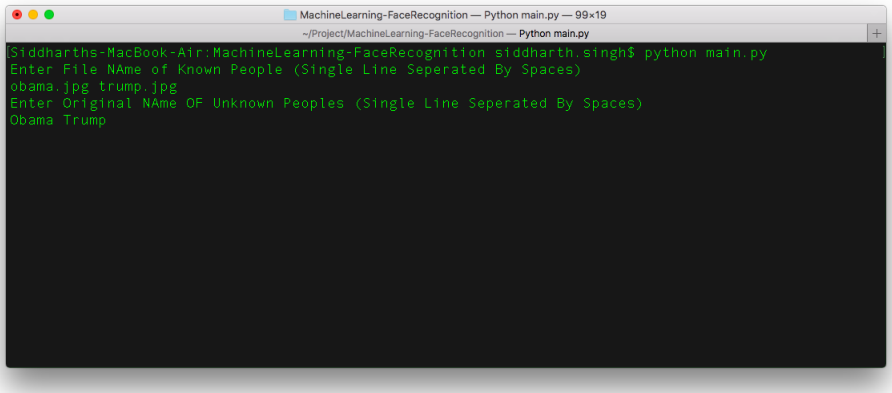

Figure 8.2: Recognition Console Input

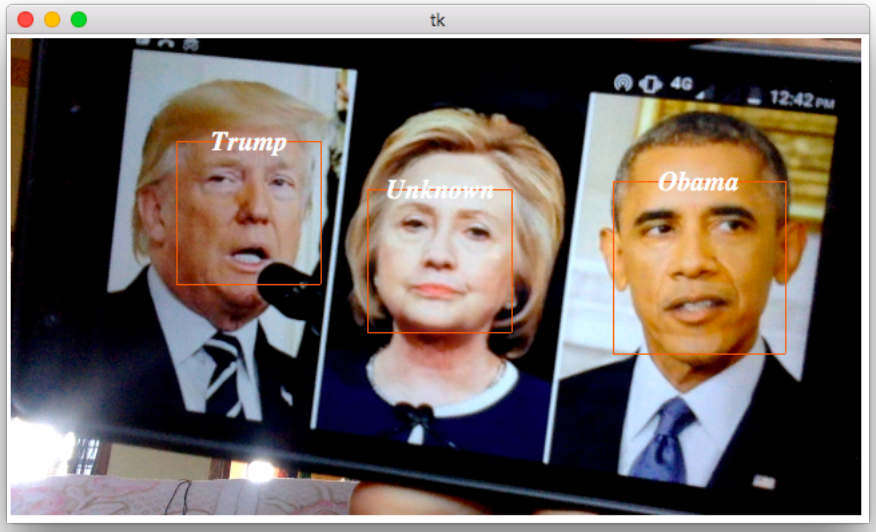

Figure 8.3: Output

Chapter 9

CONCLUSION & FUTURE SCOPE

9.1 CONCLUSION

This project was the first attempt to develop a system of this nature. We idntified from the beginning that producing a complete result would be impossible within the given time frame. We viewed the project as a journey where we learnt many lessons and gained insights to the subject which we tried to share in this report and summarised in this chapter. We tried to look at the problem from many points of view which generated some new ideas that could be explored in future. We suggested formal approaches for modelling and analysing the system which are by no means complete but could become the initiation for further research. We also created a working system and algorithms which we claim to be useful and extensible. However, as we have seen in these chapter, all these achievements are only partialy successful.

Personally, we would consider this project a success if the ideas described in the report can beme a useful reference for future work on the subject.

9.2 FUTURE SCOPE

This Project might, in the future, allow:

- robots that can see, feel, and predict the world around them
- improved stock prediction
- common usage of self-driving cars
- composition of music
- handwritten documents to be automatically transformed into formatted word processing documents
- trends found in the human genome to aid in the understanding of the data compiled by the Human Genome Project
- self-diagnosis of medical problems using neural networks
- and much more!

References

[1] Li Deng and Dong Yu ,Deep Learning: methods and applications, by Microsoft research

[2] Lillesand, T.M. and Kiefer, R.W. and Chipman, J.W., in Remote Sensing and Image Interpretation

[3] About deep learning `http://research.microsoft.com/pubs/209355`

[4] About Neural Network `http://www.uow.edu.au/œphung`

[5] A back-propagation neural network based method for post life expectancy estimation of thoracic surgery patients Abhishek Kumar Pandey; Mohd Anas Khan; Aleena Swetapadma 2017 International Conference On Smart Technologies For Smart Nation .