A Major Project Report on

SIGN LANGUAGE TO TEXT AND SPEECH TRANSLATION IN REAL TIME USING CNN

Submitted in partial fulfilment of the
requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

By

Mani Sai V	18UJ1A1233
K Abhinav	18UJ1A1211
Supriya N	18UJ1A1221
Vaibhavi	18UJ1A1232
Sandeep V	18UJ1A1234

Under the esteemed guidance of

Mr Mohammed Rayees

Assistant Professor

Department of Information Technology

At

MALLA REDDY ENGINEERING COLLEGE AND MANAGEMENT SCIENCES
(Approved by AICTE New Delhi & Affiliated to JNTU Hyderabad)
Kistapur, Medchal, Medchal Dist- 501401.

2021 - 2022

Affiliated to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD
KUKATPALLY, HYDERABAD-85.

MALLA REDDY ENGINEERING COLLEGE AND MANAGEMENT SCIENCES

(Approved by AICTE New Delhi & Affiliated to JNTU Hyderabad)

Kistapur, Medchal, Ranga Reddy Dist- 501401.

CERTIFICATE

This is to certify that the Major Project report entitled " SIGN LANGUAGE TO TEXT AND SPEECH TRANSLATION IN REAL TIME USING CNN“ is the bonafide work carried out and submitted by

K Abhinav	18UJ1A1211
Supriya N	18UJ1A1221
Vaibhavi M	18UJ1A1232
Mani Sai	18UJ1A1234
Sandeep V	

To the department of Information Technology, Malla Reddy Engineering College and Management Sciences, in partial fulfilment for the award of BACHELOR OF TECHNOLOGY in INFORMATION TECHNOLOGY during the academic year 2018-2022.

Internal Guide

Mr Mohammed Rayees

Assistant Professor
Information Technology

Head of The Department

Professor
Information Technology

EXTERNAL

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible, whose constant guidance and encouragement crowned our efforts with success. It is a pleasant aspect that I have now the opportunity to express my guidance for all of them.

I am thankful to Mr. MALLA REDDY chairmen of Malla Reddy Engineering College and Management Sciences for accepting me and providing me with an opportunity to do a project in their esteemed organization.

I am thankful to Principal Dr.CNV.SRIDHAR M.Tech., Ph.D. Malla Reddy Engineering College and Management Sciences for helped me to undergo project work as a part of university curriculum.

My special thanks to ASHWINI Ph.D, Professor & Head in Information Technology and Mr. MOHAMMED RAYEES M.Tech. Assistant Professor in INFORMATION TECHNOLOGY Department for guiding me in the right way

We would like to thank our internal project mates and department faculties for their full-fledged guidance and giving courage to carry out the project.

I am very much thankful to one and all that helped me for the successful completion of my project.

K Abhinav	18UJ1A1211
Supriya N	18UJ1A1221
Vaibhavi M	18UJ1A1232
Mani Sai	18UJ1A1233
Sandeep V	18UJ1A1234

TABLE OF CONTENT

ABSTRACT

One of the major drawback of our society is the barrier that is created between disabled or handicapped persons and the normal person. Communication is the only medium by which we can share our thoughts or convey the message but for a person with disability (deaf and dumb) faces difficulty in communication with normal person. For many deaf and dumb people, sign language is the basic means of communication. Sign language recognition (SLR) aims to interpret sign languages automatically by a computer in order to help the deaf communicate with hearing society conveniently. Our aim is to design a system to help the person who trained the hearing impaired to communicate with the rest of the world using sign language or hand gesture recognition techniques. In this project, feature detection and feature extraction of hand gesture is done with the help of CNN and OpenCV.

Creating a desktop application that uses a computer's webcam to capture a person signing gestures for American Sign Language (ASL), and translate it into corresponding text and speech in real time. The translated sign language gesture will be acquired in text which is farther converted into audio. In this manner we are implementing a finger spelling sign language translator. To enable the detection of gestures, we are making use of a Convolutional neural network (CNN). A CNN is highly efficient in tackling computer vision problems and is capable of detecting the desired features with a high degree of accuracy upon sufficient training.

CHAPTER 1 : INTRODUCTION

Image processing is a rapidly growing area in diverse applications, such as multimedia computing, secured data communication, biomedical, biometrics, remote sensing, texture understanding, pattern recognition, content-based retrieval, compression, and many more. This is all about how a computer can sense pictorial data after processing an image. Among the set of gestures intuitively performed by humans when communicating with each other, pointing gestures are especially interesting for communication and is perhaps the most intuitive interface for selection. They open up the possibility of intuitively indicating objects and locations, e.g., to make a robot change moving direction or simply mark some object. This is particularly useful in combination with speech recognition as pointing gestures can be used to specify parameters of location in verbal statements. This technology can be a boon for disable people who are not able to speak hence can't communicate. Also if the person has different language than receiver, then also, it can be used to as translator. There has been always considered a challenge the development of a natural interaction interface, where people interact with technology as they are used to interact with the real world. A hand free interface, based only on human gestures, where no devices are attached to the user, will naturally immerse the user from the real world to the virtual environment.

Hands are human organs which are used to manipulate physical objects. For this very reason hands are used most frequently by human beings to communicate and interact with machines. Mouse and Keyboard are the basic input/output to computers and the use of both of these devices require the use of hands. Most important and immediate information exchange between man and machine is through visual and actual aid, but this communication is one sided. Computers of this age provide humans with 1024 * 768 pixels at a rate of 15 frames per second and compared to it a good typist can write 60 words per minute with each word on average containing 6 letters. To help somewhat mouse remedies this problem, but there are limitations in this as well. Although hands are most commonly used for day to day physical manipulation related tasks, but in some cases they are also used for communication. Hand gestures support us in our daily communications to convey out- messages clearly. Hands are most important for mute and deaf people, who depends their hands and gestures to communicate, so hand gestures are vital for communication in sign language. If computer had the ability to translate and

understand hand gestures, it would be a leap forward in the field of human computer interaction. Some of the major problems faced by a person who are unable to speak is they cannot express their emotion as freely in this world. Utilize that voice recognition and voice search systems in smartphone(s). Audio results cannot be retrieved. They are not able to utilize (Artificial Intelligence/personal Butler) like google assistance, or Apple's SIRI etc. because all those apps are based on voice controlling.

There is a need for such platforms for such kind of people. American Sign Language (ASL) is a complete, complex language that employs signs made by moving the hands combined with facial expressions and postures of the body. It is the go-to language of many North Americans who are not able to talk and is one of various communication alternatives used by people who are deaf or hard-of-hearing.

While sign language is very essential for deaf-mute people, to communicate both with normal people and with themselves, is still getting less attention from the normal people. The importance of sign language has been tending to ignored, unless there are areas of concern with individuals who are deaf-mute. One of the solutions to talk with the deaf-mute people is by using the mechanisms of sign language.

Hand gesture is one of the methods used in sign language for non-verbal communication. It is most commonly used by deaf & dumb people who have hearing or talking disorders to communicate among themselves or with normal people. Various sign language systems have been developed by many manufacturers around the world but they are neither flexible nor cost-effective for the end users.

1.1 Problem Definition

It is well known fact that communication is very essential in day to day life. Humans with disabilities often experience issues in this area. [Deaf and Dumb].In the world of sign language, and gestures, a lot of research work has been done over past few years. Sign Language Recognition is one of the most growing fields of research area. Language is mainly used for communication of deaf-dumb people. In present scenario, it is impossible for humans to understand the sign language without any practice. We hereby facilitate a human machine interactive system that would be very

helpful for communication between deaf and dumb people and humans in real world situation.

1.2 Objectives

The main aim of the project is to provide an easy platform for the communication.
The project work focuses on the hand gesture using image processing. The main objective of the proposed system is to work as a medium of communication among deaf and dumb people to convey the message with normal person. A person who can talk and hear properly (normal person) cannot communicate with deaf & dumb person unless he/she is familiar with sign language.

1.3 Motivation

Machine learning techniques have been around us and has been compared and used for analysis for many kinds of data science applications. The major motivation behind this research-based project was to explore the feature selection methods, data preparation and processing behind the training models in the machine learning. With first hand models and libraries, the challenge we face today is data where beside their abundance, and our cooked models, the accuracy we see during training, testing and actual validation has a higher variance. Hence this project is carried out with the motivation to explore behind the models, and further implement image classification model to train the obtained data. Furthermore, as the whole machine learning is motivated to develop an appropriate computer-based system and decision support that can recognize the features occurring in an image. In this project we have developed a model which classifies the input image as either a dog or a cat image. The image input which we give to the system will be analyzed and the predicted result will be given as output.

CHAPTER 2: LITERATURE SURVEY

Using Multiple Sensors for Mobile Sign Language Recognition

Author: Helene Brashear, Thad Starner, Paul Lukowicz & Holger Junker

The authors built a constrained, lab-based Sign Language recognition system with the goal of making it a mobile assistive technology. They examine using multiple sensors for disambiguation of noisy data to improve recognition accuracy. The experiment compares the results of training a small gesture vocabulary using noisy vision data, accelerometer data and both data sets combined. The authors chose to use a rule–based grammar for sentence structure in the training and testing process. Speech recognition often uses statistical grammars for increased accuracy. These grammars are built by tying together phonemes (the simplest unit of speech) and training on the transition between the phonemes. The sets are usually done with bigrams (two phonemes tied together) or trigrams (three phonemes). Training using bigrams or trigrams requires considerably more data because representations of each transition of each word are now needed. In our case, the bigrams and trigrams would be built by tying together gestures. The current data set is too small to effectively train using bigrams or trigrams, but we intend to continue collecting data with the goal of implementing these techniques.

Advantage:

Benefit of the proposed design is that the user can monitor the camera's view via the head mounted display.

Provides accuracy.

Disadvantage:

Data set is too small to effectively train using bigrams or trigrams. The current system has only been trained on a very small vocabulary.

A Vision Based Dynamic Gesture Recognition of Indian Sign Language on Kinect based Depth Images

Author: Geetha M, Manjusha C, Unnikrishnan P and Harikrishnan R

Indian Sign Language (ISL) is a visual-spatial language which provides linguistic information using hands, arms, facial expressions, and head/body postures. The proposed work aims at recognizing 3D dynamic signs corresponding to ISL words. With the advent of 3D sensors like Microsoft Kinect Cameras, 3D geometric processing of images has received much attention in recent researches. The authors have captured 3D dynamic gestures of ISL words using Kinect camera and has proposed a novel method for feature extraction of dynamic gestures of ISL words. While languages like the American Sign Language (ASL) are of huge popularity in the field of research and development, Indian Sign Language on the other hand has been standardized recently and hence its (ISLs) recognition is less explored. The method extracts features from the signs and converts it to the intended textual form. The proposed method integrates both local as well as global information of the dynamic sign. A new trajectory based feature extraction method using the concept of Axis of Least Inertia (ALI) is proposed for global feature extraction. An Eigen distance based method using the seven3D key points- (five corresponding to each finger tips, one corresponding to center of the palm and another corresponding to lower part of palm), extracted using Kinect is proposed for local feature extraction. Integrating 3D local feature has improved the performance of the system as shown in the result. Apart from serving as an aid to the disabled people, other applications of the system also include serving as a sign language tutor, interpreter and also be of use in electronic systems that take gesture input from the users.

Advantage:

Improve the accuracy of recognition.

The proposed method integrates both local as well as global information of the dynamic sign.

Can handle different types of words in a common vision based platform.

Disadvantage:

These methods are not user friendly and are more expensive.

A Color Hand Gesture Database for Evaluating and Improving Algorithms on Hand Gesture and Posture Recognition

Author: Farhad Dadgostar, Andre L. C. Barczak, Abdolhossein Sarrafzadeh

With the increase of research activities in vision-based hand posture and gesture recognition, new methods and algorithms are being developed. Although less attention is being paid to developing a standard platform for this purpose. Developing a database of hand gesture images is a necessary first step for standardizing the research on hand gesture recognition. For this purpose, we have developed an image database of hand posture and gesture images. The database contains hand images in different lighting conditions and collected using a digital camera. Details of the automatic segmentation and clipping of the hands are also discussed in this paper.

Advantage:

Automatically vary the lighting fairly in all directions and even produce very complex patterns of lighting by introducing more than one source of light. Enable researchers to add their own backgrounds to the image or to use it as an
object with known boundaries.

Disadvantage:

Unless some special gadgets are used to control the lighting, it is very difficult to vary the positions of the light fairly along the three axis.

Low cost approach for Real Time Sign Language Recognition

Author: Matheesha Fernando, Janaka Wijayanayaka

Sign Language is the language of people who suffer from speech and hearing defects.

Still the rest of the world doesn't have a clear idea of sign language. The communication between speech impaired people and other people is very inefficient.

To overcome this problem technology can act as an intermediate flexible medium for speech impaired people to communicate amongst themselves and with other individuals as well as to enhance their level of learning / education. The suggested solutions in the literature for sign language recognition are very expensive for day to day use. Therefore, the main objective of this research is to find out a low cost affordable method of sign language interpretation. This paper discusses the possible ways to deal with the sign language postures to identify he signs and convert them into text and speech using appearance based approach with a low cost web camera. Further this approach ill be very useful to the sign language learners to practice sign language. During the research available human computer interaction approaches in posture recognition were tested and evaluated. A series of image processing techniques with Hub- moment classification was identified as the best approach. The system is able to recognize selected Sign Language signs with the accuracy of 76% without a controlled background with small light adjustments.

Advantage:

Helps in identifying a low cost, affordable method that can facilitate hearing and speech impaired people to communicate with the world in more comfortable way where they can easily get what they need from the society and also can contribute to the well-being of the society.

Can be used as a learning tool for sign language where hearing and speech impaired people can practice sign language using the application.

Disadvantage:

This project only looks at the hand postures not on hand gestures.

MILES : Multiple-Instance Learning via Embedded Instance Selection

Author: Yixin Chen, Jinbo Bi and James Z. Wang

Multiple-instance problems arise from the situations where training class labels are attached to sets of samples (named bags), instead of individual samples within each bag (called instances). Most previous multiple-instance learning (MIL) algorithms are developed based on the assumption that a bag is positive if and only if at least one of its instances is positive.

Although the assumption works well in a drug activity prediction problem, it is rather restrictive for other applications, especially those in the computer vision area. The authors proposed a learning method, MILES (Multiple-Instance Learning via Embedded instance Selection), which converts the multiple-instance learning problem to a standard supervised learning problem that does not impose the assumption relating instance labels to bag labels. MILES maps each bag into a feature space defined by the instances in the training bags via an instance similarity measure. This feature mapping often provides a large number of redundant or irrelevant features. Hence 1-norm SVM is applied to select important features as well as construct classifiers simultaneously.

Advantage:

Broad Adaptability: It provides a learning framework that converts a multiple- instance problem to a supervised learning problem.

Low complexity: It is efficient in computational complexity, therefore, can potentially be tailored to tasks that have stringent time or resource limits.

Prediction capability: In some multiple-instance problems, classification of instances is at least as important as the classification of bags.

Disadvantage:

The performance of MILES depends on whether there are “useful” features among those defined by the instances in the training bags.

In some applications, for example 3D object recognition, geometric constraints on the image patches are extremely useful in reducing the search space and improving the recognition accuracy. However, MILES is not designed to take advantage of this type of prior knowledge.

The feature vectors generated by the mapping are not sparse.

RGB-H-CbCr Skin Color Model for Human Face Detection

Author: Nusirwan Anwar bin Abdul Rahman, Kit Chong Wei and John See

While the RGB, HSV and YUV (YCbCr) are standard models used in various color imaging applications, not all of their information are necessary to classify skin color. This paper presents a novel skin color model, RGB-H-CbCr for the detection of human faces. Skin regions are extracted using a set of founding rules based on the skin color distribution obtained from a training set. The segmented face regions are further classified using a parallel combination of simple morphological operations. This model utilizes the additional hue and chrominance information of the image on top of standard RGB properties to improve the discriminality between skin pixels and non-skin pixels. In the proposed approach, skin regions are classified using the RGB boundary rules introduced by Peer et al. and also additional new rules for the H and CbCr subspaces. These rules are constructed based on the skin color distribution obtained from the training images. The classification of the extracted regions is further refined using a parallel combination of morphological operations.

Advantage:

Able to deal with various brightness and illumination conditions as well as very effective compare to the other existing systems.

Disadvantage:

Doesn't provide success of a robust face detector.

Robust Real-Time Face Detection

Author: Paul Viola and Michael J. Jones

This paper describes a face detection framework that is capable of processing images extremely rapidly while achieving high detection rates. There are three key contributions. The first is the introduction of a new image representation called the “Integral Image” which allows the features used by our detector to be computed very quickly. In order to achieve true scale invariance, almost all face detection systems must operate on multiple image scales. The integral image, by eliminating the need to compute a multi-scale image pyramid, reduces significantly. Using the integral image, face detection is completed in almost the same time as it takes for an image pyramid to be computed. The initial image processing required for face detection. The second contribution of this paper is a simple and efficient classifier built from computationally efficient features using AdaBoost for feature selection. This classifier is clearly an effective one for face detection and the authors are confident that it will also be effective in other domains such as automobile or pedestrian detection. Furthermore, the idea of an aggressive and effective technique for feature selection should have impact on a wide variety of learning tasks. The third contribution is a method for combining classifiers in a “cascade” which allows background regions of the image to be quickly discarded while spending more computation on promising face-like regions.

Advantage:

Minimizes computation time while achieving high detection accuracy and effectiveness.

Helps in very fast feature evaluation.

Is a simple and efficient classifier that is built by selecting a small number of important features from a huge library of potential features using AdaBoost.

Reduces computation time while improving detection accuracy.

Disadvantage:

The proposed system uses large and complex dataset which are difficult and time consuming.

Vision-Based Sign Language Translation Device

Author: Yellapu Madhuri, Anitha.G, Anburajan.

This report presents a mobile VISION-BASEDSIGN LANGUAGE TRANSLATION DEVICE for automatic translation of Indian sign language into speech in English to assist the hearing and/or speech impaired people to communicate with hearing people. The authors proposed a real-time vision-based system for recognizing finger spelling continuous Sign Language (ASL) using a single camera to track the user's unadorned hands. This system is broken down into three main parts starting with the image acquisition followed by image processing to extract features for recognition and last comes the recognition stage where signs are identified and audio output is given. The program starts with image acquisition,

i.e. sign images capturing by the camera. The acquired images are pre-processed to differentiate static and dynamic signs, and also the start and end of a sign. The images are processed to identify the region of interest. The unique features of each sign in the region of interest are extracted to be used in the recognition stage. In the recognition stage, the features extracted are compared with the available database of pattern matching templates. A threshold value is set for the maximum difference between the input sign and the database, if the difference is below the maximum limit, a match is found and the sign is recognized. Corresponding audio file is played on audio device. The program can be implemented in a laptop, desktop or an IOS mobile phone to operate with its inbuilt camera, processor and audio device.

Advantage:

It can be used as a translator between deaf and people that do not understand sign language, avoiding by this way the intervention of an intermediate person.

The proposed system is highly consistent, reproducible, with fairly high precision and accuracy.

Disadvantage:

This project did not focus on facial expressions although it is well known that facial expressions convey important part of sign-languages.

Survey on Various Gesture Recognition Techniques For Interfacing Machines Based On Ambient Intelligence

Author: Harshith.C, Karthik.R.Shastry, Manoj Ravindran, M.V.V.N.S Srikanth, Naveen Lakshmikhanth

Gesture recognition is mainly apprehensive on analyzing the functionality of human wits. The main goal of gesture recognition is to create a system which can recognize specific human gestures and use them to convey information or for device control. Hand gestures provide a separate complementary modality to speech for expressing ones ideas. Information associated with hand gestures in a conversation is degree, discourse structure, spatial and temporal structure. The approaches present can be mainly divided into Data- Glove Based and Vision Based approaches. An important face feature point is the nose tip. Since nose is the highest protruding point from the face. Besides that, it is not affected by facial expressions. Another important function of the nose is that it is able to indicate the head pose. Knowledge of the nose ocation will enable us to align an unknown 3D face with those in a face database. Eye detection is divided into eye position detection and eye contour detection. The purpose of this paper is to compare various human Gesture recognition systems for
interfacing machines directly to human wits without any corporeal media in an ambient environment.

Advantage:

By incorporating the proposed method the efficiency could be enhanced to greater extent.

Disadvantage:

The devices are quite expensive and bring much cumbersome experience to the users.

A New 2D Static Hand Gesture Color Image Dataset for ASL Gestures

Author: A.L.C. Barczak, N.H. Reyes, M. Abastillas, A. Piccio and T. Susnjak

It usually takes a fusion of image processing and machine learning algorithms in order to build a fully-functioning computer vision system for hand gesture recognition. Fortunately, the complexity of developing such a system could be alleviated by treating the system as a collection of multiple sub-systems working together, in such a way that they can be dealt with in isolation. Machine learning need to feed on thousands of exemplars (e.g. images, features) to automatically establish some recognizable patterns for all possible classes (e.g. and gestures) that applies to the problem domain. A good number of exemplars helps, but it is also important to note that the efficacy of these exemplars depends on the variability of illumination conditions, hand postures, angles of rotation, scaling and on the number of volunteers from whom the hand gesture images were taken. These exemplars are usually subjected to image processing first, to reduce the presence of noise and extract the important features from the images. These features serve as inputs to the machine learning system. Different sub-systems are integrated together to form a complete computer vision system for gesture recognition. The main contribution of this work is on the production of the exemplars. A minor contribution is given in the form of a specific feature extraction method called moment invariants, for which the computation method and the values are furnished with the dataset.

Advantage:

For gestures, no need to use special gloves, or any other apparatus.

Disadvantage:

Images were taken at a certain angle of rotation (perpendicular to the subject), which limits the number of samples.

CHAPTER 3: SYSTEM ANALYSIS

3.1 Existing System

The most conventional way of communication for deaf and dumb is through lip reading. The other approaches could be through interpreter (mediator). The drawbacks with interpreter is that the interpreter might manipulate the sentence with ease and also they need to take the interpreter along with them and moreover the interpreter should be well trained.
The other approach is using pen and paper. But the drawback is all the people are not literate.

3.2 Disadvantages of Existing System

But the usage of sign language interpreters could be the expensive. Cost-effective solution is required so that the deaf-mute and normal people can communicate normally and easily.
The major drawback of lip reading is misunderstanding and miscommunication.
This method may lead to confusion.
Always cannot be correct
So very inefficient method

3. 3 Proposed System

Our strategy involves implementing such an application which detects pre-defined American sign language (ASL) through hand gestures. For the detection of movement of gesture, we would use basic level of hard ware component like camera and interfacing is required. Our application would be a comprehensive User-friendly Based system built on PyQt5 module. Instead of using technology like gloves or Kinect, we are trying to solve this problem using state of the art computer vision and machine learning algorithms.
Given a hand gesture, implementing such an application which detects pre-defined American sign language (ASL) in a real time through hand gestures and providing facility for the user to be able to store the result of the character detected in a text file, also allowing such users to build their customized gesture so that the problems faced by persons who aren't able to talk vocal1y

can be accommodated with technological assistance and the barrier of expressing can be overshadowed.

To enable the detection of gestures, we are making use of a Convolutional neural network (CNN).

By use of the CNN algorithm, the gestures can be identified by the proposed application and converted to text, which can then be converted into speech.

A CNN is highly efficient in tackling computer vision problems and is capable of detecting the desired features with a high degree of accuracy upon sufficient training.

3. 4 Advantages Of Proposed System

Eliminates the need for an interpreter for communication between sign language and speech language

Easy to incorporate and execute in any supporting operating system

Real time translation

Does not require any additional hardware

More Efficiency

3. 5 Modules

- Input Image(Capture sign or Gesture) from Webcam
- Pre-processing and Segmentation
- Feature Extraction
- Classification
- Results Analysis

Input Image(Capture sign or Gesture) from Webcam

The image (gesture or sign) is captured using the laptop camera or the external device webcam to get better image clarity.

Preprocessing and segmentation

Image processing is necessary for image enhancement. During Preprocessing RGB image to convert into HSV color space. This step was taken because HSV color space was less sensitive to illumination changes compared to RGB. Then it was filtered, smoothened and finally the biggest binary linked object was being considered so as to avoid consideration of skin colored objects other than hand. To obtains the good result smoothing and filtering is done. Image segmentation is basically performed to locate the hand object in image.

Feature Extraction

Feature Extraction stage is necessary because certain features has to be extracted so that they are unique for each gesture or sign. After the decision is made that a sign is present, then the last frame is taken into consideration and features. The Feature Extraction is extract the features in all Images (gesture or sign) dataset are store in 'svm.pkl' and finally extract the labels store in 'labels.pkl' best on train data and test data.

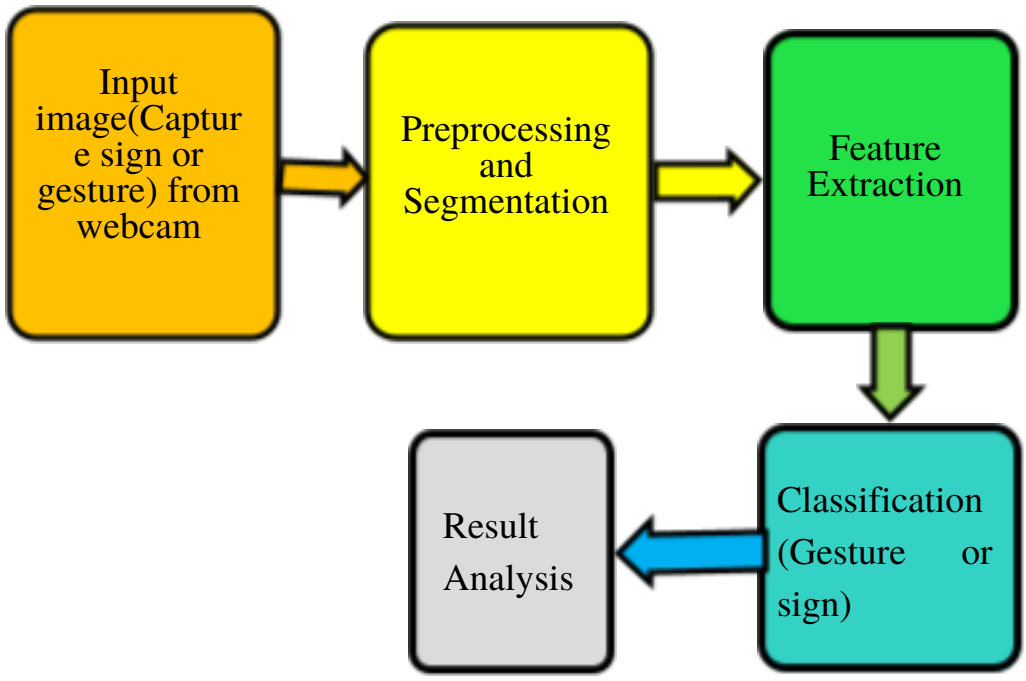


Figure - 1 : Modules

Classification

Classification of hand is done with the help of various features calculated previously. The five bit binary sequence is thus generated to uniquely recognize and utilize these recognized the recognized hand gesture for supporting human computer interaction. By the feature extraction significant peak is encoded as 1 while insignificant peak is encoded as 0 based on intersection to the threshold line.

Results analysis

Different images were tested and found that the new technique of classification was found to show 97% accuracy. Some images tested with other database images are given in the results analysis. In Results analysis are real time detect the sign language and sign recognize when live camera is start then capture the test images (gesture or sign)that time compare the features ‘svm.pkl' and ‘labels.pkl' if it is match the dataset after the process in display the result.

CHAPTER 4: FEASIBILITY 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

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.

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 avail able 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.

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 onstructive criticism, which is welcomed, as he is the final user of the system.

CHAPTER 5: SOFTWARE REQUIREMENT SPECIFICATION

A Software Requirement Specification (SRS) is a document that clearly and precisely specifies each and every requirement for the software product as well as the external interfaces to hardware and firmware. Each requirement should be defined so that it can be verified by a method such as inspection, demonstration, analysis and testing. There a number of desirable properties that an SRS should possess. In particular, the requirements documents should possess. In particular, the requirements documents should be:

- Correct
- Complete
- Consistent
- Functional
- Verifiable
- Traceable
- Easily changed

5. 1 Software Requirements:

- Operating System : Microsoft Windows 10
- IDE : Visual Studio Code
- Programming Language : Python

5. 2 Hardware Requirements:

- Processor : : Intel[i3] , 2.4 GHz
- RAM : 4 GB
- Hard disk : 250 GB

Functional Requirements

User Interface Requirements :

In addition to functions required, describe the characteristics of each interface between the product and its users (e.g., required screen formats/organization, report layouts, menu structures, error and other messages, or function keys).

Usability :

Include any specific usability requirements, for example,

Learnability

The user documentation and help should be complete

The help should be context sensitive and explain how to achieve common tasks The system should be easy to learn

Performance :

Specify static and dynamic numerical requirements placed on the system or on human interaction with the system:

Static numerical requirements may include the number of terminals to be supported, the number of simultaneous users to be supported, and the amount and type of information to be handled.

Dynamic numerical requirements may include the number of transactions and tasks and the amount of data to be processed within certain time period for both normal and peak workload conditions.

All of these requirements should be stated in measurable form. For example, "95% of the transactions shall be processed in less than 1 second" rather than “an operator shall not have to wait for the transaction to complete”.

Capacity :

Include measurable capacity requirements (e.g., the number of simultaneous users to be supported, the maximum simultaneous user load, per-user memory requirements, expected application throughput)

Availability :

Include specific and measurable requirements for:

Hours of operation

Level of availability required

Coverage for geographic areas

Impact of downtime on users and business operations

Impact of scheduled and unscheduled maintenance on uptime and maintenance communications procedures

Reliability (e.g., acceptable mean time between failures (MTBF), or the maximum permitted number of failures per hour).

Latency :

Include explicit latency requirements, e.g., the maximum acceptable time (or average time) for a service request.

Manageability/Maintainability : Monitoring :

Include any requirements for product or service health monitoring, failure conditions, error detection, logging, and correction.

Maintenance :

Specify attributes of the system that relate to ease of maintenance. These requirements may relate to modularity, complexity, or interface design. Requirements should not be placed here simply because they are thought to be good design practices.

Operations :

Specify any normal and special operations required by the user, including:

Periods of interactive operations and periods of unattended operations

Data processing support functions

Backup and recovery operations

Safety considerations and requirements

Disaster recovery and business resumption

Non-Functional 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. They are contrasted with functional requirements that define specific behaviour or functions. Non- functional requirements add tremendous value to business analysis. It is commonly misunderstood by a lot of people. It is important for business stakeholders, and Clients to clearly explain the requirements and their expectations in measurable terms. If the non-functional requirements are not measurable then they should be revised or rewritten to gain better clarity. For example, User stories help in mitigating the gap between developers and the user community in Agile Methodology.

1.Product Requirements

Portability:

While Microsoft has never implemented the full framework on any system except Microsoft Windows, it has engineered the framework to be platform-agnostic, and cross- platform implementations are available for other operating systems (see Silverlight and the Alternative implementations section below. Microsoft submitted the specifications for the Common Language Infrastructure which includes the core class libraries, Common Type System, and the Common Intermediate Language the C# language, and the C++/CLI language to both ECMA and the ISO, making them available as official standards. This makes it possible for third parties to create compatible implementations of the framework and its languages on other platforms.

Correctness :

The system followed 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.

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.

Non-functional requirements are also called the qualities of a system. These qualities can be divided into execution quality & evolution quality. Execution qualities are security and usability of the system which are observed during run time, whereas evolution quality involves testability, maintainability, extensibility or scalability.

2. 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.

Object oriented design methodology is employed to design the system. The detailed UML diagrams must be available at the end of design phase. The error correcting code, the encoding server algorithm, the peer division multiplexing method must be implemented in component style architecture so that any one extend or replace it easily.

Performance Requirements :

This system is developing in the high-level languages and using the advanced front-end and back-end technologies it will give response to the end user on the client system within very less time. Good band width, less congestion on the network.

Reliability :

The system is more reliable because of the qualities that are inherited from the chosen platform python. No harm is expected from the use of the product either to the OS or any data that resides on the client system.

Supportability :

The system is designed to be the cross platform supportable. The system is supported on a wide range of hardware and any software platform.

3. Software Quality Attributes

The system is user friendly and its accessibility is from the client side. Non-functional necessities square measure the other demand than practical necessities. This square measure the necessities that specifies criteria which will be won't to choose the operation of a system, instead of specific behaviours. Non- functional necessities square measure within the style of "system shall be ", associate degree overall property of the system as a full or of a specific facet and not a particular operate. The system's overall properties remarkably mark the distinction between whether or not the event project has succeeded or unsuccessful.

Our project qualifies all the criteria of functional and not functional accordingly, and system is up to mark performance vice. We can use simple layouts like card and grid layout etc. By varying colour and other UI combination many good intuitive interfaces can be made. Which ultimately make interface easy to use for a long time.

CHAPTER 6: SYSTEM DESIGN

The most creative and challenging phase of the system life cycle is the system design. System design is the process that states the details of how a system will meet the requirements identified during system analysis. When the analyst prepares logical system design, they specify the user needs at level of detail that virtually determines information flow into and out of the system and the required data source. First step in the design is to determine how the output is to be produced and in what format. Secondly, input data and master files have to be designed to meet the requirement of proposed output. Finally, the end of the design phase, the system flow chart will be ready which is used as the base of coding phase.

6.1 UML Diagrams

Unified Modeling Language:

The Unified Modeling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modelling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

Goals of UML:

The primary goals in the design of the UML are:

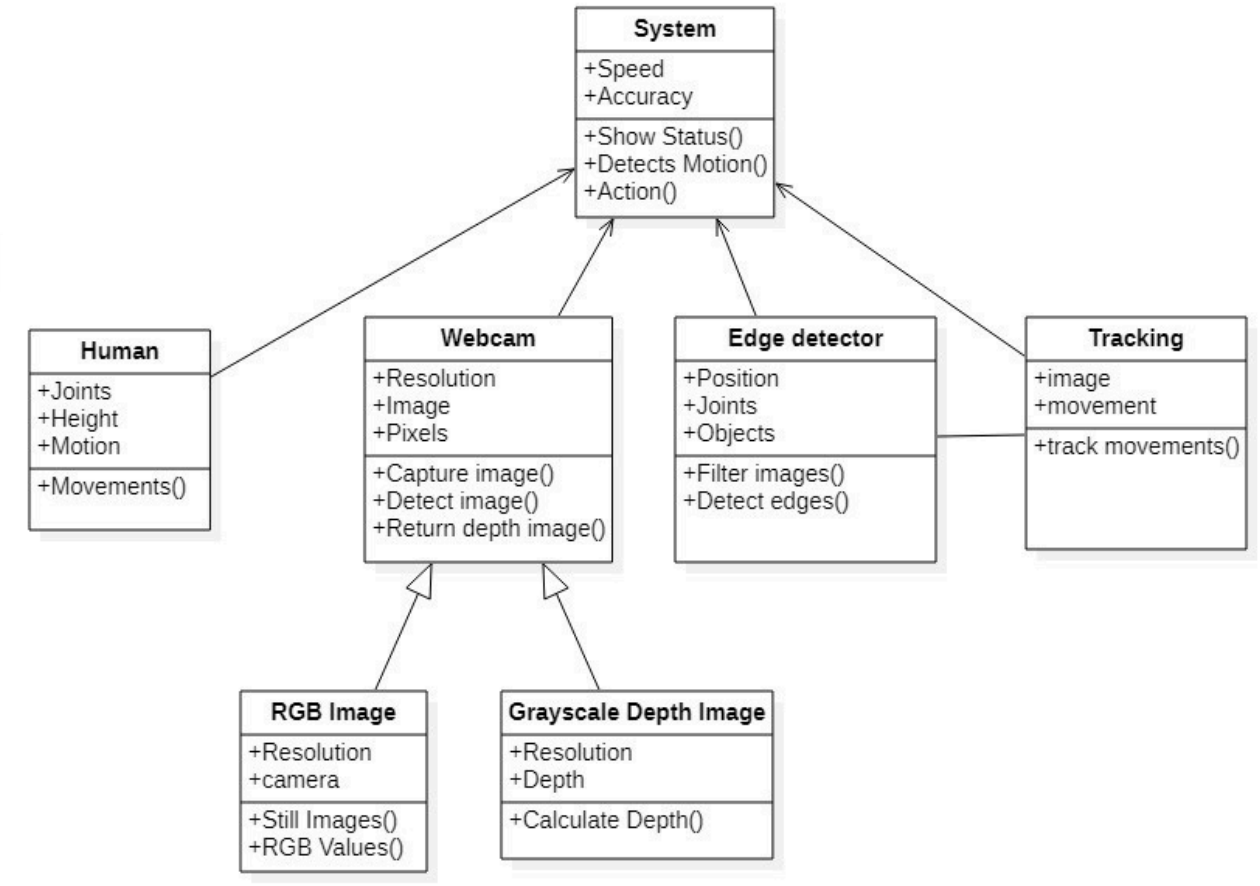
- Provide users with a ready-to-use, expressive visual modeling language so they can develop and exchange meaningful models.
- Provide extensibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development processes.
- Provide a formal basis for understanding the modeling language.
- Encourage the growth of the OO tools market.

- Support higher-level development concepts such as collaborations, frameworks, patterns and components.

As the strategic value of software increases for many companies, the industry looks for techniques to automate the production of software and to improve quality and reduce cost and time-to-market. Businesses also seek techniques to manage the complexity of systems as they increase in scope and scale. In particular, they recognize the need to solve recurring architectural problems, such as physical distribution, concurrency, replication, security, load balancing and fault tolerance. Additionally, the development for the World Wide Web, while making some things simpler, has exacerbated these architectural problems. The Unified Modeling Language (UML) was designed to respond to these needs.

6. 1 .1 Class Diagram

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. In the diagram, classes are represented with boxes which contain three parts:
The upper part holds the name of the class
The middle part contains the attributes of the class
The bottom part gives the methods or operations the class can take or undertake.
Class diagram is a static diagram and it is used to model the static view of a system. The static view describes the vocabulary of the system.
Class diagram is also considered as the foundation for component and deployment diagrams.
Class diagrams are not only used to visualize the static view of the system but they are also used to construct the executable code for forward and reverse engineering of any system.
Generally, UML diagrams are not directly mapped with any object-oriented programming languages but the class diagram is an exception.



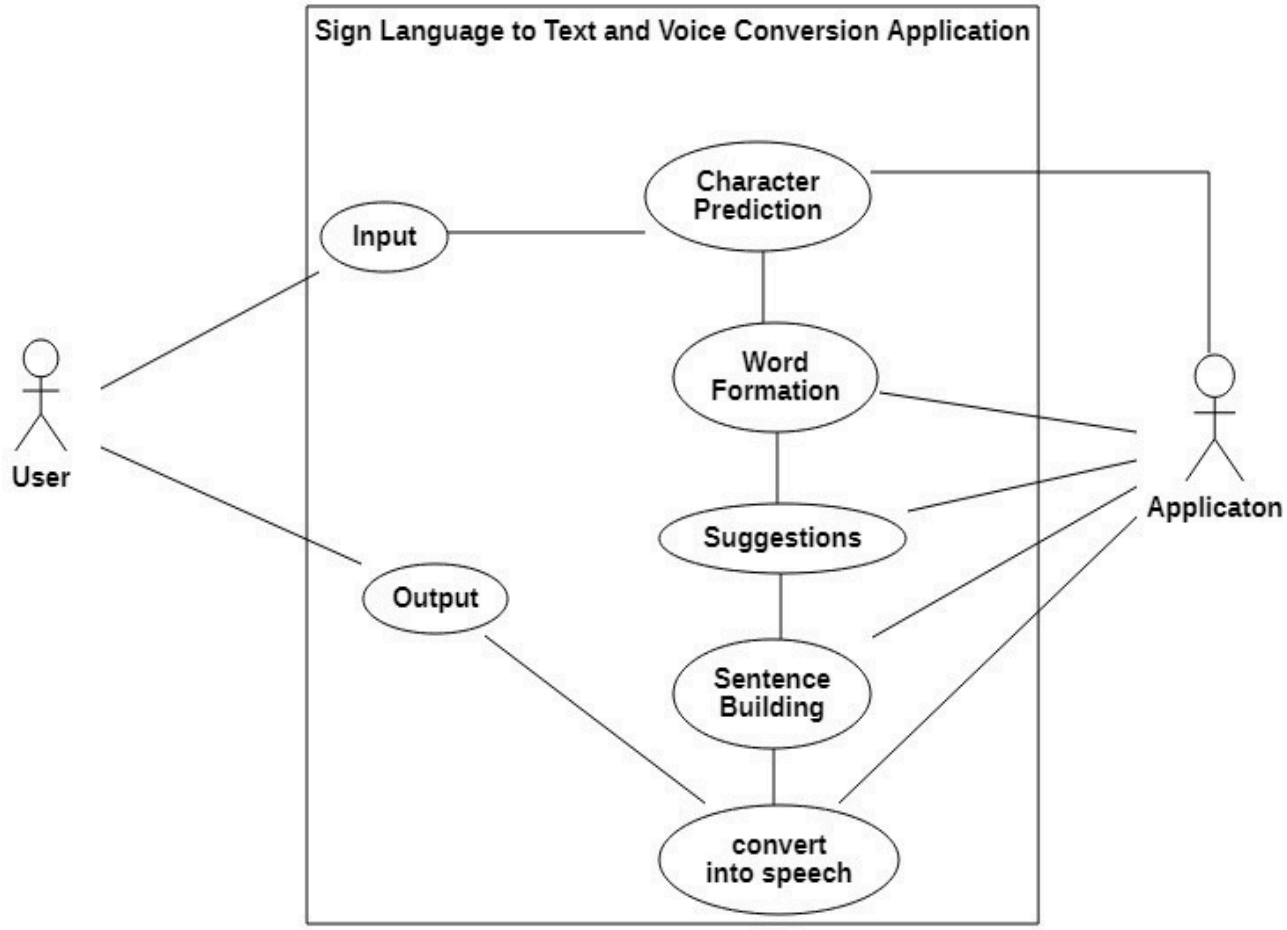
6. 1 .2 Use Case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well. Use Cases extend beyond pictorial diagrams. In fact, text-based use case descriptions are often used to supplement diagrams, and explore use case functionality in more detail.

The main purpose of a use case diagram is to portray the dynamic aspect of a system. It accumulates the system's requirement, which includes both internal as well as external influences. It invokes persons, use cases, and several things that invoke the actors and elements accountable for the implementation of use case diagrams. It represents how an entity from the external environment can interact with a part of the system.

It is essential to analyze the whole system before starting with drawing a use case diagram, and then the system's functionalities are found. And once every single functionality is identified, they are

then transformed into the use cases to be used in the use case diagram.

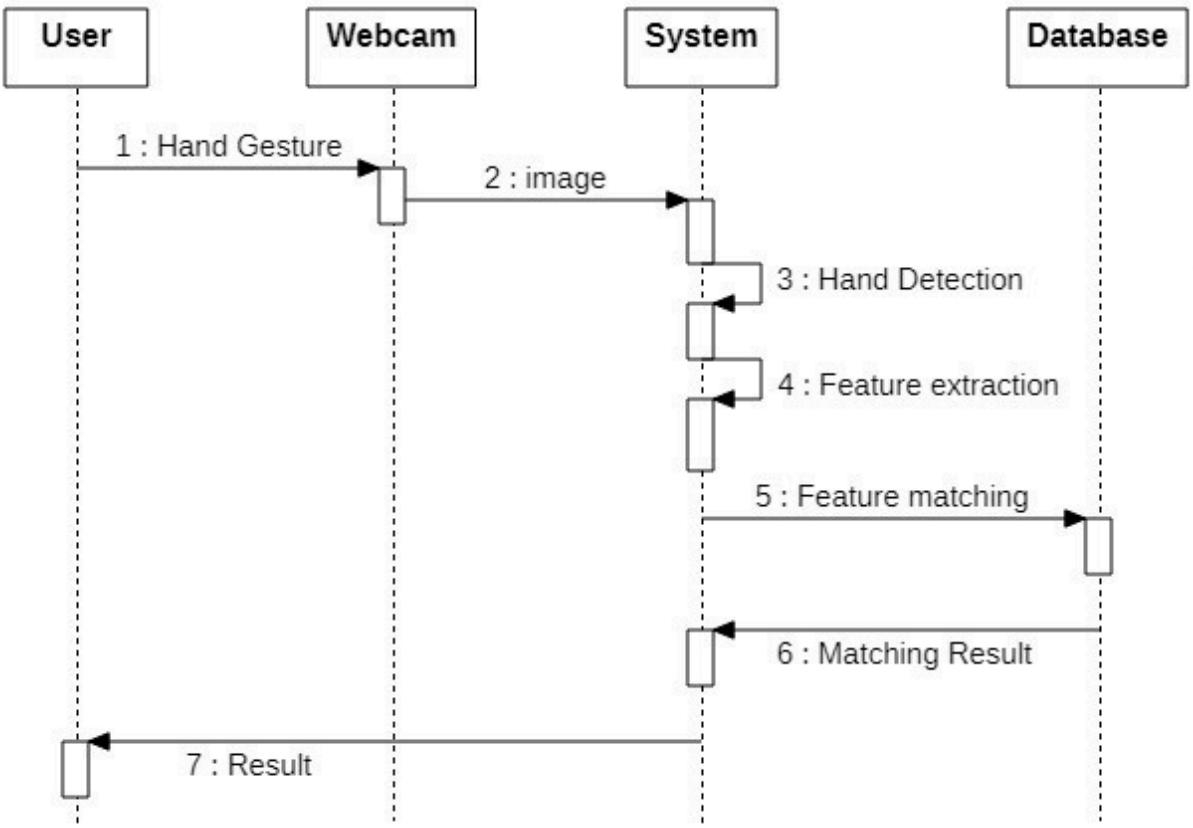


6. 1 .3 Sequence Diagram

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects

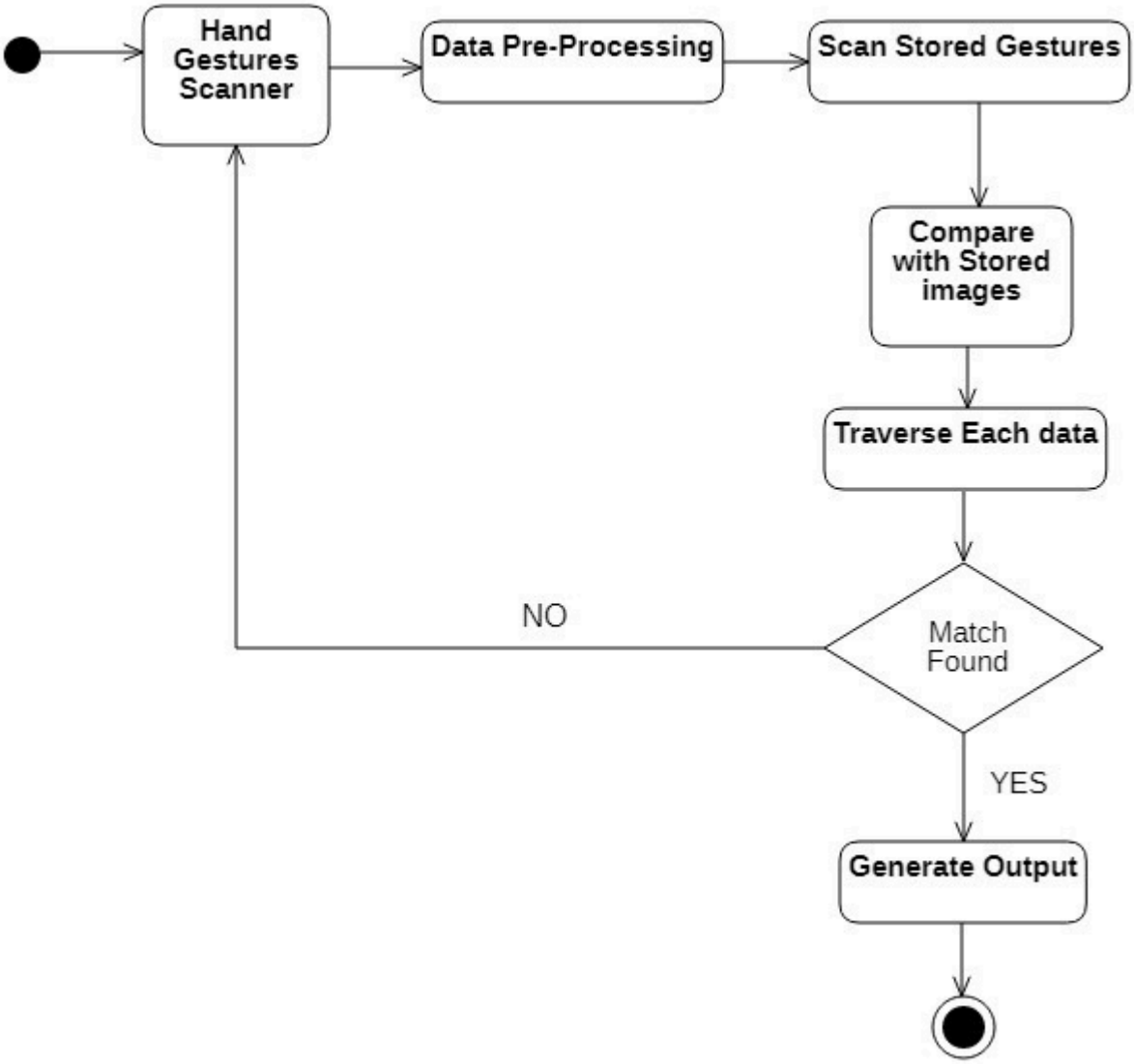
needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagram documents the interactions between classes to achieve a result, such as a use case. The sequence diagram lists objects horizontally, and time vertically, and models these messages over time.

Sequence diagrams are sometimes called event diagrams or event scenarios.



6. 1 .4 Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes (i.e., workflows), as well as the data flows intersecting with the related activities. Although activity diagrams primarily show the overall flow of control, they can also include elements showing the flow of data between activities through one or more data stores.

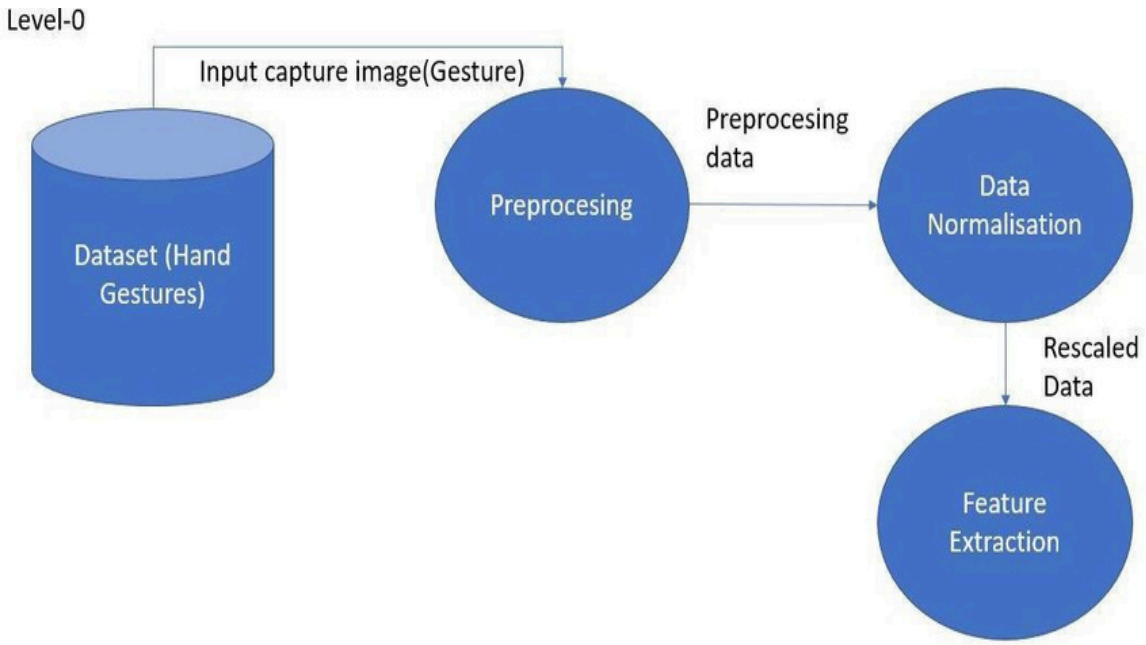

Hand Gestures Scanner
Data Pre-Processing
Scan Stored Gestures
Compare with Stored images
Traverse Each data
Match Found
NO
YES
Generate Output

6.2 Data Flow Diagram

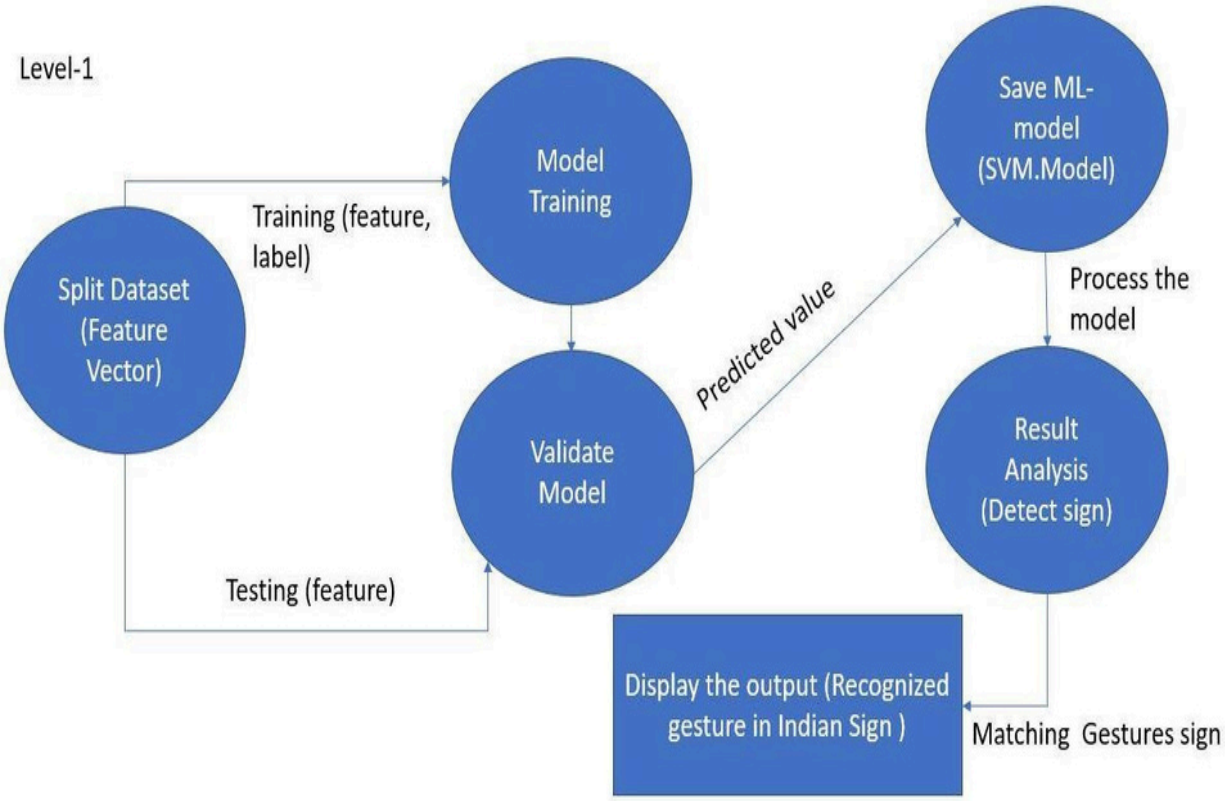
Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analyzing each of the functional areas of interest. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD can also be referred to as a Process Model. A DFD demonstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.

DFD Level 0

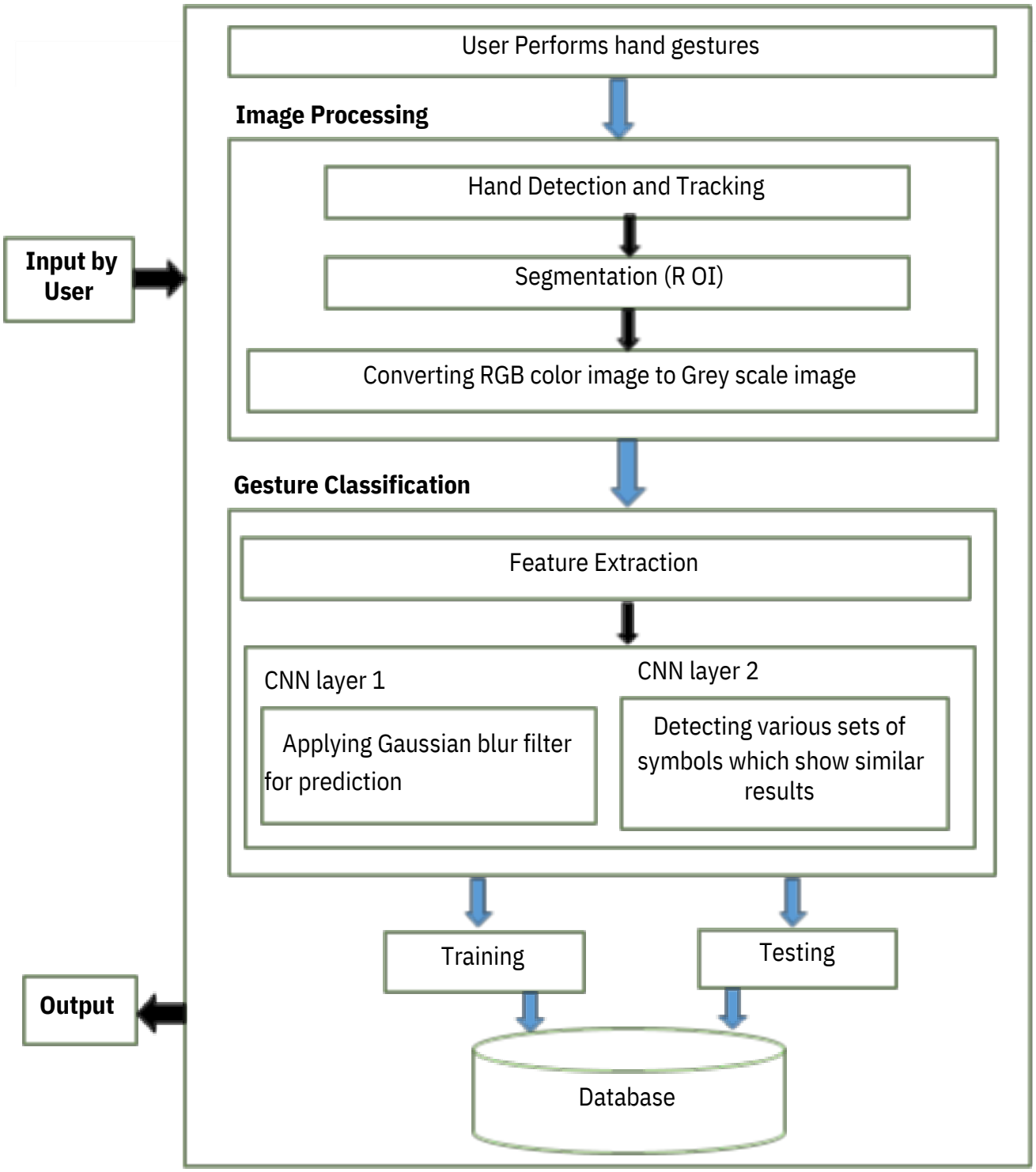


DFD Level 1


Level-1
Model Training
Save ML-model (SVM.Model)
Training (feature, label)
Split Dataset (Feature Vector)
Process the model
Predicted value
Validate Model
Result Analysis (Detect sign)
Testing (feature)
Display the output (Recognized gesture in Indian Sign)
Matching Gestures sign

CHAPTER 7: SYSTEM IMPLEMENTATION

7. 1 System Architecture



7. 2 Tools / Technologies Used

Keras

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research.

Keras is:

- Simple -- but not simplistic. Keras reduces developer cognitive load to free you to focus on the parts of the problem that really matter.
- Flexible -- Keras adopts the principle of progressive disclosure of complexity: simple workflows should be quick and easy, while arbitrarily advanced workflows should be possible via a clear path that builds upon what you've already learned.
- Powerful -- Keras provides industry-strength performance and scalability: it is used by organizations and companies including NASA, YouTube, or Waymo.
- Efficiently executing low-level tensor operations on CPU, GPU, or TPU.
- Computing the gradient of arbitrary differentiable expressions.
- Scaling computation to many devices, such as clusters of hundreds of GPUs.
- Exporting programs ("graphs") to external runtimes such as servers, browsers, mobile and embedded devices.

Keras is the high-level API of TensorFlow 2: an approachable, highly-productive interface for solving machine learning problems, with a focus on modern deep learning. It provides essential abstractions and building blocks for developing and shipping machine learning solutions with high iteration velocity.

TensorFlow

TensorFlow allows developers to create dataflow graphs—structures that describe how data moves through a graph, or a series of processing nodes. Each node in the graph represents a mathematical operation, and each connection or edge between nodes is a multidimensional data array, or *tensor*. TensorFlow provides all of this for the programmer by way of the Python language. Python is

easy to learn and work with, and provides convenient ways to express how high-level abstractions can be coupled together. Nodes and tensors in TensorFlow are Python objects, and TensorFlow applications are themselves Python applications.

The actual math operations, however, are not performed in Python. The libraries of transformations that are available through TensorFlow are written as high-performance C++ binaries. Python just directs traffic between the pieces, and provides high-level programming abstractions to hook them together.

TensorFlow applications can be run on most any target that's convenient: a local machine, a cluster in the cloud, iOS and Android devices, CPUs or GPUs. If you use Google's own cloud, you can run TensorFlow on Google's custom TensorFlow Processing Unit (TPU) silicon for further acceleration. The resulting models created by TensorFlow, though, can be deployed on most any device where they will be used to serve predictions.

TensorFlow 2.0, released in October 2019, revamped the framework in many ways based on user feedback, to make it easier to work with (e.g., by using the relatively simple Keras API for model training) and more performant.

PIL (Python Imaging Library)

In today's digital world, we come across lots of digital images. In case, we are working with Python programming language, it provides lot of image processing libraries to add image processing capabilities to digital images.

Some of the most common image processing libraries are: OpenCV, Python Imaging Library (PIL), Scikit-image, Pillow. However, in this tutorial, we are only focusing on Pillow module and will try to explore various capabilities of this module.

Pillow is built on top of PIL (Python Image Library). PIL is one of the important modules for image processing in Python. However, the PIL module is not supported since 2011 and doesn't support python 3.

Pillow module gives more functionalities, runs on all major operating system and support for python 3. It supports wide variety of images such as "jpeg", "png", "bmp", "gif", "ppm", "tiff".

You can do almost anything on digital images using pillow module. Apart from basic image processing functionality, including point operations, filtering images using built-in convolution kernels, and color space conversions.

The Pillow library contains all the basic image processing functionality. You can do image resizing, rotation and transformation.

Pillow module allows you to pull some statistics data out of image using histogram method, which later can be used for statistical analysis and automatic contrast enhancement.

Hunspell

Hunspell is a spell checker and morphological analyser designed for languages with rich morphology and complex word compounding and character encoding, originally designed for the Hungarian language.

Hunspell is based on MySpell and is backward-compatible with MySpell dictionaries. While MySpell uses a single-byte character encoding, Hunspell can use Unicode UTF-8-encoded dictionaries.

Hunspell is the spell checker of LibreOffice, OpenOffice.org, Mozilla Firefox 3 & Thunderbird, Google Chrome, and it is also used by proprietary software packages, like macOS, InDesign, memoQ, Opera and SDL Trados.

Main features:

- Extended support for language peculiarities; Unicode character encoding, compounding and complex morphology.
- Improved suggestion using n-gram similarity, rule and dictionary based pronunciation data.
- Morphological analysis, stemming and generation.
- Hunspell is based on MySpell and works also with MySpell dictionaries.
- C++ library under GPL/LGPL/MPL tri-license.

OpenCV

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

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching streetview images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe.

Machine Learning

Machine Learning is undeniably one of the most influential and powerful technologies in today's world. More importantly, we are far from seeing its full potential. There's no doubt, it will continue to be making headlines for the foreseeable future. Machine learning is a tool for turning information into knowledge. In the past 50 years, there has been an explosion of data. This mass of data is useless unless we analyze it and find the patterns hidden within. Machine learning techniques are used to automatically find the valuable underlying patterns within complex data that we would otherwise struggle to discover. The hidden patterns and knowledge about a problem can be used to predict future events and perform all kinds of complex decision making. Most of us are unaware that we already interact with Machine Learning every single day. Every time we Google something, listen to a song or even take a photo, Machine Learning is becoming part of the engine behind it, constantly learning and improving from every interaction. It's also behind world-changing advances like detecting cancer, creating new drugs and self-driving cars.

Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention. It is the scientific study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task. Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or infeasible to develop a conventional algorithm for effectively performing the task.

Machine learning is closely related to computational statistics, which focuses on making predictions using computers. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a field of study within machine learning, and focuses on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics.

Deep Learning

Deep learning is an artificial intelligence function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence (AI) that has networks capable of learning unsupervised from data that is unstructured or unlabelled. Also known as deep neural learning or deep neural network.

Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from a lamppost. It is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free speakers. Deep learning is getting lots of attention lately andfor good reason. It's achieving results that were not possible before.

In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human- level performance. Models are trained by using a large set of labelled data and neural network architectures that contain many layers.

In a word, accuracy. Deep learning achieves recognition accuracy at higher levels than ever before. Deep learning requires large amounts of labelled data. It requires substantial computing power. High-performance GPUs have a parallel architecture that is efficient for deep learning. When combined with clusters or cloud computing, this enables development teams to reduce training time for a deep learning network from weeks to hours or less. Recent advances in deep learning have improved to the point where deep learning outperforms humans in some tasks like classifying objects in images.

Deep learning architectures such as deep neural networks, deep belief networks, recurrent neural networks and convolutional neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs, where they have produced results comparable to and in some cases superior to human experts. Artificial Neural Networks (ANNs) were inspired by information processing and distributed communication nodes in biological systems. ANNs have various differences from biological brains. Specifically, neural networks tend to be static and symbolic, while the biological brain of most living organisms is dynamic and analogue.

A few applications of deep learning

Here are just a few examples of deep learning at work:

A self-driving vehicle slows down as it approaches a pedestrian crossing.

An ATM rejects a counterfeit bank note.

A smartphone app gives an instant translation of a street sign in a foreign language.

- Deep learning is especially well-suited to identification applications such as face recognition, text translation, voice recognition, and advanced driver assistance systems, including lane classification and traffic sign recognition.

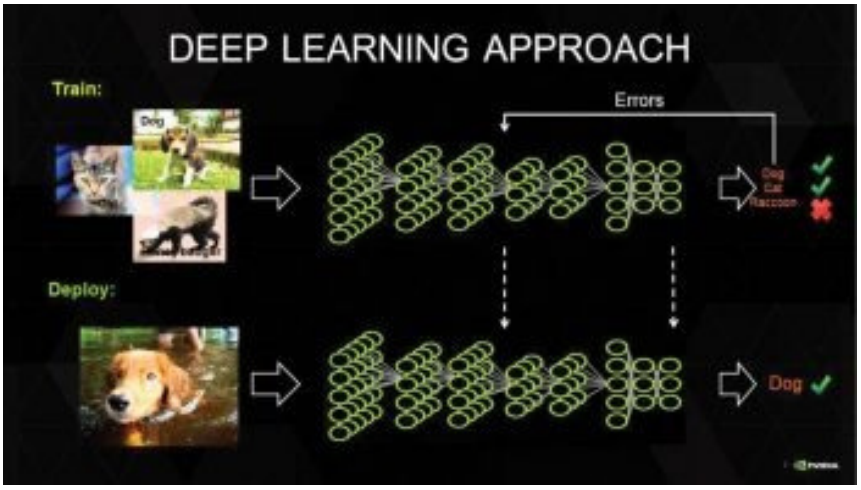


Figure 3.2: The Deep Learning Approach

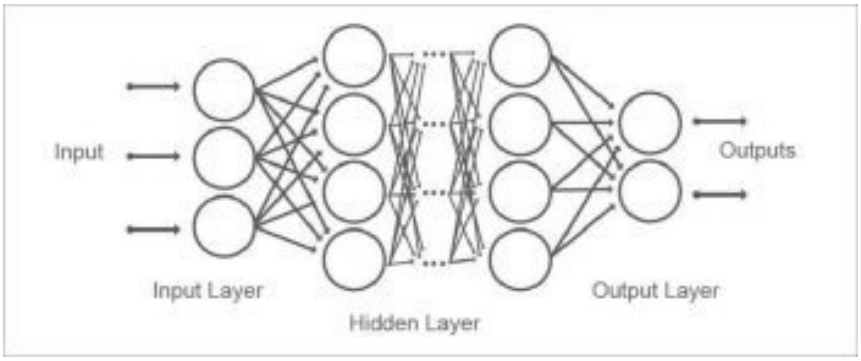


Figure 3.3: Deep learning Neural Networks

How a deep neural network learns

Let's say we have a set of images, where each image contains one of four different categories of objects, and we want the deep learning network to automatically recognise which object is in each image. We label the images in order to have training data for the network.

Using this training data, the network can then start to understand the object's specific features and associate them with the corresponding category. Each layer in the network takes in data from the previous layer, transforms it, and passes it on. The network increases the complexity and detail of what it is learning from layer to layer. Notice that the network learns directly from the data—we have no influence over what features are being learned.

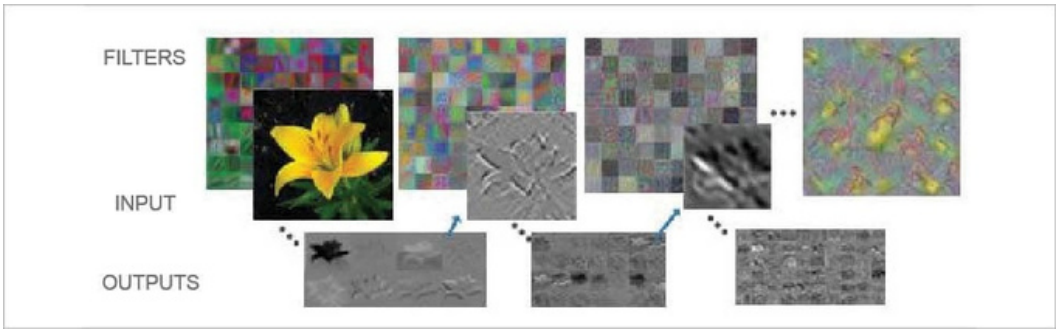


Figure 3.4: Image Processing using Deep Learning

Neural Networks

Neural networks are the workhorses of deep learning. Neural networks are multi-layer networks of neurons that we use to classify things, make predictions, etc. A neural network is a network or circuit of neurons, or in a modern sense, an artificial neural network, composed of artificial neurons or nodes. They consist of an input layer, multiple hidden layers, and an output layer. Every node in one layer is connected to every other node in the next layer. We make the network deeper by increasing the number of hidden layers. Thus, a neural network is either a biological neural network, made up of real biological neurons, or an artificial neural network, for solving artificial intelligence (AI) problems. The connections of the biological neuron are modelled as weights.

A positive weight reflects an excitatory connection, while negative values mean inhibitory connections. All inputs are modified by a weight and summed. This activity is referred as a linear combination. Finally, an activation function controls the amplitude of the output. For example, an acceptable range of output is usually between 0 and 1, or it could be −1 and 1. These artificial networks may be used for predictive modelling, adaptive control and applications where they can be trained via a dataset. Self-learning resulting from experience can occur within networks, which can derive conclusions from a complex and seemingly unrelated set of information.

Python 3.6

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications.

Some of the key advantages of learning Python:

- Python is Interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

- Python is Interactive: You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python is Object-Oriented: Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

7. 3 Algorithm

Convolutional Neural Network

A Convolution Neural Network is a type of artificial neural network used in image recognition and it detect the image with in the live video and divide the frames to the image into cells with assigning anchor box, Anchor Box means set of predefined boundary box of a certain calculation of height and width. The figure below illustrates the three layers configuration of CNN that is input layer, hidden layer and output layer. To generate region proposals it uses a selective search method. Anchor or region boxes are ranked by CNN network. CNN's are strong in image processing and video recognition that use Deep Learning to perform both generative and descriptive tasks, often using computer vision that include image and video recognition along with recommended system and natural language processing like python, java etc. Artificial Neural Networks are used in various classification tasks like image, audio, words.

Different types of Neural Networks are used for different purposes, for example for predicting the sequence of words we use Recurrent Neural Networks more precisely an LSTM, similarly for image classification we use Convolution Neural networks. In this blog, we are going to build a basic building block for CNN. Before diving into the Convolution Neural Network, let us first revisit some concepts of Neural Network.

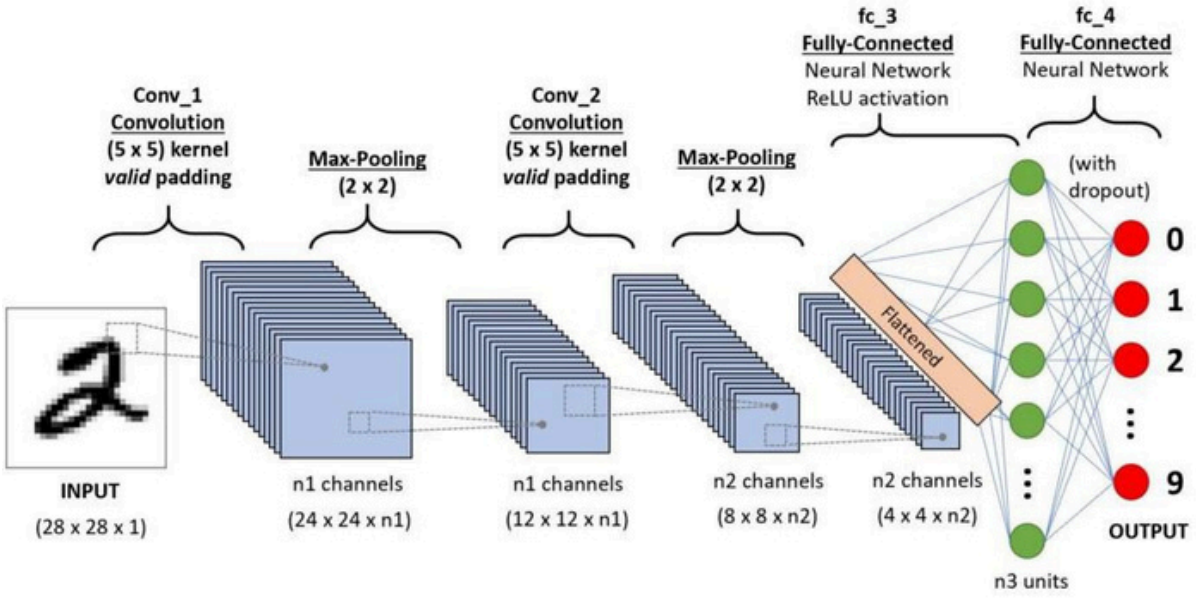
In a regular Neural Network there are three types of layers:

1. Input Layers: It's the layer in which we give input to our model. The number of neurons in this layer is equal to the total number of features in our data (number of pixels in the case of an image).
2. Hidden Layer: The input from the Input layer is then feed into the hidden layer. There can be many hidden layers depending upon our model and data size. Each hidden layer can have different numbers of neurons which are generally greater than the number of features.

A The output from each layer is computed by matrix multiplication of output of the previous layer with learnable weights of that layer and then by the addition of learnable biases followed by activation function which makes the network nonlinear.

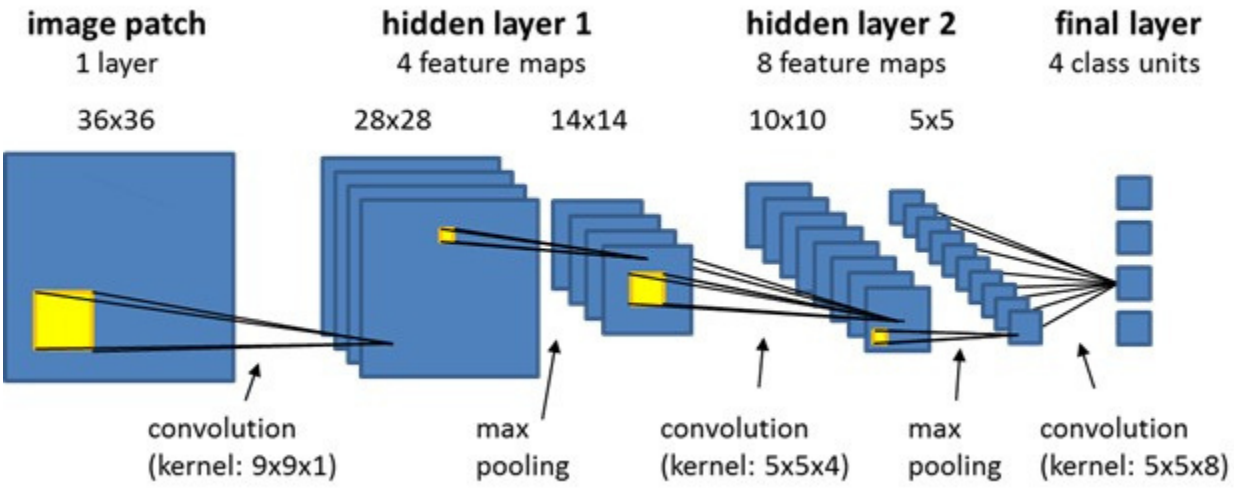
3. Output Layer: The output from the hidden layer is then fed into a logistic function like sigmoid or softmax which converts the output of each class into the probability score of each class.

The data is then fed into the model and output from each layer is obtained this step is called feedforward, we then calculate the error using an error function, some common error functions are cross-entropy, square loss error, etc. After that, we backpropagate into the model by calculating the derivatives. This step is called Backpropagation which basically is used to minimize the loss. Here's the basic python code for a neural network with random inputs and two hidden layers.



Unlike regular Neural Networks, in the layers of CNN, the neurons are arranged in 3 dimensions: width, height, depth. The neurons in a layer will only be connected to a small region of the layer (window size) before it, instead of all of the neurons in a fully-connected manner.

Moreover, the final output layer would have dimensions(number of classes), because by the end of the CNN architecture we will reduce the full image into a single vector of class scores.



1. Convolutional Layer

In convolution layer I have taken a small window size [typically of length 5*5] that extends to the depth of the input matrix. The layer consists of learnable filters of window size. During every iteration I slid the window by stride size [typically 1], and compute the dot product of filter entries and input values at a given position.

As we continue this process well create a 2-Dimensional activation matrix that gives the response of that matrix at every spatial position. That is, the network will learn filters that activate when they see some type of visual feature such as an edge of some orientation or a blotch of some colour.

2. Pooling Layer

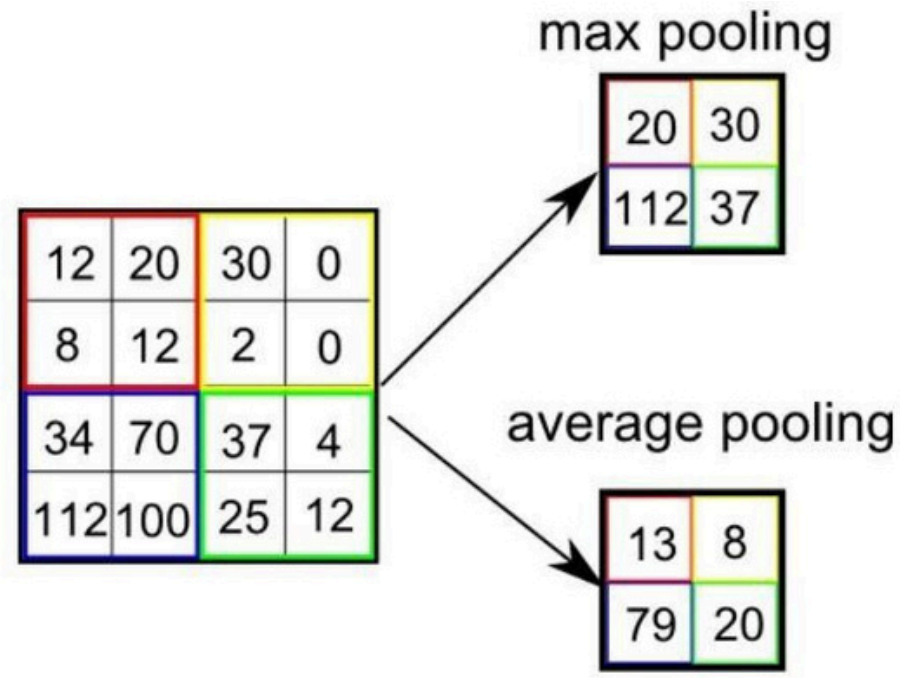
We use pooling layer to decrease the size of activation matrix and ultimately reduce the learnable parameters. There are two types of pooling:

a. Max Pooling

In max pooling we take a window size [for example window of size 2*2], and only taken the maximum of 4 values. Well lid this window and continue this process, so well finally get an activation matrix half of its original Size.

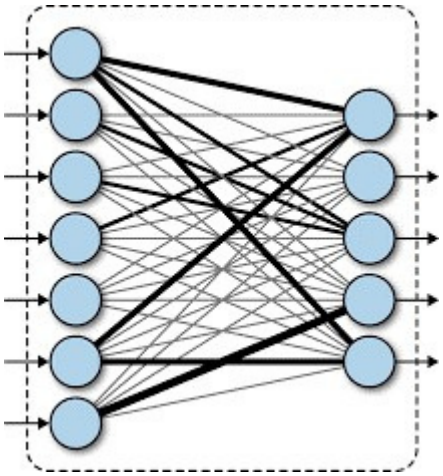
b. Average Pooling

In average pooling we take average of all Values in a window.



3. Fully Connected Layer

In convolution layer neurons are connected only to a local region, while in a fully connected region, well connect the all the inputs to neurons.



4. Final Output Layer

After getting values from fully connected layer, well connect them to final layer of neurons [having count equal to total number of classes], that will predict the probability of each image to be in different classes.

```
Epoch 1/5
1285/1285 [==============================] - 190s 148ms/step - loss: 0.0691 - accuracy: 0.9811 - val_loss: 0.0046 - val_accuracy: 0.9986
Epoch 2/5
1285/1285 [==============================] - 167s 130ms/step - loss: 0.0539 - accuracy: 0.9853 - val_loss: 0.0069 - val_accuracy: 0.9977
Epoch 3/5
1285/1285 [==============================] - 164s 128ms/step - loss: 0.0433 - accuracy: 0.9889 - val_loss: 0.0207 - val_accuracy: 0.9946
Epoch 4/5
1285/1285 [==============================] - 164s 127ms/step - loss: 0.0503 - accuracy: 0.9863 - val_loss: 0.0059 - val_accuracy: 0.9988
Epoch 5/5
1285/1285 [==============================] - 168s 131ms/step - loss: 0.0370 - accuracy: 0.9900 - val_loss: 0.0018 - val_accuracy: 0.9993
<tensorflow.python.keras.callbacks.History at 0x21004d31730>
```

7.4 Sample Code

```
from PIL import Image, ImageTk
import tkinter as tk
import cv2
import os
import numpy as np
from keras.models import model_from_json
import operator
import time
import sys, os
import matplotlib.pyplot as plt
from hunspell import Hunspell
from string import ascii_uppercase
import pyttsx3
language = 'en'
engine = pyttsx3.init()
class Application:
def __init__(self):
#self.directory = 'model'
self.hs = Hunspell('en_US')
self.vs = cv2.VideoCapture(0)
self.current_image = None
self.current_image2 = None

self.json_file = open("model\model-bw.json", "r")
self.model_json = self.json_file.read()
self.json_file.close()
self.loaded_model = model_from_json(self.model_json)
self.loaded_model.load_weights("model\model-bw.h5")
```

```
self.json_file_dru = open("model\model-bw_dru.json" , "r")
self.model_json_dru = self.json_file_dru.read()
self.json_file_dru.close()
self.loaded_model_dru  =  model_from_json(self.model_json_dru)
self.loaded_model_dru.load_weights("model\model-bw_dru.h5")

self.json_file_tkdi = open("model\model-bw_tkdi.json" , "r")
self.model_json_tkdi = self.json_file_tkdi.read()
self.json_file_tkdi.close()
self.loaded_model_tkdi = model_from_json(self.model_json_tkdi)
self.loaded_model_tkdi.load_weights("model\model-bw_tkdi.h5")

self.json_file_smn = open("model\model-bw_smn.json" , "r")
self.model_json_smn = self.json_file_smn.read()
self.json_file_smn.close()
self.loaded_model_smn = model_from_json(self.model_json_smn)
self.loaded_model_smn.load_weights("model\model-bw_smn.h5")

 self.ct = {}
self.ct['blank'] = 0
self.blank_flag = 0

for i in ascii_uppercase:
self.ct[i] = 0

print("Loaded model from disk")

self.root = tk.Tk()
self.root.title("Sign Language To Text Conversion")
```

```
self.root.protocol('WM_DELETE_WINDOW',  self.destructor)
self.root.geometry("900x900")

self.panel = tk.Label(self.root)
self.panel.place(x = 100, y = 10, width = 580, height = 580)

self.panel2 = tk.Label(self.root) # initialize image panel
self.panel2.place(x = 400, y = 65, width = 275, height = 275)

self.T = tk.Label(self.root)
self.T.place(x = 60, y = 5)
self.T.config(text = "Sign Language To Text Conversion", font = ("Courier", 30,
"bold"))

self.panel3 = tk.Label(self.root) # Current Symbol
self.panel3.place(x = 500, y = 540)

self.T1 = tk.Label(self.root)
self.T1.place(x = 10, y = 540)
self.T1.config(text = "Character :", font = ("Courier", 30, "bold"))

self.panel4 = tk.Label(self.root) # Word
self.panel4.place(x = 220, y = 595)
self.T2 = tk.Label(self.root)
self.T2.place(x = 10,y = 595)
self.T2.config(text = "Word :", font = ("Courier", 30, "bold"))

self.panel5 = tk.Label(self.root) # Sentence
self.panel5.place(x = 350, y = 645)
```

```
\self.T3 = tk.Label(self.root)
self.T3.place(x = 10, y = 645)
self.T3.config(text = "Sentence :",font = ("Courier", 30, "bold"))

self.T4 = tk.Label(self.root)
self.T4.place(x = 700, y = 150)
self.T4.config(text = "Suggestions :", fg = "red", font = ("Courier", 30, "bold"))
self.btcall = tk.Button(self.root,command = self.action_call,height = 0,width = 0)
self.btcall.config(text = "About",font = ("Courier",14))
self.btcall.place(x = 1280, y = 0)

self.bt1 = tk.Button(self.root, command = self.action1, height = 0, width = 0)
self.bt1.place(x = 1100, y = 150)

self.bt2 = tk.Button(self.root, command = self.action2, height = 0, width = 0)
self.bt2.place(x = 1100, y = 220)

self.bt3 = tk.Button(self.root, command = self.action3, height = 0, width = 0)
self.bt3.place(x = 1100, y = 290)

self.bt4=tk.Button(self.root, command=self.action4,height = 0,width = 0)
self.bt4.place(x = 1100,y=360)
# self.bt4.grid(row = bt1, column = 0, columnspan = 1, padx = 10, pady = 10, sticky =
tk.N)
self.bt5=tk.Button(self.root, command=self.action5,height = 0,width = 0)
self.bt5.place(x = 1100,y=430)
# self.bt5.grid(row = 5, column = 1, columnspan = 1, padx = 10, pady = 10, sticky =
tk.N)
```

```
\self.T3 = tk.Label(self.root)
self.T3.place(x = 10, y = 645)
self.T3.config(text = "Sentence :",font = ("Courier", 30, "bold"))

self.T4 = tk.Label(self.root)
self.T4.place(x = 700, y = 150)
self.T4.config(text = "Suggestions :", fg = "red", font = ("Courier", 30, "bold"))
self.btcall = tk.Button(self.root,command = self.action_call,height = 0,width = 0)
self.btcall.config(text = "About",font = ("Courier",14))
self.btcall.place(x = 1280, y = 0)

self.bt1 = tk.Button(self.root, command = self.action1, height = 0, width = 0)
self.bt1.place(x = 1100, y = 150)

self.bt2 = tk.Button(self.root, command = self.action2, height = 0, width = 0)
self.bt2.place(x = 1100, y = 220)

self.bt3 = tk.Button(self.root, command = self.action3, height = 0, width = 0)
self.bt3.place(x = 1100, y = 290)

self.bt4=tk.Button(self.root, command=self.action4,height = 0,width = 0)
self.bt4.place(x = 1100,y=360)
# self.bt4.grid(row = bt1, column = 0, columnspan = 1, padx = 10, pady = 10, sticky =
tk.N)
self.bt5=tk.Button(self.root, command=self.action5,height = 0,width = 0)
self.bt5.place(x = 1100,y=430)
# self.bt5.grid(row = 5, column = 1, columnspan = 1, padx = 10, pady = 10, sticky =
tk.N)
```

```
self.str = ""
self.word = " "
self.current_symbol = "Empty"
self.photo = "Empty"
self.video_loop()

def video_loop(self):
ok, frame = self.vs.read()
if ok:
cv2image = cv2.flip(frame, 1)
x1 = int(0.5*frame.shape[1])
y1 = 10
x2 = frame.shape[1]-10
y2 = int(0.5*frame.shape[1])
cv2.rectangle(frame, (x1-1, y1-1), (x2+1, y2+1), (255,0,0) ,1)
cv2image = cv2.cvtColor(cv2image, cv2.COLOR_BGR2RGBA)
self.current_image = Image.fromarray(cv2image)
imgtk = ImageTk.PhotoImage(image=self.current_image)
self.panel.imgtk = imgtk
self.panel.config(image=imgtk)
cv2image = cv2image[y1:y2, x1:x2]
gray = cv2.cvtColor(cv2image, cv2.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(gray,(5,5),2)
th3 =
cv2.adaptiveThreshold(blur,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH
_ BINARY_INV,11,2)
ret, res = cv2.threshold(th3, 70, 255, cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
self.predict(res)
self.current_image2 = Image.fromarray(res)
imgtk = ImageTk.PhotoImage(image=self.current_image2)
self.panel2.imgtk = imgtk
```

```
self.panel3.config(text=self.current_symbol,font=("Courier",50))
self.panel4.config(text=self.word,font=("Courier",40))
print(self.str)
engine.say(self.str)
engine.runAndWait()
self.panel5.config(text=self.str,font=("Courier",40))
predicts=self.hs.suggest(self.word)
if(len(predicts) > 0):
self.bt1.config(text=predicts[0],font = ("Courier",20))
else:
self.bt1.config(text="")
if(len(predicts) > 1):
self.bt2.config(text=predicts[1],font = ("Courier",20))
else:
self.bt2.config(text="")
if(len(predicts) > 2):
self.bt3.config(text=predicts[2],font = ("Courier",20))
else:
self.bt3.config(text="")
if(len(predicts) > 3):
self.bt4.config(text=predicts[3],font = ("Courier",20))
else:
self.bt4.config(text="")
if(len(predicts) > 4):
self.bt4.config(text=predicts[4],font = ("Courier",20))
else:
self.bt4.config(text="")
self.root.after(30, self.video_loop)
def predict(self,test_image):
test_image = cv2.resize(test_image, (128,128))
```

```
result = self.loaded_model.predict(test_image.reshape(1, 128, 128, 1))
result_dru = self.loaded_model_dru.predict(test_image.reshape(1 , 128 , 128 , 1))
result_tkdi = self.loaded_model_tkdi.predict(test_image.reshape(1 , 128 , 128 , 1)) result_smn
= self.loaded_model_smn.predict(test_image.reshape(1 , 128 , 128 , 1))
prediction={}
prediction['blank'] = result[0][0]
inde = 1
for i in ascii_uppercase:
prediction[i] = result[0][inde]
inde += 1

#LAYER 1
prediction = sorted(prediction.items(), key=operator.itemgetter(1), reverse=True)
self.current_symbol = prediction[0][0]
#LAYER 2
if(self.current_symbol == 'D' or self.current_symbol == 'R' or self.current_symbol ==
'U'):
prediction = {}
prediction['D'] = result_dru[0][0]
prediction['R'] = result_dru[0][1]
prediction['U'] = result_dru[0][2]
prediction = sorted(prediction.items(), key=operator.itemgetter(1), reverse=True)
self.current_symbol = prediction[0][0]

if(self.current_symbol == 'D' or self.current_symbol == 'I' or self.current_symbol == 'K'
or self.current_symbol == 'T'):
prediction = {}
prediction['D'] = result_tkdi[0][0]
prediction['I'] = result_tkdi[0][1]
prediction['K'] = result_tkdi[0][2]
prediction['T'] = result_tkdi[0][3]
```

```
prediction = sorted(prediction.items(), key=operator.itemgetter(1), reverse=True)
self.current_symbol = prediction[0][0]
if(self.current_symbol == 'M' or self.current_symbol == 'N' or self.current_symbol ==
'S'):
prediction1 = {}
prediction1['M'] = result_smn[0][0]
prediction1['N'] = result_smn[0][1]
prediction1['S'] = result_smn[0][2]
prediction1 = sorted(prediction1.items(), key=operator.itemgetter(1), reverse=True)
if(prediction1[0][0] == 'S'):
self.current_symbol = prediction1[0][0]
else:
self.current_symbol = prediction[0][0]
if(self.current_symbol == 'blank'):
for i in ascii_uppercase:
self.ct[i] = 0
self.ct[self.current_symbol] += 1
if(self.ct[self.current_symbol] > 60):
for i in ascii_uppercase:
if i == self.current_symbol:
continue
tmp = self.ct[self.current_symbol] - self.ct[i]
if tmp < 0:
tmp *= -1
if tmp <= 20:
self.ct['blank'] = 0
for i in ascii_uppercase:
self.ct[i] = 0
return
self.ct['blank'] = 0
for i in ascii_uppercase:
self.ct[i] = 0
```

```
 if self.current_symbol == 'blank':
if self.blank_flag == 0:
self.blank_flag = 1
if len(self.str) > 0:
self.str += " "
self.str += self.word
self.word = ""
else:
if(len(self.str) > 16):
self.str = ""
self.blank_flag = 0
self.word += self.current_symbol

def action1(self):
predicts=self.hs.suggest(self.word)
if(len(predicts) > 0):
self.word=""
self.str+=" "
self.str+=predicts[0]
def action2(self):
predicts=self.hs.suggest(self.word)
if(len(predicts) > 1):
self.word=""
self.str+=" "
self.str+=predicts[1]
def action3(self):
predicts=self.hs.suggest(self.word)
if(len(predicts) > 2):
self.word=""
self.str+=" "
```

```
self.str+=predicts[2]
def action4(self):
predicts=self.hs.suggest(self.word)
if(len(predicts) > 3):
self.word=""
self.str+=" "
self.str+=predicts[3]
def action5(self):
predicts=self.hs.suggest(self.word)
if(len(predicts) > 4):
self.word=""
self.str+=" "
self.str+=predicts[4]
def destructor(self):
print("Closing Application...")
self.root.destroy()
self.vs.release()
cv2.destroyAllWindows()

def destructor1(self):

 print("Closing Application...")
self.root1.destroy()
def action_call(self) :
self.root1 = tk.Toplevel(self.root)
self.root1.title("About")
self.root1.protocol('WM_DELETE_WINDOW', self.destructor1)
self.root1.geometry("900x900")

self.tx = tk.Label(self.root1)
self.tx.place(x = 330,y = 20)
```

```
self.tx.config(text = "Efforts By", fg="red", font = ("Courier",30,"bold"))

self.photo1 = tk.PhotoImage(file='Pictures/ravi.png')
self.w1 = tk.Label(self.root1, image = self.photo1)
self.w1.place(x = 20, y = 105)
self.tx6 = tk.Label(self.root1)
self.tx6.place(x = 20,y = 250)
self.tx6.config(text = "Mani Sai\n18UJ1A1233", font = ("Courier",15,"bold"))

#self.photo2 = tk.PhotoImage(file='Pictures/nitin.png')
#self.w2 = tk.Label(self.root1, image = self.photo2)
#self.w2.place(x = 200, y = 105)
#self.tx2 = tk.Label(self.root1)
#self.tx2.place(x = 200,y = 250)
#self.tx2.config(text = "Abhinav\18UJ1A1211", font = ("Courier",15,"bold"))

self.photo3 = tk.PhotoImage(file='Pictures/luv.png')
self.w3 = tk.Label(self.root1, image = self.photo3)
self.w3.place(x = 380, y = 105)
self.tx3 = tk.Label(self.root1)
self.tx3.place(x = 385,y = 250)
self.tx3.config(text = "Supriya\n18UJ1A1221", font = ("Courier",15,"bold"))
#self.photo4 = tk.PhotoImage(file='Pictures/sheldon.png')
#self.w4 = tk.Label(self.root1, image = self.photo4)
#self.w4.place(x = 560, y = 105)
#self.tx4 = tk.Label(self.root1)
#self.tx4.place(x = 560,y = 250)
#self.tx4.config(text = "Sandeep\n18UJ1A1234", font = ("Courier",15,"bold"))
```

```
self.photo5 = tk.PhotoImage(file='Pictures/sid.png')
self.w5 = tk.Label(self.root1, image = self.photo5)
self.w5.place(x = 740, y = 105)
self.tx5 = tk.Label(self.root1)
self.tx5.place(x = 745,y = 250)
self.tx5.config(text = "Vaibhavi\n18UJ1A112322", font = ("Courier",15,"bold"))

self.tx7 = tk.Label(self.root1)
self.tx7.place(x = 170,y = 340)
self.tx7.config(text = "Under the supervision of", fg="red", font = ("Courier",30,"bold"))

self.photo6 = tk.PhotoImage(file='Pictures/sir.png')
self.w6 = tk.Label(self.root1, image = self.photo6)
self.w6.place(x = 350, y = 400)
self.tx6 = tk.Label(self.root1)
self.tx6.place(x = 280,y = 650)
self.tx6.config(text = "P. THIMMA REDDY", font = ("Courier",30,"bold"))

print("Starting Application...")
pba = Application()
pba.root.mainloop()

# Collect data
import cv2
import numpy as np
import os
import string
# Create the directory structure
if not os.path.exists("data"):
os.makedirs("data")
```

```
if not os.path.exists("data/train"):
os.makedirs("data/train")
if not os.path.exists("data/test"):
os.makedirs("data/test")
for i in range(3):
if not os.path.exists("data/train/" + str(i)):
os.makedirs("data/train/"+str(i))
if not os.path.exists("data/test/" + str(i)):
os.makedirs("data/test/"+str(i))

for i in string.ascii_uppercase:
if not os.path.exists("data/train/" + i):
os.makedirs("data/train/"+i)
if not os.path.exists("data/test/" + i):
os.makedirs("data/test/"+i)

# Train or test
mode = 'train'
directory = 'data/'+mode+'/'
minValue = 70

cap = cv2.VideoCapture(0)
interrupt = -1

while True:
_, frame = cap.read()
# Simulating mirror image
frame = cv2.flip(frame, 1)
# Getting count of existing images
count = {
```

```
'zero': len(os.listdir(directory+"/0")),
'one': len(os.listdir(directory+"/1")),
'two': len(os.listdir(directory+"/2")),
# 'three': len(os.listdir(directory+"/3")),
#  'four': len(os.listdir(directory+"/4")),
# 'five': len(os.listdir(directory+"/5")),
# 'six': len(os.listdir(directory+"/6")),
'a': len(os.listdir(directory+"/A")),
'b': len(os.listdir(directory+"/B")),
'c': len(os.listdir(directory+"/C")),
'd': len(os.listdir(directory+"/D")),
'e': len(os.listdir(directory+"/E")),
'f': len(os.listdir(directory+"/F")),
'g': len(os.listdir(directory+"/G")),
'h': len(os.listdir(directory+"/H")),
'i': len(os.listdir(directory+"/I")),
'j': len(os.listdir(directory+"/J")),
'k': len(os.listdir(directory+"/K")),
'l': len(os.listdir(directory+"/L")),
'm': len(os.listdir(directory+"/M")),
'n': len(os.listdir(directory+"/N")),
'o': len(os.listdir(directory+"/O")),
'p': len(os.listdir(directory+"/P")),
'q': len(os.listdir(directory+"/Q")),
'r': len(os.listdir(directory+"/R")),
's': len(os.listdir(directory+"/S")),
't': len(os.listdir(directory+"/T")),
'u': len(os.listdir(directory+"/U")),
'v': len(os.listdir(directory+"/V")),
'w': len(os.listdir(directory+"/W")),
```

```
"x':  len(os.listdir(directory+"/X")),
'y':   len(os.listdir(directory+"/Y")),
'z': len(os.listdir(directory+"/Z"))
}

# Printing the count in each set to the screen
# cv2.putText(frame, "MODE : "+mode, (10, 50), cv2.FONT_HERSHEY_PLAIN, 1,
(0,255,255), 1)
# cv2.putText(frame, "IMAGE COUNT", (10, ), cv2.FONT_HERSHEY_PLAIN, 1,
(0,255,255), 1)
cv2.putText(frame, "ZERO : "+str(count['zero']), (10, 70),
cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
cv2.putText(frame, "ONE : "+str(count['one']), (10, 80), cv2.FONT_HERSHEY_PLAIN,
1, (0,255,255), 1)
cv2.putText(frame, "TWO : "+str(count['two']), (10, 90), cv2.FONT_HERSHEY_PLAIN,
1, (0,255,255), 1)
# cv2.putText(frame, "THREE : "+str(count['three']), (10, 180),
cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
# cv2.putText(frame, "FOUR : "+str(count['four']), (10, 200),
cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
# cv2.putText(frame, "FIVE : "+str(count['five']), (10, 220),
cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
# cv2.putText(frame, "SIX : "+str(count['six']), (10, 230), cv2.FONT_HERSHEY_PLAIN,
1, (0,255,255), 1)
cv2.putText(frame, "a : "+str(count['a']), (10, 100), cv2.FONT_HERSHEY_PLAIN, 1,
(0,255,255), 1)
cv2.putText(frame, "b : "+str(count['b']), (10, 110), cv2.FONT_HERSHEY_PLAIN, 1,
(0,255,255), 1)
cv2.putText(frame, "c : "+str(count['c']), (10, 120), cv2.FONT_HERSHEY_PLAIN, 1,
(0,255,255), 1)
cv2.putText(frame, "d : "+str(count['d']), (10, 130), cv2.FONT_HERSHEY_PLAIN, 1,
(0,255,255), 1)
```

```
cv2.putText(frame, "e : "+str(count['e']), (10, 140), cv2.FONT_HERSHEY_PLAIN, 1,
(0,255,255), 1)
cv2.putText(frame, "f : "+str(count['f']), (10, 150), cv2.FONT_HERSHEY_PLAIN, 1,
(0,255,255), 1)
cv2.putText(frame, "g : "+str(count['g']), (10, 160), cv2.FONT_HERSHEY_PLAIN, 1,
(0,255,255), 1)
cv2.putText(frame, "h : "+str(count['h']), (10, 170), cv2.FONT_HERSHEY_PLAIN, 1,
(0,255,255), 1)
cv2.putText(frame, "i : "+str(count['i']), (10, 180), cv2.FONT_HERSHEY_PLAIN, 1,
(0,255,255), 1)
cv2.putText(frame, "k : "+str(count['k']), (10, 190), cv2.FONT_HERSHEY_PLAIN, 1,
(0,255,255), 1)
cv2.putText(frame, "l : "+str(count['l']), (10, 200), cv2.FONT_HERSHEY_PLAIN, 1,
(0,255,255), 1)
cv2.putText(frame, "m : "+str(count['m']), (10, 210), cv2.FONT_HERSHEY_PLAIN, 1,
(0,255,255), 1)
cv2.putText(frame, "n : "+str(count['n']), (10, 220), cv2.FONT_HERSHEY_PLAIN, 1,
(0,255,255), 1)
cv2.putText(frame, "o : "+str(count['o']), (10, 230), cv2.FONT_HERSHEY_PLAIN, 1,
(0,255,255), 1)
cv2.putText(frame, "p : "+str(count['p']), (10, 240), cv2.FONT_HERSHEY_PLAIN, 1,
(0,255,255), 1)
cv2.putText(frame, "q : "+str(count['q']), (10, 250), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255),
1)
cv2.putText(frame, "r : "+str(count['r']), (10, 260), cv2.FONT_HERSHEY_PLAIN, 1,
(0,255,255), 1)
cv2.putText(frame, "s : "+str(count['s']), (10, 270), cv2.FONT_HERSHEY_PLAIN, 1,
(0,255,255), 1)
cv2.putText(frame, "t : "+str(count['t']), (10, 280), cv2.FONT_HERSHEY_PLAIN, 1,
(0,255,255), 1)
```

```
cv2.putText(frame, "u : "+str(count['u']), (10, 290), cv2.FONT_HERSHEY_PLAIN, 1,
(0,255,255), 1)
cv2.putText(frame, "v : "+str(count['v']), (10, 300), cv2.FONT_HERSHEY_PLAIN, 1,
(0,255,255), 1)
cv2.putText(frame, "w : "+str(count['w']), (10, 310), cv2.FONT_HERSHEY_PLAIN, 1,
(0,255,255), 1)
cv2.putText(frame, "x : "+str(count['x']), (10, 320), cv2.FONT_HERSHEY_PLAIN, 1,
(0,255,255), 1)
cv2.putText(frame, "y : "+str(count['y']), (10, 330), cv2.FONT_HERSHEY_PLAIN, 1,
(0,255,255), 1)
cv2.putText(frame, "z : "+str(count['z']), (10, 340), cv2.FONT_HERSHEY_PLAIN, 1,
(0,255,255), 1)
# Coordinates of the ROI
x1 = int(0.5*frame.shape[1])
y1 = 10
x2 = frame.shape[1]-10
y2 = int(0.5*frame.shape[1])
# Drawing the ROI
# The increment/decrement by 1 is to compensate for the bounding box
cv2.rectangle(frame, (220-1, 9), (620+1, 419), (255,0,0) ,1)
# Extracting the ROI
roi = frame[10:410, 220:520]
# roi = cv2.resize(roi, (64, 64))
cv2.imshow("Frame", frame)
gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(gray,(5,5),2)
# #blur = cv2.bilateralFilter(roi,9,75,75)
th3 =
cv2.adaptiveThreshold(blur,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH
_ BINARY_INV,11,2)
```

```
ret, test_image = cv2.threshold(th3, minValue, 255,
cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
#time.sleep(5)
#cv2.imwrite("/home/rc/Downloads/soe/im1.jpg", roi)
#test_image = func("/home/rc/Downloads/soe/im1.jpg")

test_image = cv2.resize(test_image, (300,300))

cv2.imshow("test", test_image)

#_, mask = cv2.threshold(mask, 200, 255, cv2.THRESH_BINARY)
#kernel = np.ones((1, 1), np.uint8)
#img = cv2.dilate(mask, kernel, iterations=1)
#img = cv2.erode(mask, kernel, iterations=1)
# do the processing after capturing the image!
# roi = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
# _, roi = cv2.threshold(roi, 120, 255, cv2.THRESH_BINARY)

##gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
##cv2.imshow("GrayScale", gray)
##blur = cv2.GaussianBlur(gray,(5,5),2)

#blur = cv2.bilateralFilter(roi,9,75,75)

##th3 =
cv2.adaptiveThreshold(frame,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH
_BINARY_INV,11,2)
##ret, res = cv2.threshold(th3, minValue, 255,
cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
# cv2.imshow("ROI", roi)
#roi = frame
```

```
interrupt = cv2.waitKey(10)
if interrupt & 0xFF == 27: # esc key
break
if interrupt & 0xFF == ord('0'):
cv2.imwrite(directory+'0/'+str(count['zero'])+'.jpg', roi)
if interrupt & 0xFF == ord('1'):
cv2.imwrite(directory+'1/'+str(count['one'])+'.jpg', roi)
if interrupt & 0xFF == ord('2'):
cv2.imwrite(directory+'2/'+str(count['two'])+'.jpg', roi)
# if interrupt & 0xFF == ord('3'):
#  cv2.imwrite(directory+'3/'+str(count['three'])+'.jpg',  roi)
# if interrupt & 0xFF == ord('4'):
#   cv2.imwrite(directory+'4/'+str(count['four'])+'.jpg',   roi)
# if interrupt & 0xFF == ord('5'):
# cv2.imwrite(directory+'5/'+str(count['five'])+'.jpg', roi)
# if interrupt & 0xFF == ord('6'):
# cv2.imwrite(directory+'6/'+str(count['six'])+'.jpg', roi)
if interrupt & 0xFF == ord('a'):
cv2.imwrite(directory+'A/'+str(count['a'])+'.jpg', roi)
if interrupt & 0xFF == ord('b'):
cv2.imwrite(directory+'B/'+str(count['b'])+'.jpg', roi)
if interrupt & 0xFF == ord('c'):
cv2.imwrite(directory+'C/'+str(count['c'])+'.jpg', roi)
if interrupt & 0xFF == ord('d'):
cv2.imwrite(directory+'D/'+str(count['d'])+'.jpg', roi)
if interrupt & 0xFF == ord('e'):
cv2.imwrite(directory+'E/'+str(count['e'])+'.jpg', roi)
if interrupt & 0xFF == ord('f'):
cv2.imwrite(directory+'F/'+str(count['f'])+'.jpg', roi)
if interrupt & 0xFF == ord('g'):
```

```
cv2.imwrite(directory+'G/'+str(count['g'])+'.jpg', roi)
if interrupt & 0xFF == ord('h'):
cv2.imwrite(directory+'H/'+str(count['h'])+'.jpg', roi)
if interrupt & 0xFF == ord('i'):
cv2.imwrite(directory+'I/'+str(count['i'])+'.jpg', roi)
if interrupt & 0xFF == ord('j'):
cv2.imwrite(directory+'J/'+str(count['j'])+'.jpg', roi)
if interrupt & 0xFF == ord('k'):
cv2.imwrite(directory+'K/'+str(count['k'])+'.jpg', roi)
if interrupt & 0xFF == ord('l'):
cv2.imwrite(directory+'L/'+str(count['l'])+'.jpg', roi)
if interrupt & 0xFF == ord('m'):
cv2.imwrite(directory+'M/'+str(count['m'])+'.jpg', roi)
if interrupt & 0xFF == ord('n'):
cv2.imwrite(directory+'N/'+str(count['n'])+'.jpg', roi)
if interrupt & 0xFF == ord('o'):
cv2.imwrite(directory+'O/'+str(count['o'])+'.jpg', roi)
if interrupt & 0xFF == ord('p'):
cv2.imwrite(directory+'P/'+str(count['p'])+'.jpg', roi)
if interrupt & 0xFF == ord('q'):
cv2.imwrite(directory+'Q/'+str(count['q'])+'.jpg', roi)
if interrupt & 0xFF == ord('r'):
cv2.imwrite(directory+'R/'+str(count['r'])+'.jpg', roi)
if interrupt & 0xFF == ord('s'):
cv2.imwrite(directory+'S/'+str(count['s'])+'.jpg', roi)
if interrupt & 0xFF == ord('t'):
cv2.imwrite(directory+'T/'+str(count['t'])+'.jpg', roi)
if interrupt & 0xFF == ord('u'):
cv2.imwrite(directory+'U/'+str(count['u'])+'.jpg', roi)
if interrupt & 0xFF == ord('v'):
```

```
cv2.imwrite(directory+'V/'+str(count['v'])+'.jpg', roi)
if interrupt & 0xFF == ord('w'):
cv2.imwrite(directory+'W/'+str(count['w'])+'.jpg', roi)
if interrupt & 0xFF == ord('x'):
cv2.imwrite(directory+'X/'+str(count['x'])+'.jpg', roi)
if interrupt & 0xFF == ord('y'):
cv2.imwrite(directory+'Y/'+str(count['y'])+'.jpg', roi)
if interrupt & 0xFF == ord('z'):
cv2.imwrite(directory+'Z/'+str(count['z'])+'.jpg', roi)

cap.release()
cv2.destroyAllWindows()
"""
d = "old-data/test/0"
newd = "data/test/0"
for walk in os.walk(d):
for file in walk[2]:
roi = cv2.imread(d+"/"+file)
roi = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
_, mask = cv2.threshold(roi, 120, 255, cv2.THRESH_BINARY)
cv2.imwrite(newd+"/"+file, mask)
"""

# image processing
import numpy as np
import cv2
minValue = 70
def func(path):
frame = cv2.imread(path)
```

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(gray,(5,5),2)

th3 =
cv2.adaptiveThreshold(blur,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH
_ BINARY_INV,11,2)
ret, res = cv2.threshold(th3, minValue, 255,
cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
return res

# Preprocessing
import numpy as np
import cv2
import os
import csv
from image_processing import func if not os.path.exists("data2"):
os.makedirs("data2")
if not os.path.exists("data2/train"):
os.makedirs("dataluv/train")
if not os.path.exists("data2/test"):
os.makedirs("data2/test")
path="train"
path1 = "data2"
a=['label']

for i in range(64*64):
a.append("pixel"+str(i))

#outputLine = a.tolist()
label=0
```

```
var = 0
c1 = 0
c2 = 0

for (dirpath,dirnames,filenames) in os.walk(path):
for dirname in dirnames:
print(dirname)
for(direcpath,direcnames,files) in os.walk(path+"/"+dirname):
if not os.path.exists(path1+"/train/"+dirname):
os.makedirs(path1+"/train/"+dirname)
if not os.path.exists(path1+"/test/"+dirname):
os.makedirs(path1+"/test/"+dirname)
# num=0.75*len(files)
num = 100000000000000000
i=0
for file in files:
var+=1
actual_path=path+"/"+dirname+"/"+file
actual_path1=path1+"/"+"train/"+dirname+"/"+file
actual_path2=path1+"/"+"test/"+dirname+"/"+file
img = cv2.imread(actual_path, 0)
bw_image = func(actual_path)
if i<num:
c1 += 1
cv2.imwrite(actual_path1 , bw_image)
else:
c2 += 1
cv2.imwrite(actual_path2 , bw_image)

i=i+1
```

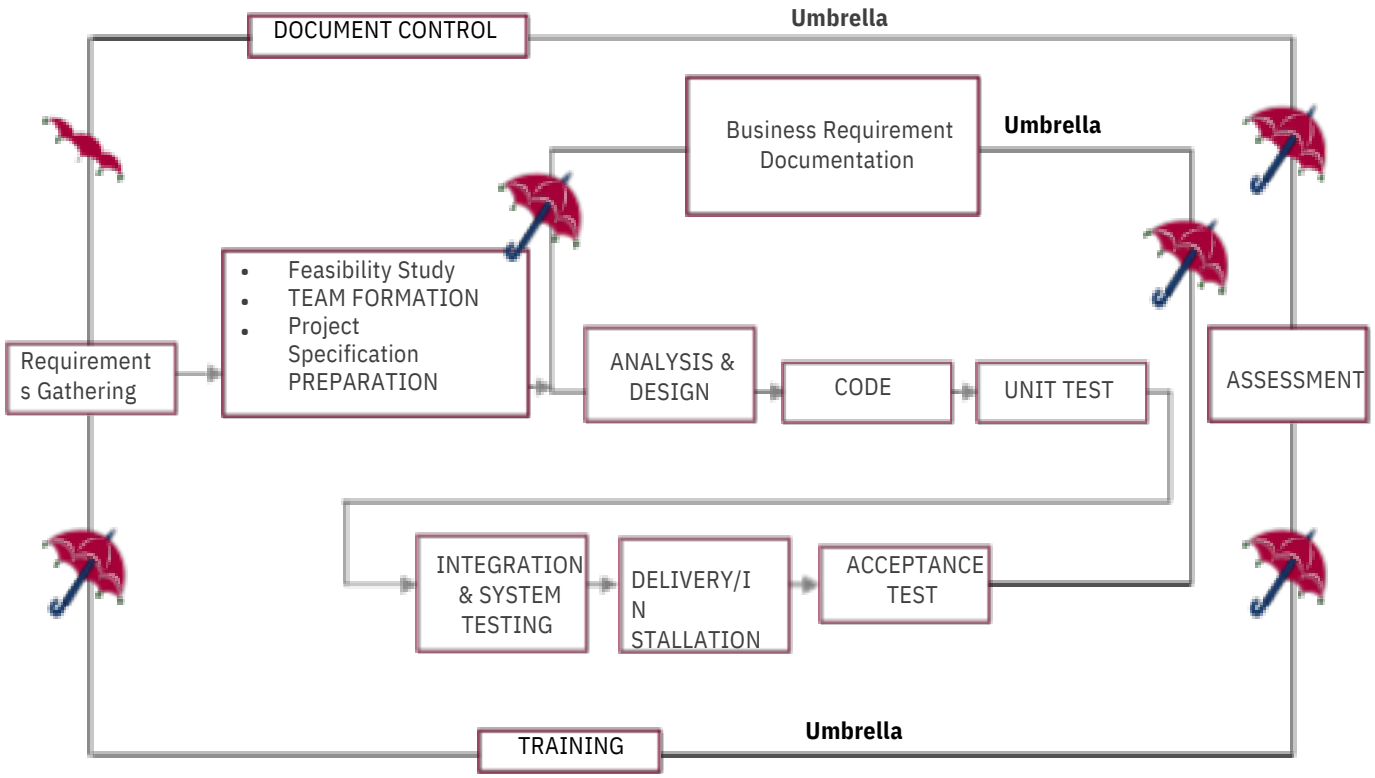
```
label=label+1
print(var)
print(c1)
print(c2)

# Train
# Importing the Keras libraries and packages
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 , Dropout
import os os.environ["CUDA_VISIBLE_DEVICES"] = "1" sz = 128
# Step 1 - Building the CNN

# Initializing the CNN
classifier = Sequential()
# First convolution layer and pooling
classifier.add(Convolution2D(32, (3, 3), input_shape=(sz, sz, 1), activation='relu'))
classifier.add(MaxPooling2D(pool_size=(2, 2)))
# Second convolution layer and pooling
classifier.add(Convolution2D(32, (3, 3), activation='relu'))
# input_shape is going to be the pooled feature maps from the previous convolution layer
classifier.add(MaxPooling2D(pool_size=(2, 2)))
#classifier.add(Convolution2D(32, (3, 3), activation='relu'))
# input_shape is going to be the pooled feature maps from the previous convolution layer
#classifier.add(MaxPooling2D(pool_size=(2, 2)))
# Flattening the layers
classifier.add(Flatten())
```

```
# Adding a fully connected layer
classifier.add(Dense(units=128, activation='relu'))
classifier.add(Dropout(0.40))
classifier.add(Dense(units=96, activation='relu'))
classifier.add(Dropout(0.40))
classifier.add(Dense(units=64, activation='relu'))
classifier.add(Dense(units=27, activation='softmax')) # softmax for more than 2
# Compiling the CNN
classifier.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy']) #
categorical_crossentropy for more than 2

# Step 2 - Preparing the train/test data and training the model
classifier.summary()
# Code copied from - https://keras.io/preprocessing/image/
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(
rescale=1./255,
shear_range=0.2,
zoom_range=0.2,
horizontal_flip=True)
test_datagen = ImageDataGenerator(rescale=1./255)
training_set = train_datagen.flow_from_directory('data2/train',
target_size=(sz, sz),
batch_size=10,
color_mode='grayscale',
class_mode='categorical')
test_set = test_datagen.flow_from_directory('data2/test',
target_size=(sz , sz),
batch_size=10,
color_mode='grayscale',
```

```
class_mode='categorical')
classifier.fit_generator(
training_set,
steps_per_epoch=12841, # No of images in training set
epochs=5,
validation_data=test_set,
validation_steps=4268)# No of images in test set


# Saving the model
model_json = classifier.to_json()
with open("model-bw.json", "w") as json_file:
json_file.write(model_json)
print('Model Saved')
classifier.save_weights('model-bw.h5')
print('Weights saved')
```

CHAPTER 8: SYSTEM TESTING

SDLC (Umbrella Model) :



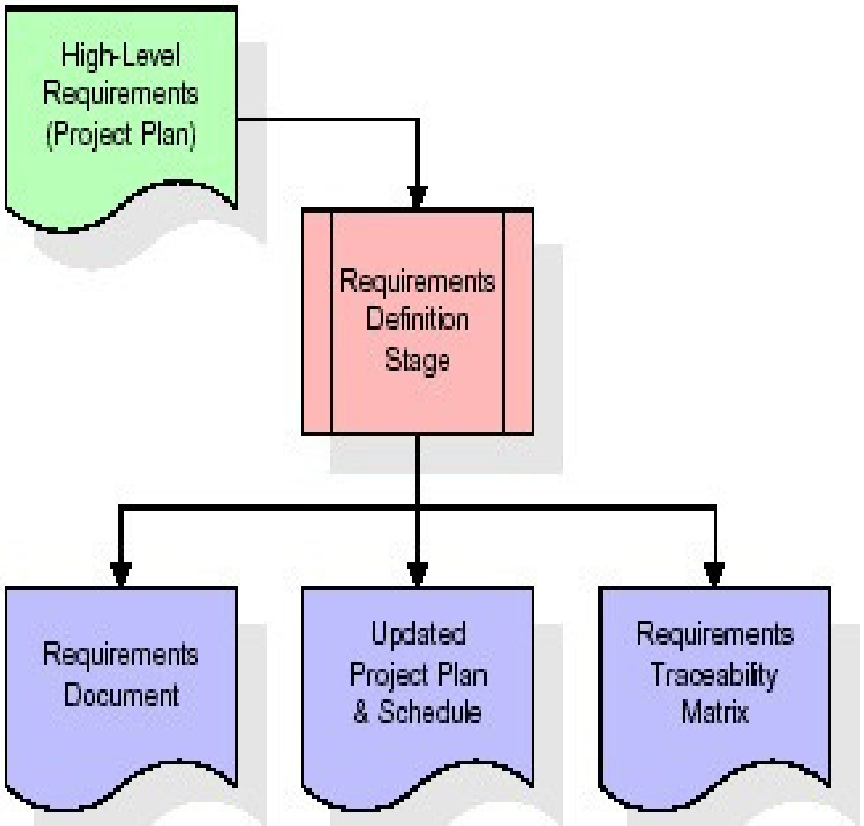
SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

8.1 Stages in SDLC

- Requirement Gathering Analysis
- Designing
- Coding
- Testing
- Maintenance

Requirements Gathering Stage

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.



These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are not included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

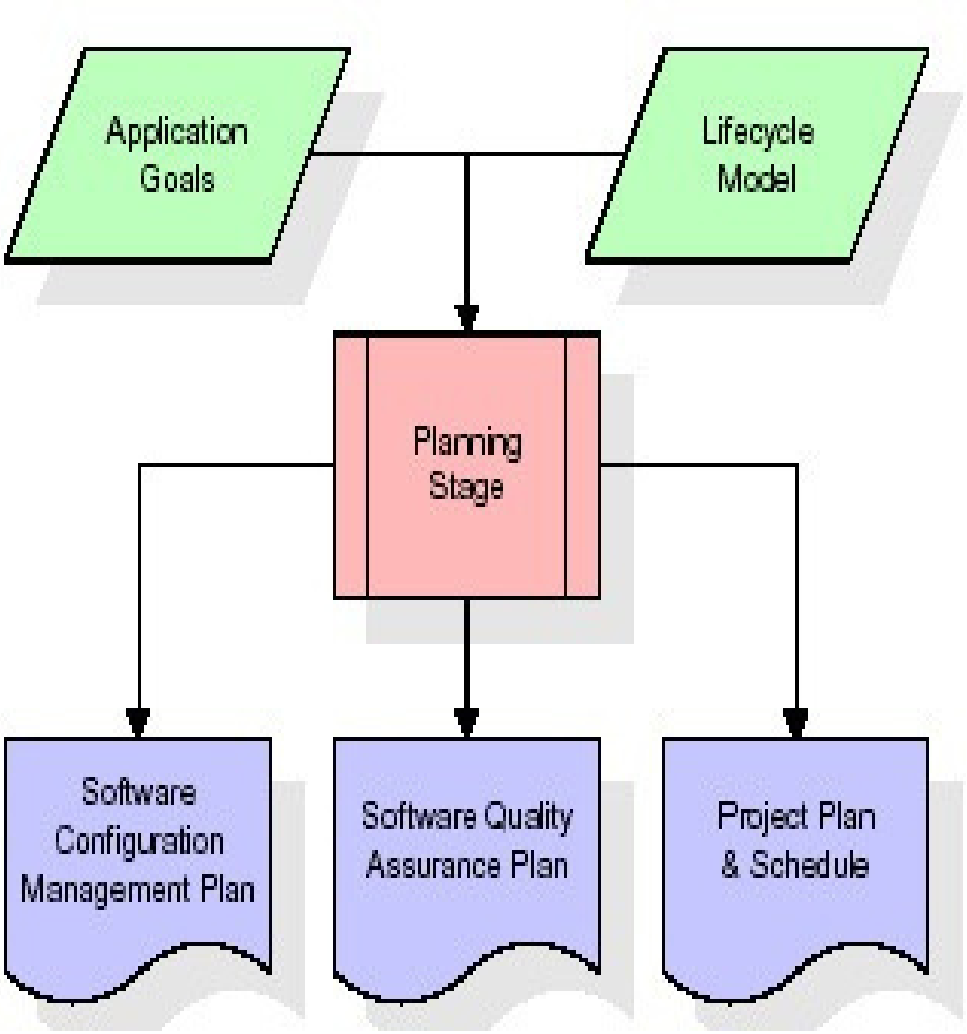
In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term requirements traceability.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

- Feasibility study is all about identification of problems in a project.
- No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
- Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

Analysis Stage

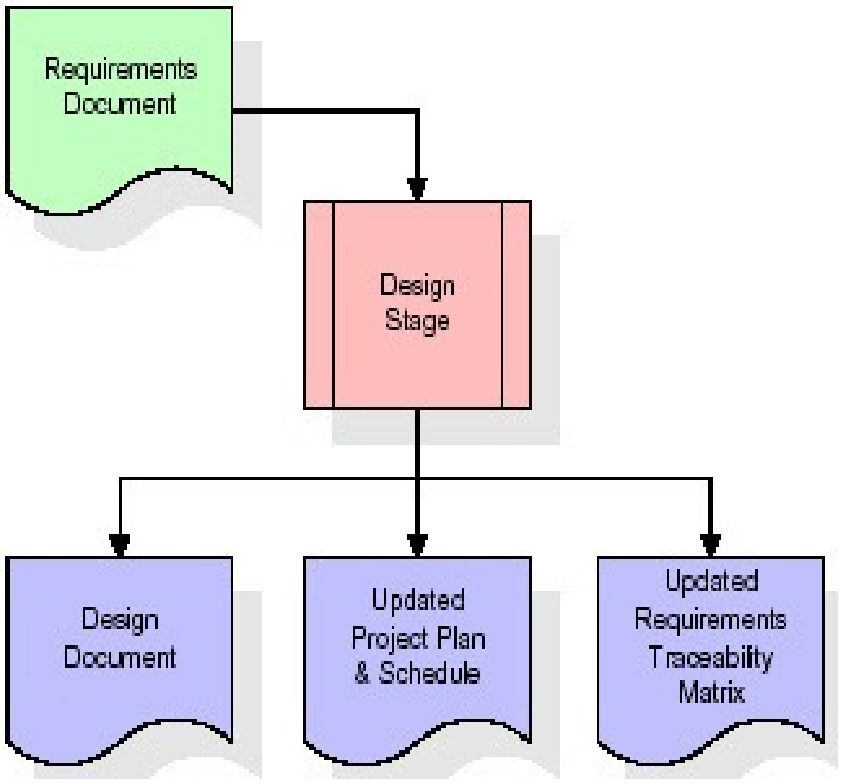
The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.



The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

Designing Stage

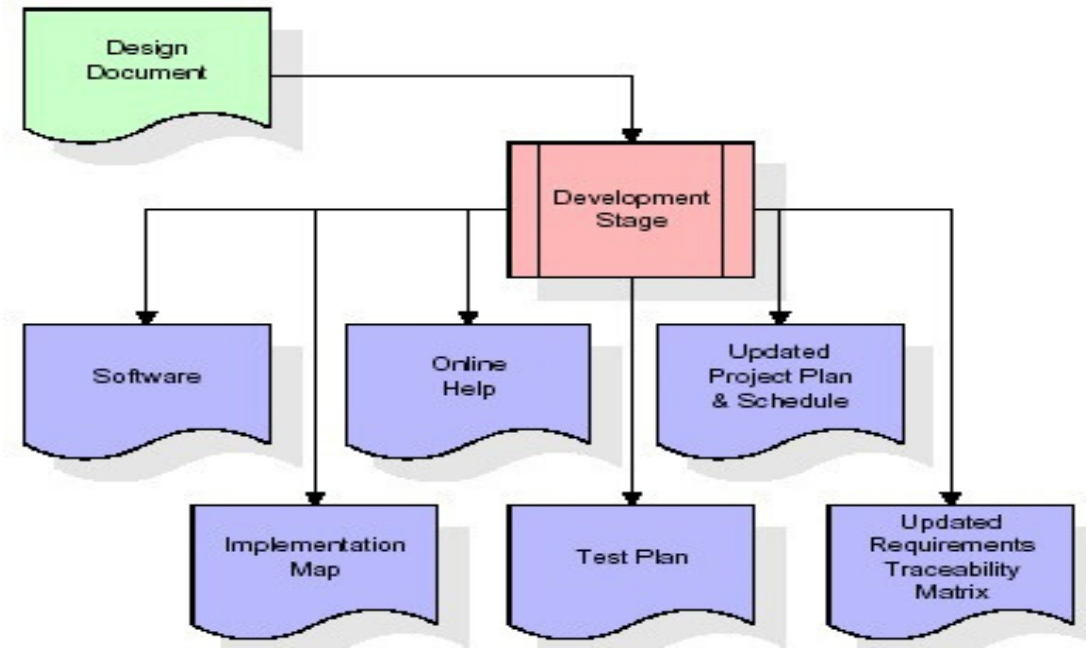
The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity- relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.



When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

Development (Coding) Stage

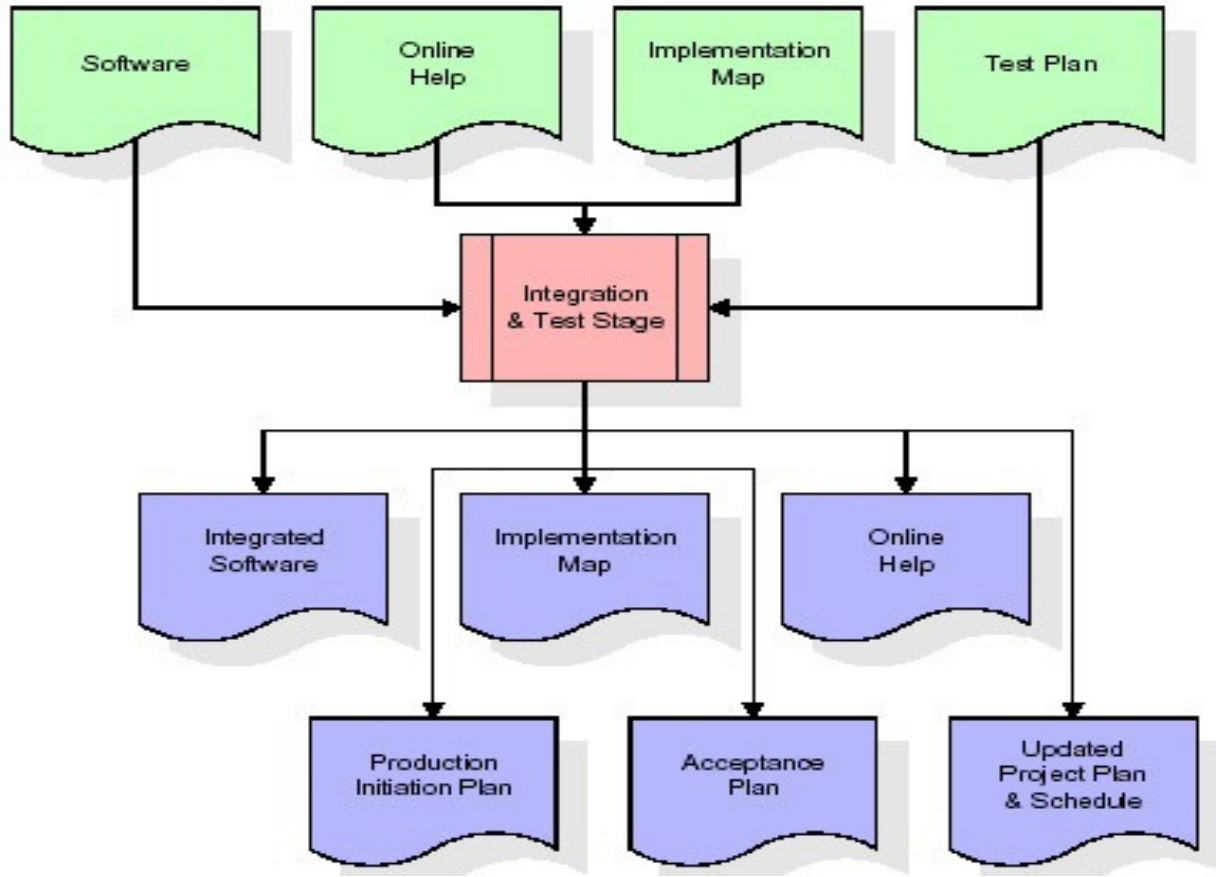
The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, and data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.



The RTM will be updated to show that each developed artifact is linked to a specific design element, and that each developed artifact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.

Integration and Test Stage

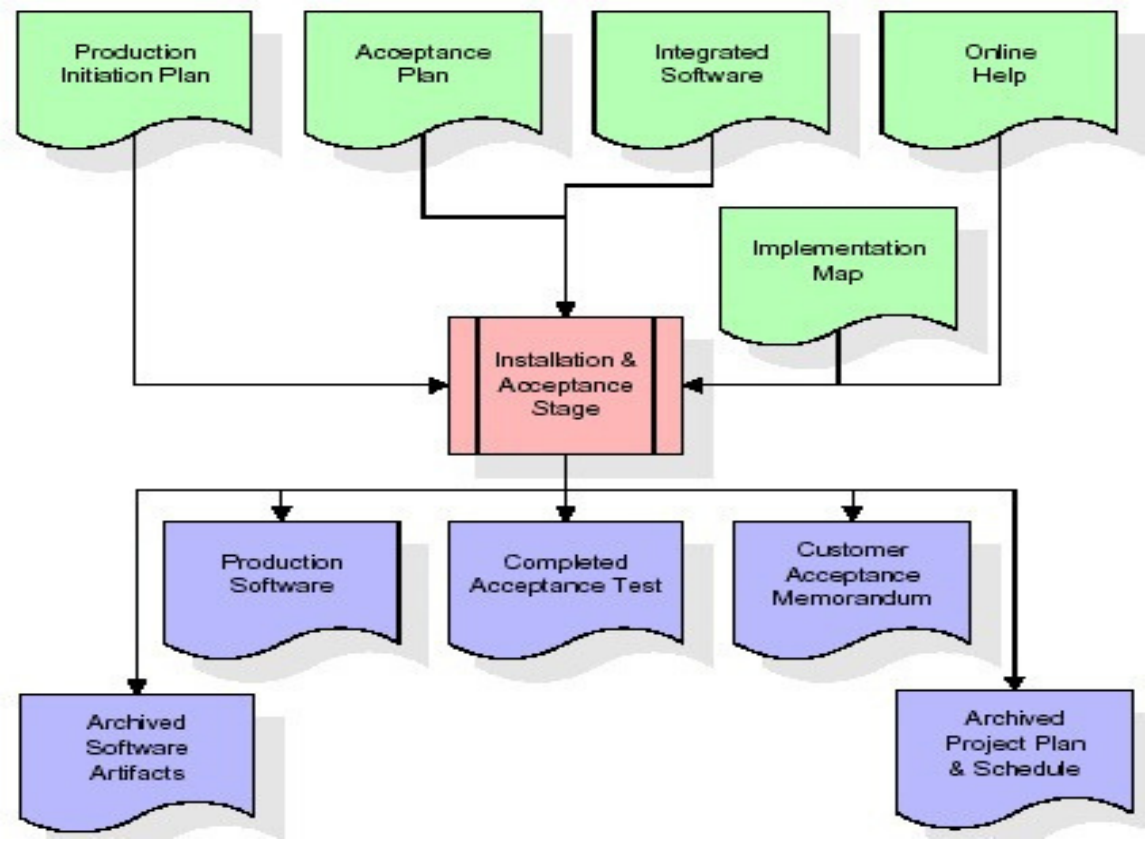
During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.



The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

Installation and Acceptance Test

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loaded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer. After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery



The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the actual labor data into the project schedule and locks the project as a permanent project record. At this point the PDR "locks" the project by archiving all software items, the implementation map, the source code, and the documentation for future reference.

Maintenance

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category. For this life cycle there is no end, it will be continued so on like an umbrella (no ending point to umbrella sticks).

8. 2. Software Testing Techniques

Each Module can be tested using the following two Strategies:

- Black Box Testing
- White Box Testing

Black Box Testing

Black box testing is a software - testing techniques in which functionality of the software under test (SUT) is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software. This type of testing is based entirely on the software requirements and specifications. In Black Box Testing we just focus on inputs and output of the software system without bothering about internal knowledge of the software program.

Steps:

Here are the generic steps followed to carry out any type of Black Box Testing.

- Initially requirements and specifications of the system are examined.
- Tester chooses valid inputs (positive test scenario) to check whether SUT processes them correctly. Also some invalid inputs (negative test scenario) are chosen to verify that the SUT is able to detect them.
- Tester determines expected outputs for all those inputs.
- Software tester constructs test cases with the selected inputs.
- The test cases are executed.
- Software tester compares the actual outputs with the expected outputs.

Types of Black Box Testing:

There are many types of Black Box Testing but following are the prominent ones –

- Functional testing – This black box testing type is related to functional requirements of a system; it is done by software testers.
- Non-functional testing – This type of black box testing is not related to testing of a specific functionality, but non-functional requirements such as performance, scalability, usability.
- Regression testing – Regression testing is done after code fixes , upgrades or any other system maintenance to check the new code has not affected the existing code.

White Box Testing

White Box Testing is the testing of a software solution's internal coding and infrastructure. It focuses primarily on strengthening security, the flow of inputs and outputs through the application, and improving design and usability. White box testing is also known as clear, open, structural, and glass box testing.

It is one of two parts of the “box testing” approach of software testing. Its counter-part, black box testing, involves testing from an external or end-user type perspective. On the other hand, Whitebox testing is based on the inner workings of an application and revolves around internal testing. The term “white box” was used because of the see-through box concept. The clear box or white box name symbolizes the ability to see through the software’s outer shell (or “box”) into its inner workings. Likewise, the “black box” in “black box testing” symbolizes not being able to see the inner workings of the software so that only the end-user experience can be tested.

White box testing involves the testing of the software code for the following:

- Internal security holes
- Broken or poorly structured paths in the coding processes
- The flow of specific inputs through the code
- Expected output

8. 3 Levels of Testing

This examines the logic of the program. For example, the logic for updating various sample data and with the sample files and directories were tested and verified.

Specification Testing

Executing this specification starting what the program should do and how it should performed under various conditions. Test cases for various situation and combination of conditions in all the modules are tested.

Unit testing

In the unit testing we test each module individually and integrate with the overall system. Unit testing focuses verification efforts on the smallest unit of software design in the module. This is also known as module testing. The module of the system is tested separately. This testing is carried out during programming stage itself. In the testing step each module is found to work satisfactorily as regard to expected output from the module. There are some validation checks for fields also. For example, the validation check is done for varying the user input given by the user which validity of the data entered. It is very easy to find error debut the system.

Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised. Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

8. 4. Test Cases and Results

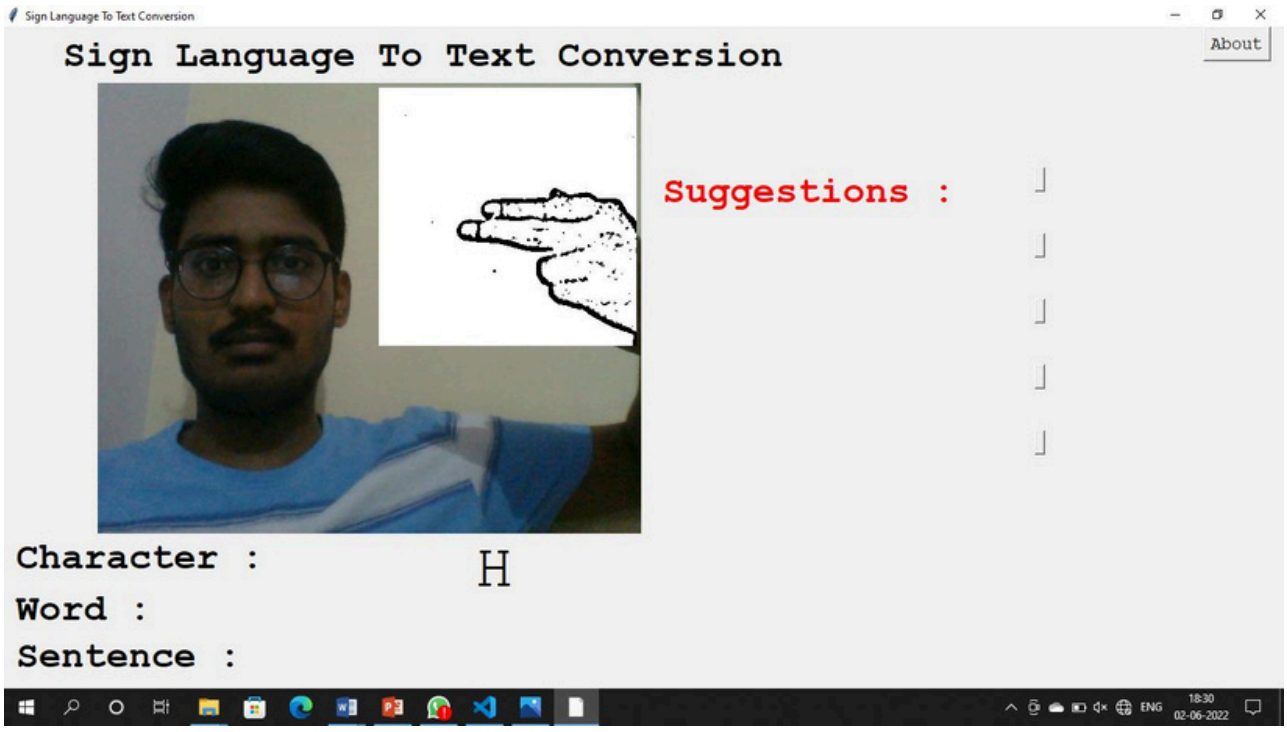
Test Case id	Test Case Name	Test Case Description	Test Steps			Test Execution Result (PASS / FAIL)
			Step	Expected	Actual	
1	Pre-processing the image	Check whether the image is been formatted or not	Input RGB image	Convert to greysc ale. Apply gaussian filter	Convert to greysc ale. Apply gaussian filter	PASS
2	Segmentation	Partitioning a digital image into multiple image segments	Filtered image as input	Grey to binary image. Removing noise. Segmented image.	Grey to binary image. Removing noise. Segmented image.	PASS
3	Feature Extraction	Process of defining a set of features, which are most efficient or meaningful	Segmente d image	GLCM Feature extraction Feature matrix.	GLCM Feature extraction Feature matrix.	PASS
4	Training	Process done to create a model	Training a model with the images	Train ML model with feature matrix. Save trained model	Train ML model with feature matrix. Save trained model	PASS

5	Classification	Task to identify the category of new observations on the basis of training data	Input rgb image	1.Preprocessing 2.Segmentation 3. Feature extraction Pass features to trained ml model	1.Preprocessing 2.Segmentation 3.Featurextraction Pass features to trained ml model	PASS
6	Sign Recognition	Recognizing the sign language and showing the output	Showing signs with the hand in front of camera	A new window is opened and Sign is recognized when live camera is start and display the result as text	A new window is opened and Sign is recognized when live camera is start and display the result as text	PASS
7	Voice Conversion	The text displayed is converted into voice	Showing signs with the hand in front of camera	The output text we get is converted into voice	The output text we get is converted into voice	PASS

CHAPTER 9: OUTPUT SCREENS

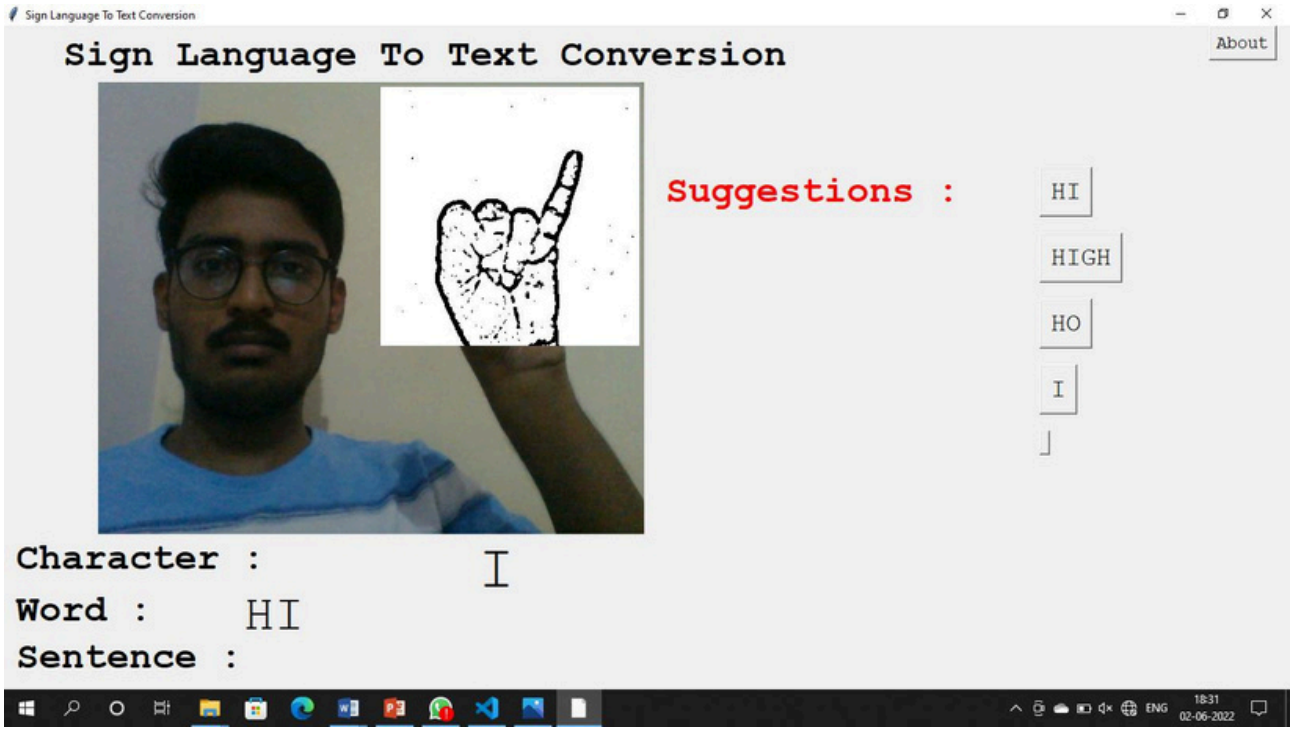
SIGN LANGUAGE TO TEXT AND SPEECH TRANSLATION

1. Character Prediction



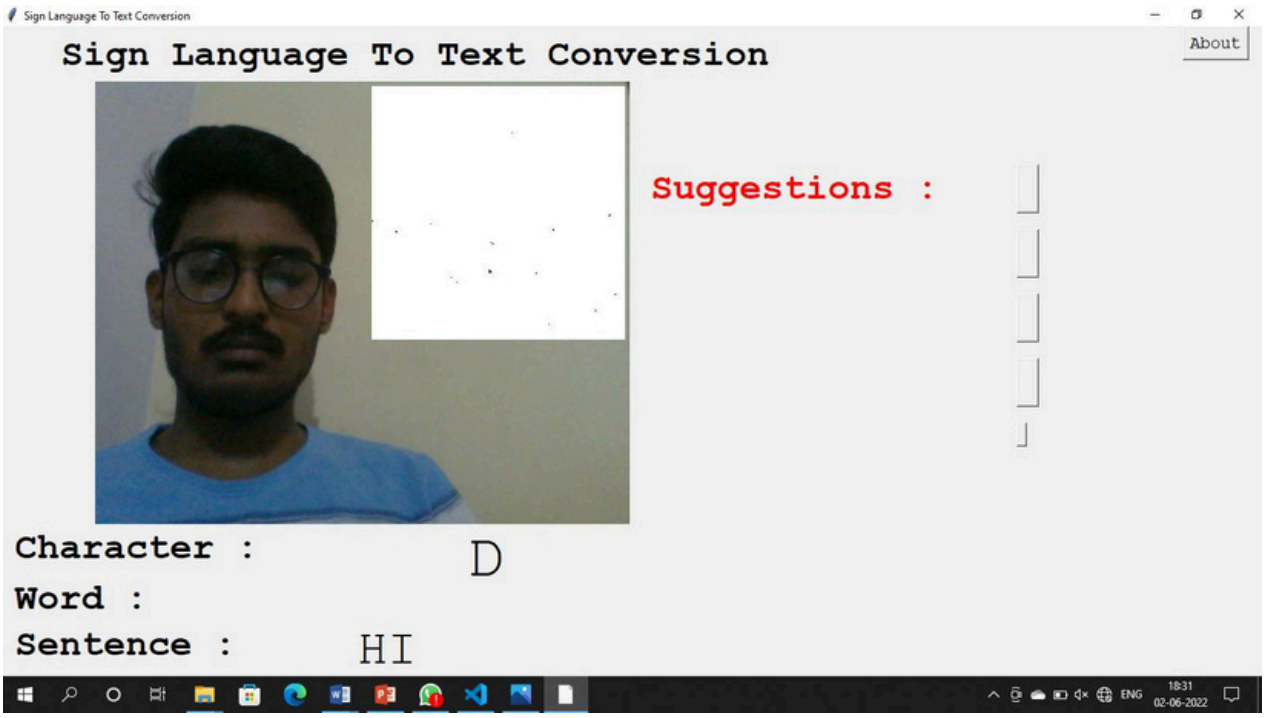
In above output screen we can see that a letter is being predicted when a hand gesture is shown through the camera.

2. Word Prediction



In above output screen we can see that a Word is being predicted when the hand gesture is shown through the camera.

3. Sentence Prediction and Voice Conversion :



In above output screen we can see that a Sentence is being formed.

We can see the suggestions highlights in the right side, when we click on a suggestion word , it will be added in the sentence

Voice : The Sentence displayed on the screen will be heard as a voice.

CHAPTER 10: CONCLUSION

From this Major project we have tried to overshadow some of the major problems faced by the disabled persons in terms of talking. We found out the root cause of why they can't express more freely. The result that we got was the other side of the audience are not able to interpret what these persons are trying to say or what is the message that they want to convey.

Thereby this application serves the person who wants to learn and talk in sign languages. With this application a person will quickly adapt various gestures and their meaning as per ASL standards. They can quickly learn what alphabet is assigned to which gesture. Add- on to this custom gesture facility is also provided along with sentence formation. A user need not be a literate person if they know the action of the gesture, they can quickly form the gesture and appropriate assigned character will be shown onto the screen.

Concerning to the implementation, we have used TensorFlow framework, with keras APL And for the user feasibility complete front-end is designed using PyQT5. Appropriate user-friendly messages are prompted as per the user actions along with what gesture means which character window. Additionally, an export to file module is also provided with TTS(Text-To-Speech) assistance meaning whatever the sentence was formed a user will be able to listen to it and then quickly export along with observing what gesture he/she made during the sentence formation.

CHAPTER 11: FUTURE SCOPE

For future work, there's still so many possibility of improvement, like noise reduction, dataset collection, better feature extraction, or better model to be used.

In future, we are looking at developing a system for Indian sign language words that works in real-time. And, we will make efforts to extend our work towards more words and sentences of Indian sign language.

It can be integrated with various search engines and texting application such as google, WhatsApp. So that even the illiterate people could be able to chat with other persons, or query something from web just with the help of gesture.

This project is working on image currently; further development can lead to detecting the motion of video sequence and assigning it to a meaningful sentence with ITS assistance.

CHAPTER 8: REFERENCES

[1] Shobhit Agarwal, "What are some problems faced by deaf and dumb people while using todays common tech like phones and PCs", 2017 [Online].Available: https://www .quora.com/What-are-some-problems-faced-by-deaf-and-dumb-people- while using-todays-common-tech-like-phones-and-PCs, [Accessed April 06, 2019].

[2] NIDCD, "American sign language", 2017 [Online].Available: https://www .nidcd.nih.gov/health/american-sign-language, [Accessed April 06, 2019].

[3] Suharjito MT, "Sign Language Recognition Application Systems for Deaf-Mute People A Review Based on Input-Process-Output",2017 [Online]. Available:https://www.academia.edu /35314119/Sign_Language_Recognition_Application_Systems_for_Deaf Mute_People_A_Review_Based_on_Input-Process-Output

[4] M. Ibrahim,"Sign Language Translation via Image Processing", [Online]. Available: https://www.kics.edu.pk/project/startup /203

[5] NAD, "American sign language-community and culture frequently asked questions", 2017 [Online]. Available: https://www .nad.org /resources/american-sign language/community-and-cult urefrequently-asked-questions/ [Accessed April 06, 2019].

[6]Sanil Jain and K.V.Sameer Raja,"Indian Sign Language Character Recognition",[Online]. Available:https://cse.iitk.ac.in/users/cs365/2015/_submissions/vinsam/report.Pdf [AccessedApril 06, 2019]