A Project Report

On

App for Deaf and Mute

Submitted in partial fulfilment of the requirement of

University of Mumbai

For the Degree of

Bachelor of Engineering

in

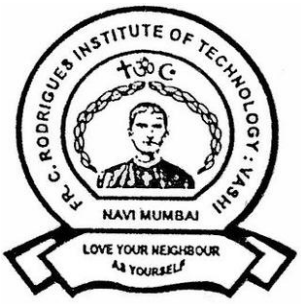
COMPUTER ENGINEERING

Submitted by

Tushar Gupta 1019127
Aditya Jadhav 1019131
Joel Thomas 1019133

Supervised by

Ms. Shagufta Rajguru



Department of Computer Engineering

Fr. Conceicao Rodrigues Institute of Technology
Sector 9A, Vashi, Navi Mumbai - 400703

UNIVERSITY OF MUMBAI

2022-2023

APPROVAL SHEET

This is to certify that the project entitled

"App For Deaf and Mute"

Submitted by

Tushar Gupta 1019127
Aditya Jadhav 1019131
Joel Thomas 1019133

In partial fulfillment of degree of B.E. in Computer Engineering for term work of the project is approved.

Supervisors : Ms. Shagufta Rajguru

Project Coordinator : Dr. Pravin Rahate

Examiners : 1. _____

2. _____

Head of Department : Mrs. Lata Ragha

Date :

Place :

Declaration

We declare that this written submission for B.E. Declaration entitled "**App For Deaf and Mute**" represent our ideas in our own words and where others' ideas or words have been included. We have adequately cited and referenced the original sources. We also declared that we have adhere to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any ideas / data / fact / source in our submission. We understand that any violation of the above will cause for disciplinary action by institute and also evoke penal action from the sources which have thus not been properly cited or from whom paper permission have not been taken when needed.

Project Group Members:

1. Tushar Gupta, 1019127

_____

2. Aditya Jadhav, 1019131

_____

3. Joel Thomas, 1019133

_____

Abstract

Hand gesture is one of the methods used in sign language for non-verbal communication. It is most commonly used by people having hearing or speech problems. Hand gestures play a key role in non verbal communication of our daily life. The primary goal of gesture recognition is to create systems, which can identify specific human gestures and use them, for example, to convey information. More than 360 million of the world population suffers from hearing and speech impairments. It is necessary to create a platform using which deaf and mute people can communicate using non verbal communication. Another dilemma faced is that, images these days are information-rich and in-order to understand these images extensive processing is required. Therefore a medium that can recognize and translate gestures into understandable words by common people is required. The objective of this project is to create a machine learning-based real-time vision application that can recognize and interpret hand gestures into language that is accessible to all users. Our goal is to create a tool that will assist in removing the communication barriers that exist between mute people and others. This project is an implementation for designing a model in which a web camera is used for capturing images of hand gestures which is done by Opencv. Methods like identifying hand motion trajectories for distinct signs and segmenting hands from the background to forecast and string them into sentences that are both semantically correct and meaningful are also used. After capturing images, labeling of images is required.

Keywords—*Computer and information processing, Feature extraction, Gesture recognition, Image processing, Sign Language Recognition, Speech recognition*

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The sign language is an important method of communication for deaf-dumb persons. As sign language is well structured code gesture, each gesture has a meaning assigned to it. It can be used to express complex meanings by combining basic elements. 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 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. This app 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 the last several years there has been an increased interest among the researchers in the field of sign language recognition to introduce means of interaction from human – human to human – computer interaction. Deaf and Dumb people rely on sign language interpreters for communications. However, finding experienced and qualified interpreters for their day to day affairs throughout life period is a very difficult task and also unaffordable.
The common problem faced by many systems is the poor quality of images that usually don't get detected by the system. In the literature, the issue faced by different hand gesture recognition systems has been studied systematically and draws a considerable number of researchers. To reduce the problems of image orientation faced by these systems, we use feature extraction. Feature extraction is a process that aims to reduce the number of features in a data set by creating new features from the existing one and discarding the original one's. Here, we transform the original data to features with strong pattern recognition ability, where the original data can be regarded as features with weak recognition ability.

1.2 Motivation

Communication of humans with computers has always been a challenge. From the early mediums such as perforated cards, people have spent more than half of the past century experimenting with various ways to interact with computers – in pursuit of more efficient and intuitive interfaces. Keyboard and mouse slowly became the de-facto industry standard for

user input. Much progress has been made, but we believe that people can communicate even more effectively when given a richer lexicon of gestures. Hand gestures can add another dimension to communication. Just like verbal communication involves enriching voice with subtle facial expressions and gestures, so does the machine interface benefit from an additional layer of interactivity provided by gestures. Machine learning is already used to recognize facial expressions. But gesture recognition has always been more challenging and only recently started to gain popularity. Powered by machine learning, gesture recognition is now not only possible, but also more accurate than ever.
This project is an implementation for designing a model in which a web camera is used for capturing images of hand gestures which is done by opencv. Methods like identifying hand motion trajectories for distinct signs and segmenting hands from the background to forecast and string them into sentences that are both semantically correct and meaningful are also used. After capturing images, labeling of images is required. Thus, an effective path of communication can be developed between deaf and normal audiences.
Three steps must be completed in real time to solve our problem, obtaining footage of the user signing is step one (input). The second step then is to classify each frame in the video to a sign. Finally, we reconstruct and display the most likely Sign from classification scores (output).

1.3 Aim and Objective

Hand gesture is one of the methods used in sign language for non-verbal communication. It is most commonly used by people having hearing or speech problems. Hand gestures play a key role in non verbal communication of our daily life. The primary goal of gesture recognition is to create systems, which can identify specific human gestures and use them, for example, to convey information. Therefore a medium that can recognize and translate gestures into understandable words by common people is required.
The goal of this project is to create a machine learning-based real-time vision application that can recognize and interpret hand gestures into language that is accessible to all users, thereby removing the communication barriers that exist between mute people and others.

1.4 Report Outline

In this paper we have discussed various approach used for Hand Gesture Recognition. We have also mentioned the approach utilized for building a recognition engine for detecting correct hand gestures in the later part of the part. Also this report details our approach to create a machine learning-based real-time vision application that can recognize and interpret hand gestures into language that is accessible to all users thereby, removing the communication barrier that exist between mute people and others.

Chapter 2

Study of the System

2.1 Related Works

Researchers are trying their best to solve the problem of this topic. There are so many research had been done. But still need to be improved to be a decent to solve this topic's problem. We are here to discuss about other's research.

In the proposed model [1], the authors proposed a Tiny Hand Gesture Recognition without Localization via a Deep Convolutional Network using Deep Convolutional Network to provide a system that is able to directly recognize hand gestures from the whole image without using any image region selection framework. The proposed deep CNN design is composed by 9 convolutional layers, 4 pooling layers, 3 fully connected layers, interlaced with ReLU (Rectified Linear Unit) and dropout layers. This system can classify seven sorts of hand gestures in a user-independent manner and on real time, achieving an accuracy of 85.3% in the dataset with complex backgrounds. The paper proposes a system for hand gesture recognition that uses a deep convolutional neural network (CNN) for feature extraction and a support vector machine (SVM) for classification. Unlike other hand gesture recognition systems, this system does not require explicit localization of the hand. Instead, it works directly on input images of hand gestures. This system is designed to work on low-power devices, making it suitable for use in wearable technology. The authors evaluate the performance of their system on a public dataset and compare it with other existing methods. They show that their system outperforms existing methods in terms of recognition accuracy. Overall, the paper demonstrates the feasibility and effectiveness of using deep learning for hand gesture recognition without the need for explicit localization, which has practical implications for real-world applications of the technology.

The model in paper [2] provided the user with a natural interaction and a good experience when interacting with a computer in contexts of application such as the interaction with maps, allowing intuitive movements of the earth surface. Architecture consists of the definition of synthetic motion patterns which will be compared with the hand motion estimations computed from the real data set videos. The model is able to work in real-time allowing the interaction between a user and a virtual environment or computer menu. The proposed system uses a combination of feature extraction and classification techniques to recognize hand gestures from ToF video streams in real-time. The system detects the hand region using a background subtraction technique and extracts motion features from the hand region using a spatiotemporal descriptor. The extracted features

are then classified using a support vector machine (SVM) classifier. The authors evaluate the performance of their system on a public dataset and show that it achieves high recognition accuracy and fast processing times, making it suitable for real-time applications.Overall, the paper demonstrates the effectiveness of using ToF video and motion-based features for hand gesture recognition in real-time, which has practical implications for a wide range of applications, including human-computer interaction, virtual reality, and robotics.

Another model proposed in paper [3] is based on Appearance based models. Here the authors developed a User adaptive hand gesture recognition system with interactive training using Appearance Based Methods. They implemented the recognition method in a camera– projector configuration. The goal of this model is to demonstrate an effective hand gesture-based human–computer interface extended with an interactive training method for augmented reality (AR) applications. The minimal recognition rate of the gestures is about 65%. The paper presents a novel approach to hand gesture recognition that takes into account the user's individual characteristics, such as hand shape and size, to improve recognition accuracy. The system uses interactive training to adapt to the user's hand gestures and provides real-time feedback to guide the user during the training process. The authors describe the system architecture, which consists of a pre-processing module, feature extraction module, classification module, and user adaptation module. The pre-processing module removes noise from the input data, and the feature extraction module extracts relevant features from the hand gesture images. The classification module uses a support vector machine (SVM) to classify the hand gestures, and the user adaptation module updates the classification model based on the user's feedback during training. The authors evaluate the performance of the system using a dataset of hand gesture images and report an overall recognition accuracy of 91.8%. They also conduct a user study to demonstrate the effectiveness of the interactive training approach, showing that users can improve their gesture recognition accuracy with the system over time. Overall, the paper presents an innovative approach to hand gesture recognition that takes into account the user's individual characteristics and provides interactive training to improve recognition accuracy. The system has potential applications in various fields, including human-computer interaction, virtual reality, and robotics.

Models in paper [4],[5] and [6] proposed by the authors of the respective papers are all based on Convolutional Neural Network. All these models have achieved quite high accuracy in terms of recognizing the right hand

gestures. While model [4] has an average success rate over all signs to be classified as 79.3%, model [6] can detect in a semi-supervised case, the eight types of gesture recognition with an accuracy of 98.52%. While Convolutional Neural Networks may be the most effective algorithm in terms of accuracy, it also does have drawbacks. As specified by the authors of paper [7], who proposed an American Sign Language Recognition system using Deep Learning and Computer Vision, faced a problem with facial features and skin tones. While testing with different skin tones, the model dropped accuracy.

The authors of paper [8] proposed a Hand Gesture Based Mobile Control Application which is based on Acceleration and Surface Electromyographical (SEMG) signals for gesture identification. The goal of the model is to recognize both small scale and large scale gestures. On the basis of the results shown by this model, recognition of hand gesture through SEMG sensor and Accelerometer is the best technique. The paper provides a review of the various hand gesture recognition techniques used in mobile control applications. The authors highlight the advantages of using hand gestures for controlling mobile devices, such as ease of use and the ability to operate the device without touching it. The paper also discusses the challenges faced in implementing hand gesture recognition, such as the need for robust algorithms that can handle variations in lighting and hand orientation. The authors provide an overview of the different types of sensors and cameras that can be used for hand gesture recognition, including RGB cameras, depth sensors, and inertial sensors. Overall, the paper provides a useful overview of the state of the art in hand gesture recognition for mobile control applications. It can be a valuable resource for researchers and practitioners interested in developing new mobile control applications based on hand gesture recognition.

The model proposed by the authors of paper [9] is an an American Sign Language Character Recognition with Capsule Networks. In this study, recognition of sign language characters via a system, which was trained using images of letters in American sign language,is aimed. The paper discusses the use of capsule networks for recognizing American Sign Language (ASL) characters. Capsule networks are a type of neural network architecture that was introduced as an improvement over traditional convolutional neural networks (CNNs). Capsule networks are designed to better handle hierarchical relationships between features in an image, which is important for recognizing complex objects such as ASL characters. The authors used a dataset of ASL characters to train and test their capsule network model. They compared the performance of their model to a traditional CNN model

and found that the capsule network achieved higher accuracy in recognizing ASL characters. In LeNet-5 model, input layer accepts single-channel grayscale image with size of 32x32 pixels.CapsNet consists of three layers. In this study, input images resized to 32x32 pixels to be compatible with LeNet. As a result of the experimental study, LeNet reached 82% accuracy on the test set, while the accuracy of CapsNet reached 88%. The paper concludes that capsule networks can be a promising approach for recognizing ASL characters and potentially other types of sign languages.

Table 2.1: Summary of Papers

Sr. No.	Name of the paper	Algorithm/Model used
1.	Tiny Hand Gesture Recognition without Localization	Deep Convolutional Network
2.	Real-time Motion-based Hand Gestures Recognition from Time-of-Flight Video	Convolution Neural Network
3.	User-adaptive hand gesture recognition system with interactive training	Appearance Based Methods
4.	Sign Language Recognition Using Modified Convolution Neural Network Model	Convolution Neural Network
5.	Czech Sign Language Single Hand Alphabet Letters Classification	Convolution Neural Network
6.	American Sign Language Recognition using Deep Learning and Computer Vision	Convolution Neural Network,Recurrent Neural Network
7.	Hand Gesture Based Mobile Control Application	Acceleration and Surface Electromyographical (SEMG) signals for gesture identification
8.	American Sign Language Character Recognition with Capsule Networks	LeNet-5 model,CapsNet model

2.2 Literature Review

2.2.1 Convolutional Neural Network

In deep learning, a convolutional neural network (CNN/ConvNet) is a class of deep neural networks, most commonly applied to analyze visual imagery. Most deep learning methods usually use a neural network architecture, which is the reason why deep learning models are often known as deep neural networks. The term “deep” usually refers to the number of hidden layers in the neural network. Our neural network contains only 2-3 hidden layers, while deep networks can have as many as 200. There is no need of

manually extracting the features because deep learning models are already trained using large sets of labeled data and neural network architectures which learn features from the given data. One of the most used deep neural network is convolutional neural network (CNN). A ConvNet convolves learned features with given input data and uses convolutional layers making a well suited architecture for extracting objects directly from the images, so there is no need of identifying the features used to classify images. This way of extraction makes deep learning models to give least inaccuracy in larger context to classify objects in computer vision. One of the most used deep neural network is convolutional neural network (CNN). A ConvNet convolves learned features with given input data and uses convolutional layers making a well suited architecture for processing image datasets. ConvNet extracts features directly from the images, so there is no need of identifying the features used to classify images. This way of extraction makes deep learning models to give least inaccuracy in larger.

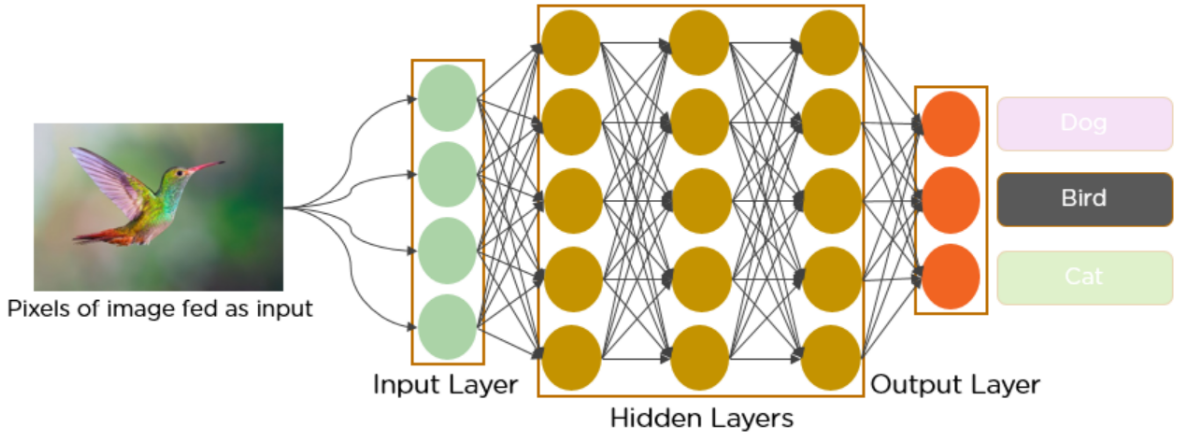


Figure 2.1: Convolution Neural Network

2.2.2 TensorFlow

TensorFlow is an open-source library developed by Google primarily for deep learning applications. It also supports traditional machine learning. TensorFlow was originally developed for large numerical computations without keeping deep learning in mind. However, it proved to be very useful for deep learning development as well, and therefore Google open-sourced it.
TensorFlow accepts data in the form of multi-dimensional arrays of higher dimensions called tensors. Multi-dimensional arrays are very handy in handling large amounts of data. TensorFlow works on the basis of data

flow graphs that have nodes and edges. As the execution mechanism is in the form of graphs, it is much easier to execute TensorFlow code in a distributed manner across a cluster of computers while using GPUs.
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[10].

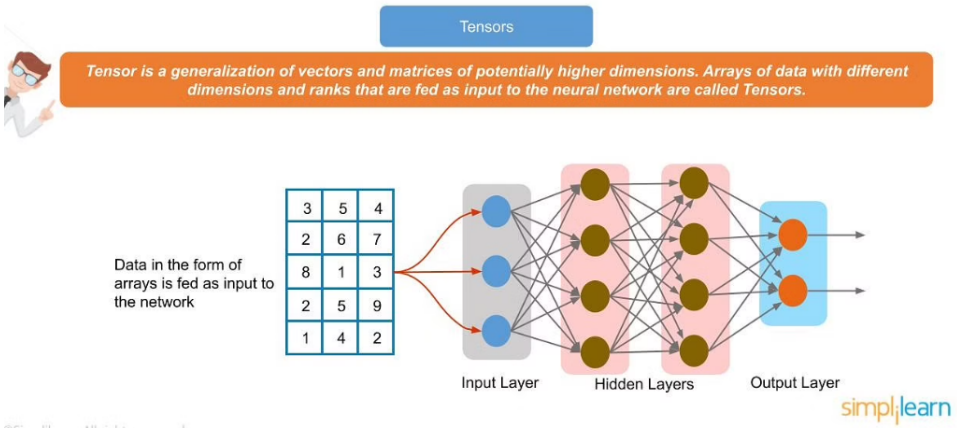


Figure 2.2: Working of TensorFlow

2.2.3 Keras

Keras runs on top of open source machine libraries like TensorFlow, Theano or Cognitive Toolkit (CNTK). Theano is a python library used for fast numerical computation tasks. TensorFlow is the most famous symbolic math library used for creating neural networks and deep learning models. TensorFlow is very flexible and the primary benefit is distributed computing. CNTK is deep learning framework developed by Microsoft. It uses libraries such as Python, C, C++ or standalone machine learning toolkits. Theano and TensorFlow are very powerful libraries but difficult to understand for creating neural networks. Keras is based on minimal structure that provides a clean and easy way to create deep learning models based on TensorFlow or Theano. Keras is designed to quickly define deep learning models. Well, Keras is an optimal choice for deep learning applications.
Keras leverages various optimization techniques to make high level neural network API easier and more performant. It supports the following features:

- Consistent, simple and extensible API.

- Minimal structure - easy to achieve the result without any frills.
- It supports multiple platforms and backends.
- It is user friendly framework which runs on both CPU and GPU.
- Highly scalability of computation.

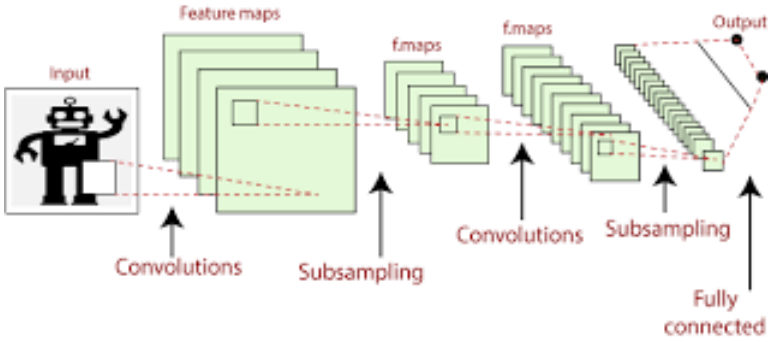


Figure 2.3: Working of Keras

Need for Keras:

- Keras is an API that was made to be easy to learn for people. Keras was made to be simple. It offers consistent simple APIs, reduces the actions required to implement common code, and explains user error clearly.
- Prototyping time in Keras is less. This means that your ideas can be implemented and deployed in a shorter time. Keras also provides a variety of deployment options depending on user needs.
- Languages with a high level of abstraction and inbuilt features are slow and building custom features in then can be hard. But Keras runs on top of TensorFlow and is relatively fast. Keras is also deeply integrated with TensorFlow, so you can create customized workflows with ease.
- The research community for Keras is vast and highly developed. The documentation and help available are far more extensive than other deep learning frameworks.
- Keras is used commercially by many companies like Netflix, Uber, Square, Yelp, etc which have deployed products in the public domain which are built using Keras.

Table 2.2: Tensorflow vs Keras

Sr.no.	Tensorflow	Keras
1.	Tensorhigh-performanceFlow is written in C++, CUDA, Python.	Keras is written in Python.
2.	TensorFlow is used for large datasets and high performance models.	Keras is usually used for small datasets.
3.	TensorFlow is a framework that offers both high and low-level APIs.	Keras is a framework that offers high-Level API.
4.	TensorFlow is used for high-performance models.	Keras is used for low-performance models.
5.	In TensorFlow performing debugging leads to complexities.	In Keras framework, there is only minimal requirement for debugging the simple networks.
6.	TensorFlow has a complex architecture and not easy to use.	Keras has a simple architecture and easy to use.
7.	TensorFlow was developed by the Google Brain team.	Keras was developed by François Chollet while he was working on the part of the research effort of project ONEIROS.

2.2.3.1 Tensorflow vs Keras

Both Tensorflow and Keras are famous machine learning modules used in the field of data science.

- TensorFlow is an open-source platform for machine learning and a symbolic math library that is used for machine learning applications.
- Keras is an Open Source Neural Network library that runs on top of Theano or Tensorflow. It is a useful library to construct any deep learning algorithm of whatever choice we want.

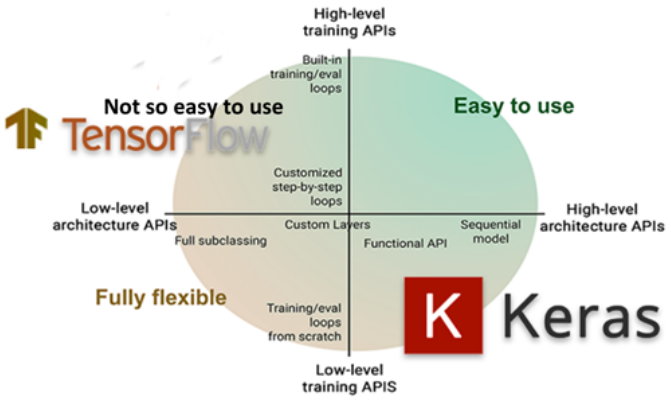


Figure 2.4: Tensorflow vs Keras

2.2.4 OpenCV

OpenCV is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. When it is integrated with various libraries, such as Numpy which is a highly optimized library for numerical operations, then the number of weapons increases in your Arsenal i.e whatever operations one can do in Numpy can be combined with OpenCV. OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.
Following are the main library modules of the OpenCV library:

- Core Functionality: This module covers the basic data structures such as Scalar, Point, Range, etc., that are used to build OpenCV applications. In addition to these, it also includes the multidimensional array Mat, which is used to store the images. In the Java library of OpenCV, this module is included as a package with the name org.opencv.core.

- Image Processing: This module covers various image processing operations such as image filtering, geometrical image transformations, color space conversion, histograms, etc. In the Java library of OpenCV, this module is included as a package with the name org.opencv.imgproc.

- Video: This module covers the video analysis concepts such as motion estimation, background subtraction, and object tracking. In the Java library of OpenCV, this module is included as a package with the name org.opencv.video.

- Video I/O: This module explains the video capturing and video codecs using OpenCV library. In the Java library of OpenCV, this module is included as a package with the name org.opencv.videoio.

- features2d: This module includes the concepts of feature detection and description. In the Java library of OpenCV, this module is included as a package with the name org.opencv.features2d.

- Objdetect This module includes the detection of objects and instances of the predefined classes such as faces, eyes, mugs, people, cars, etc. In the Java library of OpenCV, this module is included as a package with the name org.opencv.objdetect.

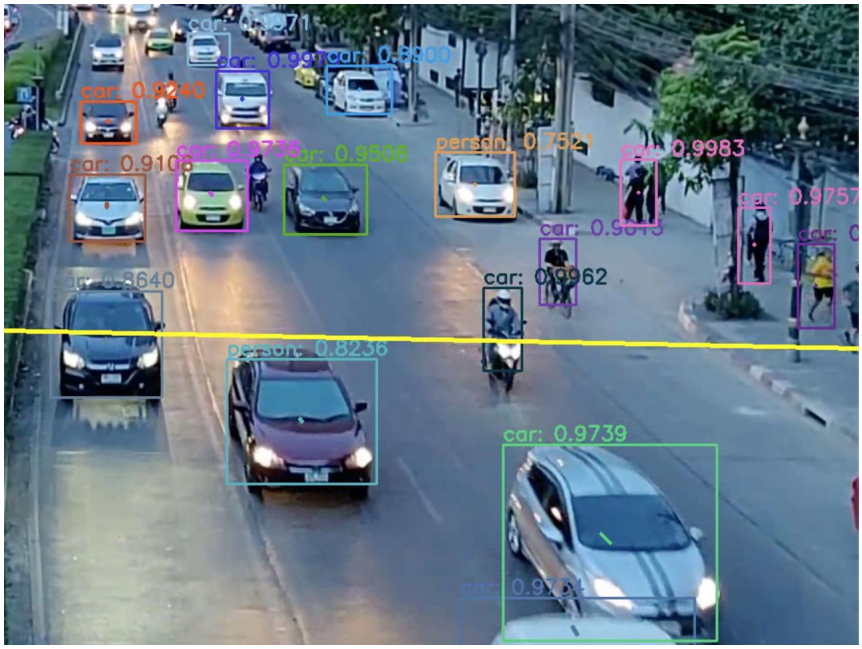


Figure 2.5: OpenCV

The features of OpenCV include:

- Read and write images.
- Capture and save videos.
- Process images (filter, transform).
- Perform feature detection.
- Detect specific objects such as faces, eyes, cars, in the videos or images.
- Analyze the video, i.e., estimate the motion in it, subtract the background, and track objects in it.

2.2.5 Mediapipe

MediaPipe is a Framework for building machine learning pipelines for processing time-series data like video, audio, etc. This cross-platform Framework works on Desktop/Server, Android, iOS, and embedded devices like Raspberry Pi and Jetson Nano. Since 2012, Google has used it internally in several products and services. It was initially developed for real-time analysis of video and audio on YouTube. Gradually it got integrated into many more products.
The following are some uses:

- Perception system in NestCam.

- Object detection by Google Lens.
- Augmented Reality Ads.
- Google Photos.
- Google Home.
- Gmail.
- Cloud Visison API.

MediaPipe powers revolutionary products and services we use daily. Unlike power-hungry machine learning Frameworks, MediaPipe requires minimal resources. It is so tiny and efficient that even embedded IoT devices can run it. MediaPipe Toolkit comprises the Framework and the Solutions. The Framework is written in C++, Java, and Obj-C. The MediaPipe perception pipeline is called a Graph. Let us take the example of the first solution, Hands. We feed a stream of images as input which comes out with hand landmarks rendered on the images.
MediaPipe supports multimodal graphs. To speed up the processing, different calculators run in separate threads. For performance optimization, many built-in calculators come with options for GPU acceleration. Working with time series data must be synchronized properly; otherwise, the system will break. The graph ensures that flow is handled correctly according to the timestamps of packets. The Framework handles synchronization, context sharing, and inter-operations with CPU calculators.

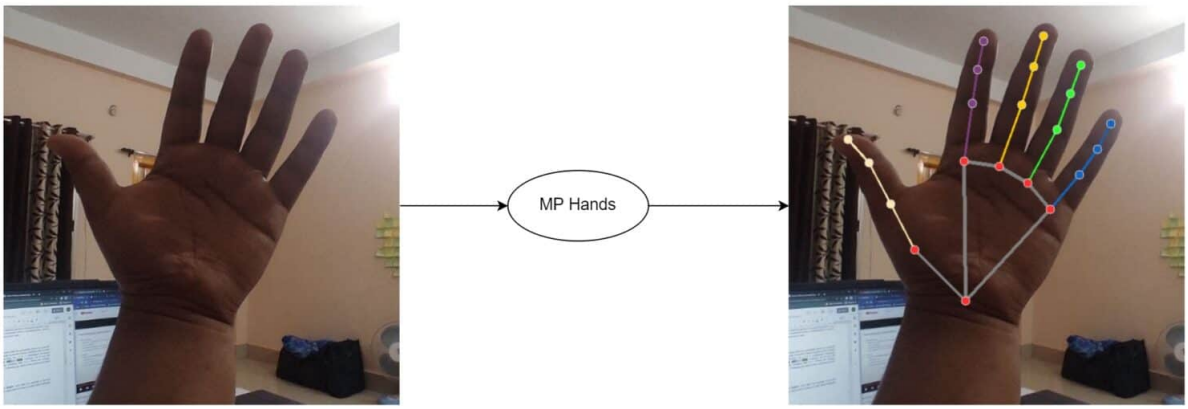


Figure 2.6: Mediapipe Hand-solution graph

2.2.6 Feature Extraction

Feature extraction is a technique used to reduce a large input data set into relevant features. This is done with dimensionality reduction to transform

large input data into smaller, meaningful groups for processing. It refers to the process of transforming raw data into numerical features that can be processed while preserving the information in the original data set. It yields better results than applying machine learning directly to the raw data.

Feature extraction can be accomplished manually or automatically:

- Manual feature extraction requires identifying and describing the features that are relevant for a given problem and implementing a way to extract those features. In many situations, having a good understanding of the background or domain can help make informed decisions as to which features could be useful. Over decades of research, engineers and scientists have developed feature extraction methods for images, signals, and text. An example of a simple feature is the mean of a window in a signal.
- Automated feature extraction uses specialized algorithms or deep networks to extract features automatically from signals or images without the need for human intervention. This technique can be very useful when you want to move quickly from raw data to developing machine learning algorithms. Wavelet scattering is an example of automated feature extraction.

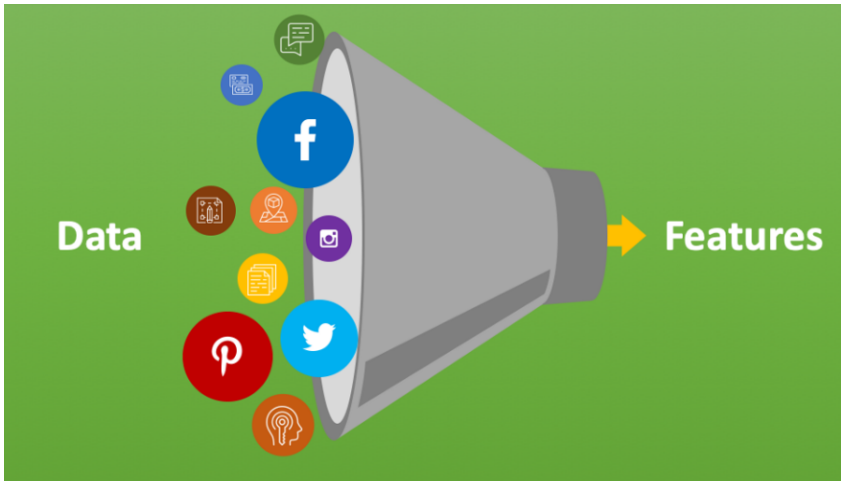


Figure 2.7: Feature Extraction

With the ascent of deep learning, feature extraction has been largely replaced by the first layers of deep networks – but mostly for image data. For signal and time-series applications, feature extraction remains the first challenge that requires significant expertise before one can build effective

predictive models.
Feature extraction can prove helpful when training a machine learning model. It leads to:

- A boost in training speed
- An improvement in model accuracy
- A reduction in risk of overfitting
- A rise in model explainability
- Better data visualization

Feature Extraction for Image Data:
Feature extraction for image data represents the interesting parts of an image as a compact feature vector. In the past, this was accomplished with specialized feature detection, feature extraction, and feature matching algorithms. Today, deep learning is prevalent in image and video analysis, and has become known for its ability to take raw image data as input, skipping the feature extraction step. Regardless of which approach you take, computer vision applications such as image registration, object detection and classification, and content-based image retrieval, all require effective representation of image features – either implicitly by the first layers of a deep network, or explicitly applying some of the longstanding image feature extraction techniques.

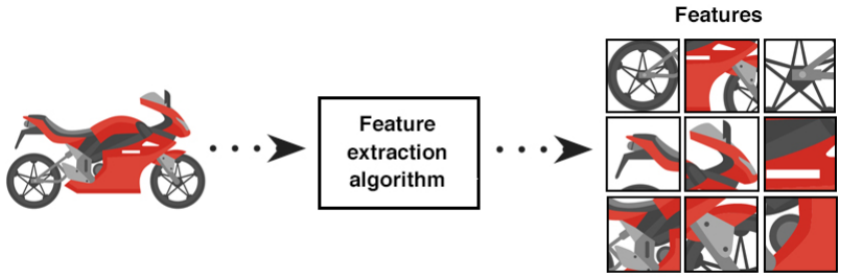


Figure 2.8: Example of Feature Extraction

Feature extraction techniques:
The following is a list of some common feature extraction techniques used by the Machine Learning community:

- Principle Components Analysis (PCA)
- Independent Component Analysis (ICA)
- Linear Discriminant Analysis (LDA)

- Locally Linear Embedding (LLE)
- t-distributed Stochastic Neighbor Embedding (t-SNE)

2.2.7 Comparison of Algorithms:

The following table shows the comparison of various Machine Learning algorithms [11].

Table 2.3: Comparison of Algorithms

Sr.No.	Algorithm/Network	Advantages	Disadvantages
1.	Linear Regression	Fast to train and Forecast. Good for small classification.	Not very Accurate.Not used for non- linear data. Overfitting problem.
2.	Nearest Neighbours	Adaptable to problem. Accurate. Use saptial trees to improve space issue.	Memory intensive.Higher cost. Inaccurate results on choosing wrong distance measure.
3.	Random Forest	High Accuracy, flexible and fit variety of data. Fast execution. Used for regression and classification.	Slow at Training,Overfitting. Not for small samples.
4.	CNN	Good performance.Learning of model is fast and accuracy is high.	Needs lot of data for classification.
5.	RNN	Learns sequential events, can model time dependencies.	Issue of Gradient vanishing and needs large dataset.
6.	SVM	Durable to noise,Can model non-linear data. Overfitting unlikely to occur.	Slow training. Difficult to determine optical parameters, difficult to understand structure.

Chapter 3

Proposed System

3.1 Problem Statement

Ideally, gesture recognition should be based on a photo of a still hand showing only a single gesture against a clear background in well-lit conditions. But real-life conditions are hardly ever like that. We don't always get the comfort to use solid, clear backgrounds when presenting gestures. A gesture recognition model should bring great results no matter the background: it should work whether you're in the car, at home, or walking down the street. It should be able to recognize not just static images but also dynamic hand gestures. Moreover, gestures could consist of several movements, so we need to provide some context and recognize patterns like moving fingers clockwise and showing a thumb could be used to mark some limited number of files or some limited area. The gesture detection system must be designed to eliminate the lag between performing a gesture and its classification. Adoption of hand gestures can only be encouraged by showing how consistent and instantaneous it can be. There is really no other reason to start using gestures if they don't make your interaction faster and more convenient. These issues make it exceedingly difficult to build a solid method of detecting gestures in difficult conditions. This research intends to offer a solution to this problem by creating a method that can accurately recognize gestures taking into account the various complexities associated with it.

3.2 Scope

The scope of this project offers method to recognize hand gestures in real-time. The trained model only works for seven different categories of gestures. Adding more categories will take a larger database for corresponding gestures.

3.3 Proposed System

The proposed system for detecting Sign Language will be based on Keras model. Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. Since a practical model should detect objects in real time we have used OpenCV. OpenCV is a open source library which aids in real time detection. Teachable machine is used to train the model.Teachable Machine is a web tool that makes it fast and easy to create machine learn-

ing models for your projects, no coding required. Train a computer to recognize your images, sounds, poses, then export your model for your sites, apps, and more.

Approach Utilized:

Teachable Machine:
Teachable Machine is a tool created by Google, stoj and Use All Five, with the help of TensorFlowJS, to make machine learning and artificial intelligence available to everyone. Usually machine learning algorithms are developed using different frameworks specialized for machine learning, like Pytorch or TensorFlow. For somebody who is not really well versed on machine learning, digging inside the technology needed and understanding all the steps to generate a result from a machine learning algorithm can take a long time. Not only mental energy is needed but also money, because executing the training steps on the cloud can be expensive.

With Teachable Machine, people are able to create simple and effective algorithms in the browser that can:

- Identify patterns on images
- Identify patterns on audio
- Identify poses or gestures

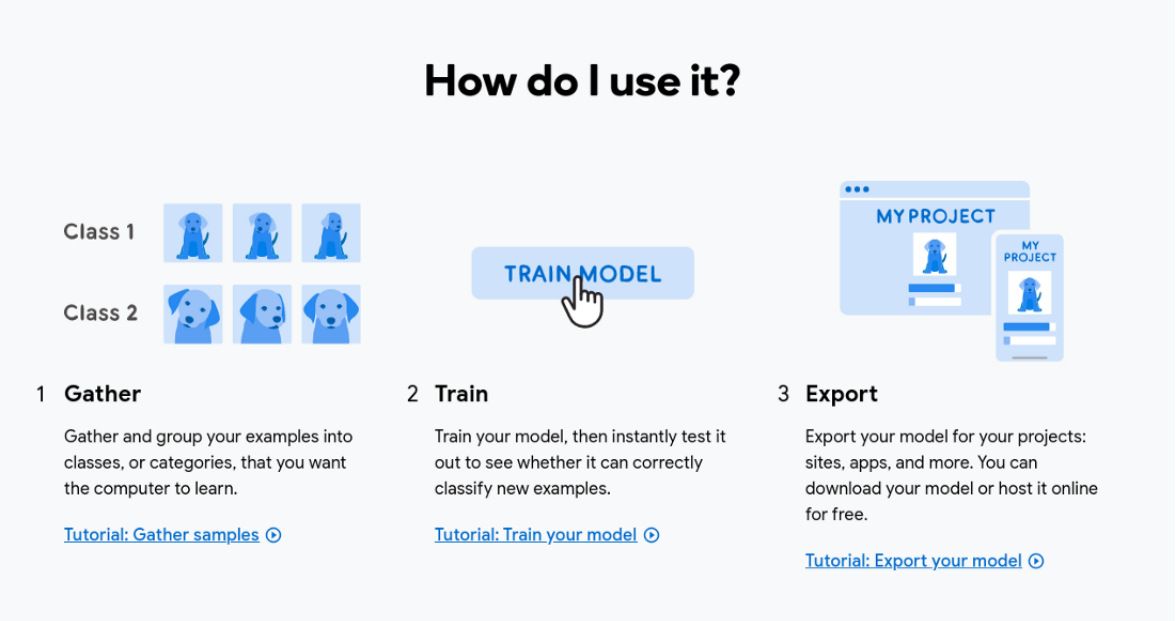


Figure 3.1: Working of Teachable Machine

Chapter 4

Design of the System

4.1 Design

The below figures shows the design of of our proposed model.

4.1.1 Use Case Diagram of system

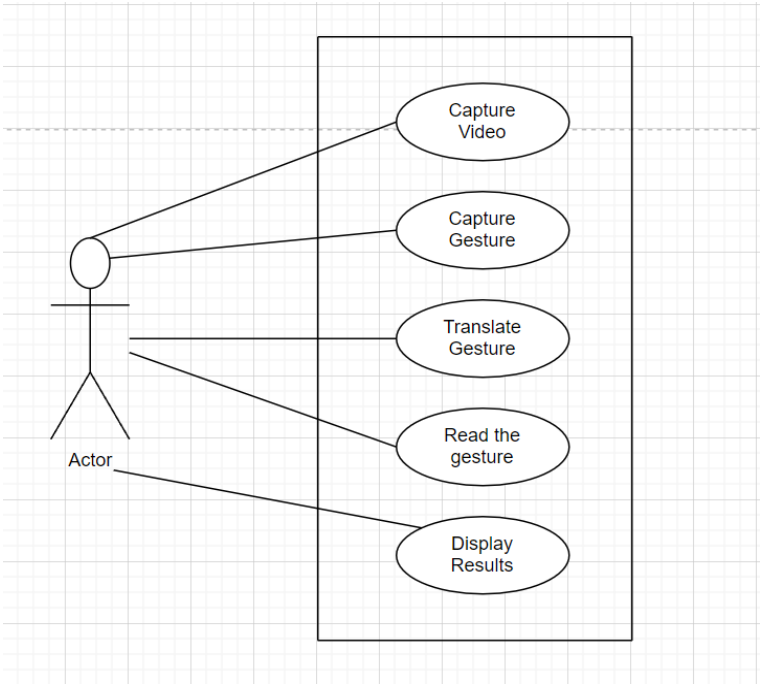


Figure 4.1: Use Case Diagram

A use case diagram for a hand gesture recognition system would typically include several actors and use cases. Here's a summary of some of the key elements that might be included:

1. Actors:

- User: the person who interacts with the hand gesture recognition system to perform actions using hand gestures.

2. Use Cases:

- Capture Gesture: the system captures the user's hand gesture using a camera or sensor.
- Recognize gestures: the system analyzes the captured gesture and recognizes the corresponding action or command.
- Translate Gestures: the system translates the captured gestures into sentence.
- Display Results: the system displays the result in text or speech form.

4.1.2 Activity Diagram for Hand Gesture model

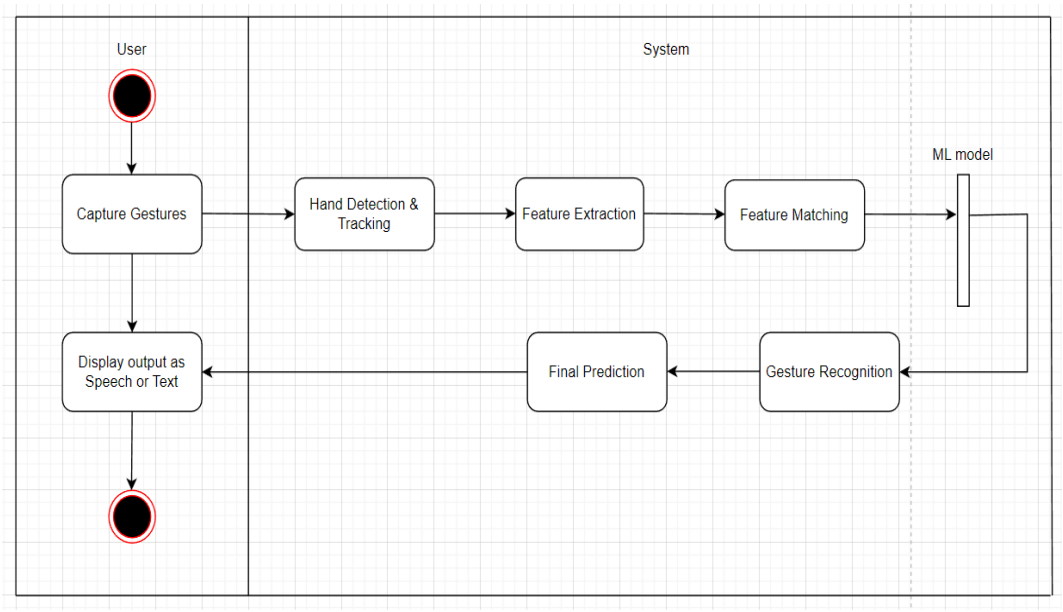


Figure 4.2: Activity Diagram of SL model

The activity diagram for hand gesture recognition system involves capturing video of hand gestures, pre-processing the video, extracting features, recognizing the gesture, making a decision based on the recognized gesture, and concluding the process. It outlines the steps involved in identifying specific gestures and triggering corresponding actions.

4.1.3 Sequence Diagram of SL model

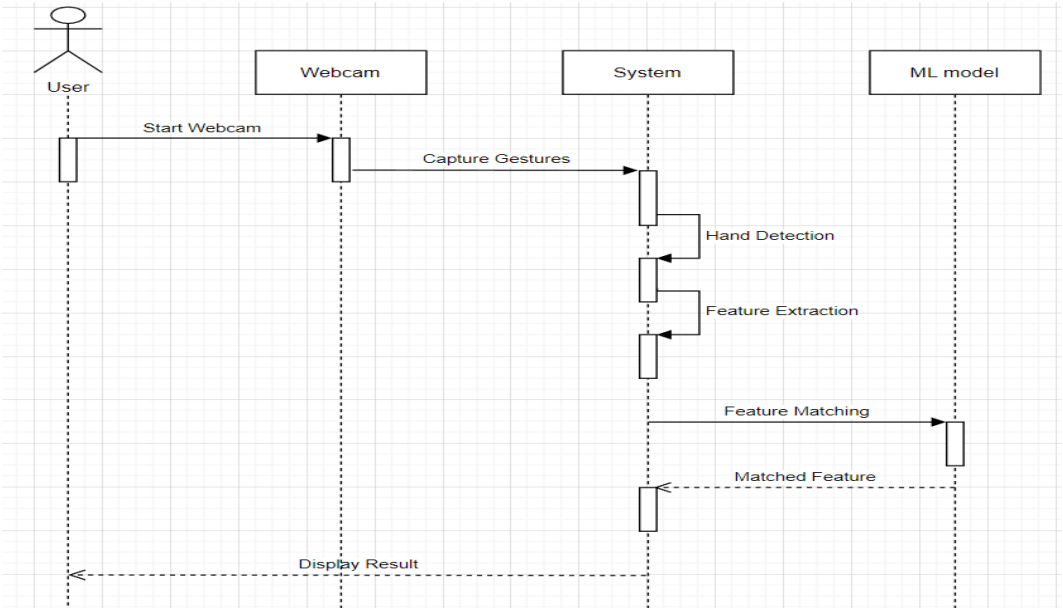


Figure 4.3: Sequence Diagram of SL model

The sequence diagram for hand gesture recognition system would show the interaction between various components in the system, including the camera, preprocessing module, feature extraction module, recognition module, and decision module. It would illustrate how video is captured and processed, features are extracted, and a decision is made based on the recognized gesture. The sequence diagram would also demonstrate how feedback or actions are triggered based on the recognized gesture. Steps are:

- User performs a hand gesture in front of the camera or sensor.
- The webcam or sensor captures the gesture and sends the image data to the image processing module.
- The image processing module analyzes the image data and extracts the key features of the hand gesture.
- The extracted features are sent to the gesture recognition module for classification.
- The gesture recognition module compares the extracted features with the trained gestures in the gesture database.
- If the extracted features match a gesture in the database, the corresponding action or command is sent to the application or system.
- If the extracted features do not match any gesture in the database, an error message is displayed or an appropriate action is taken.

4.1.4 Hardware and software requirement

Hardware Requirements
All the physical equipments i.e. input devices, processor, and output device inter connecting processor of the computer is called as hardware.
Hard Disk minimum of 40 GB.
RAM minimum of 2 GB.
Dual Core and up ,15" Monitor.
Integrated webcam or external webcam (15 -20 fps).

Software Requirements
A set of instructions or program required to make hardware platform suitable for desired task is known as software. Software can also be defined as the utility programs that are required to drive hardware of computer.
Operating system- Microsoft Windows 7 SP 1 or above.
Microsoft Visual Studio 2010.

MinGW and Visual C++ compilers (for Windows).
Supporting Webcam Drivers.

4.2 System architecture

4.2.1 Block Diagram of SL model

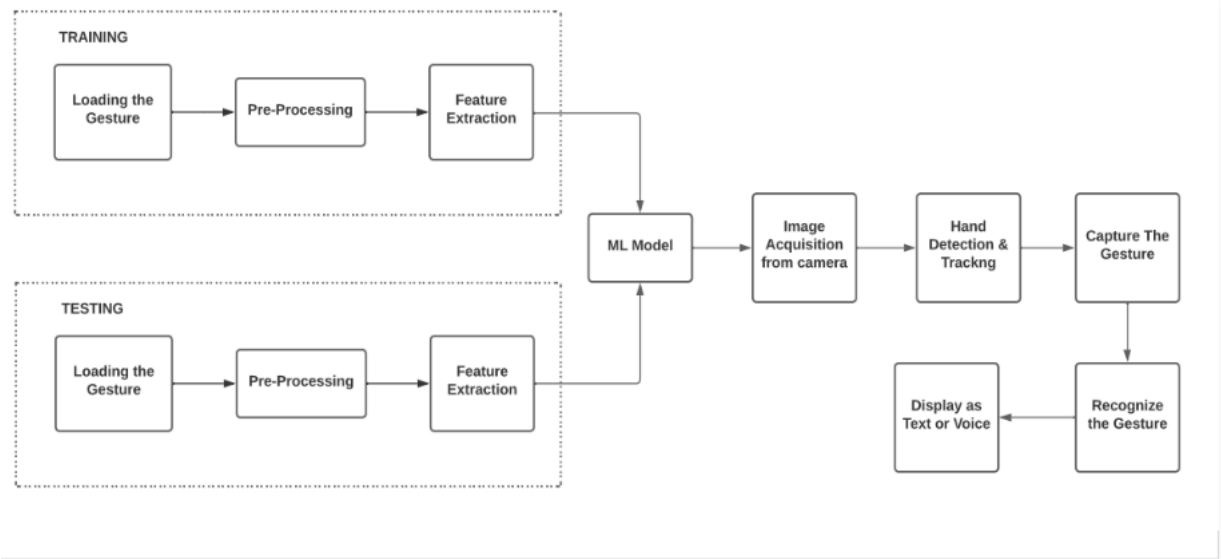


Figure 4.4: Block Diagram of SL model

The block diagram for hand gesture recognition system typically includes four main blocks: image acquisition, preprocessing, feature extraction, and recognition. The image acquisition block captures video of the user's hand gestures, which is then preprocessed to enhance image quality. Feature extraction block identifies key features of the preprocessed image, which are compared to a database of known gestures in the recognition block to determine the gesture being performed.

Chapter 5

Result and Discussion

5.1 Screenshots of the System

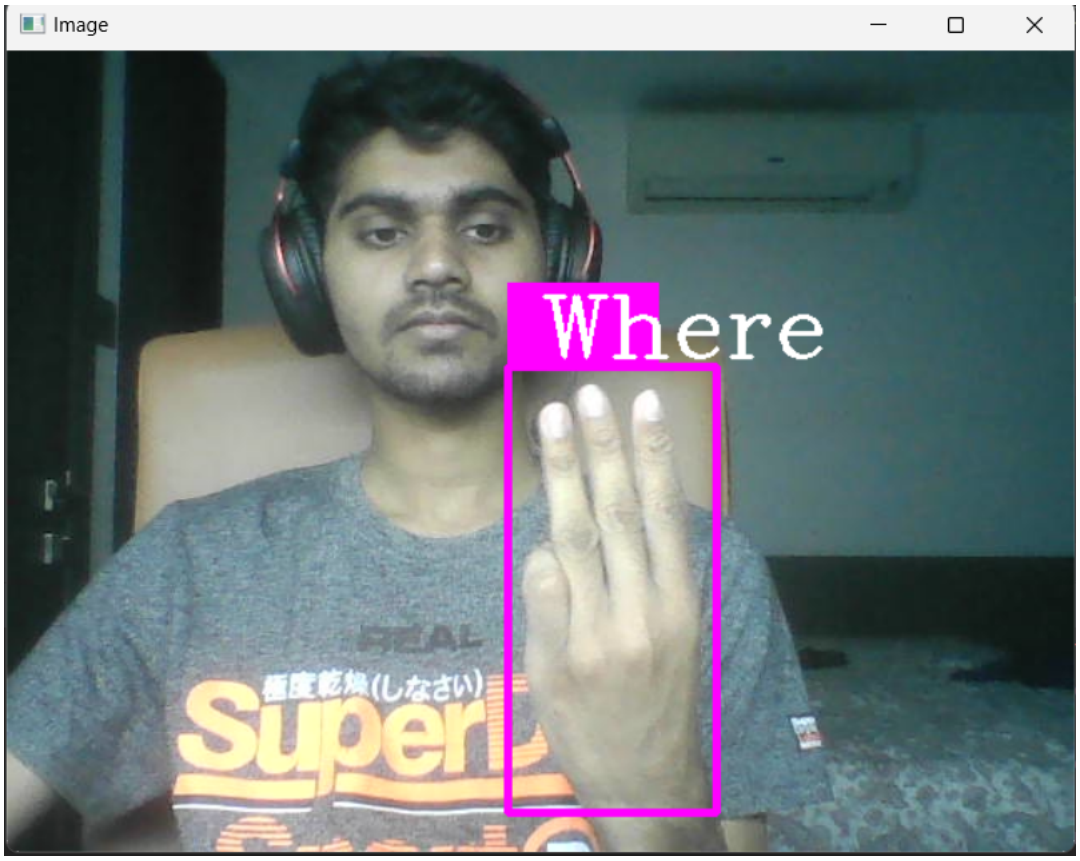


Figure 5.1: Gesture Detection 1

Figure 5.2: Gesture Detection 2

Figure 5.3: Gesture Detection 3

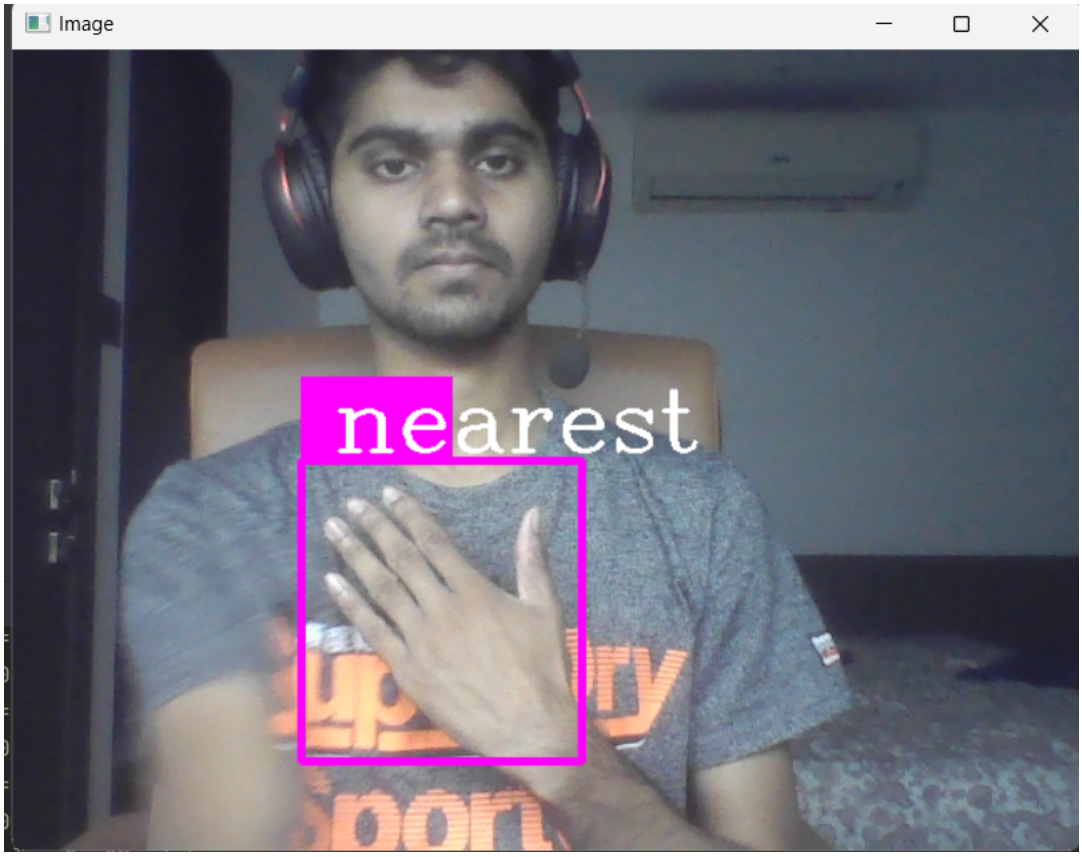


Figure 5.4: Gesture Detection 4

Figure 5.5: Gesture Detection 5

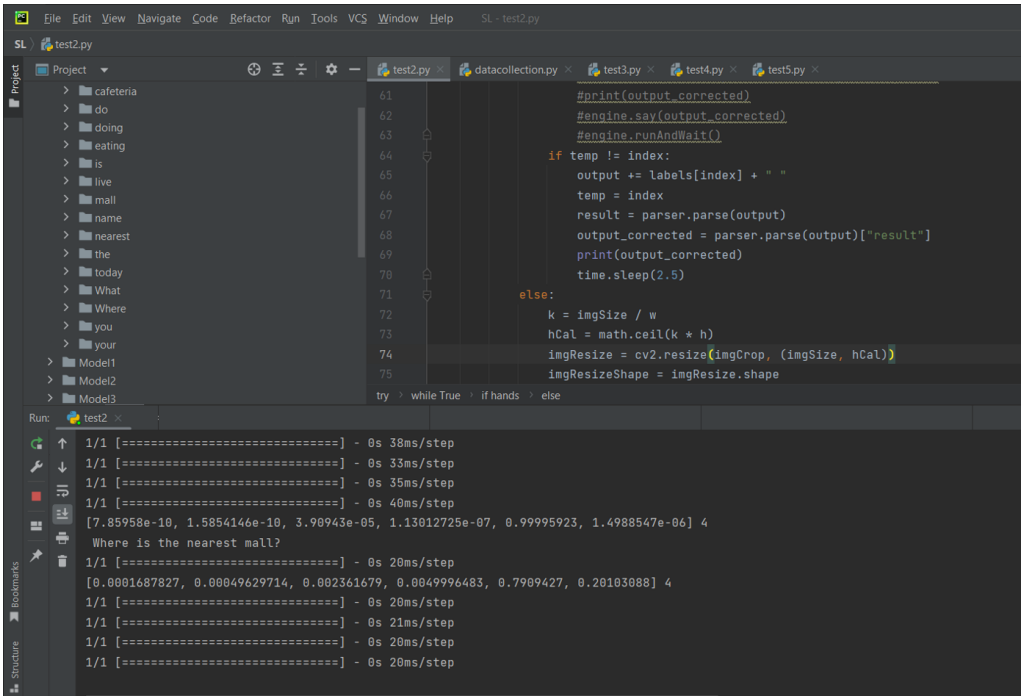


Figure 5.6: Sentence Formation

5.2 Sample Code (of imp part/ main logic)

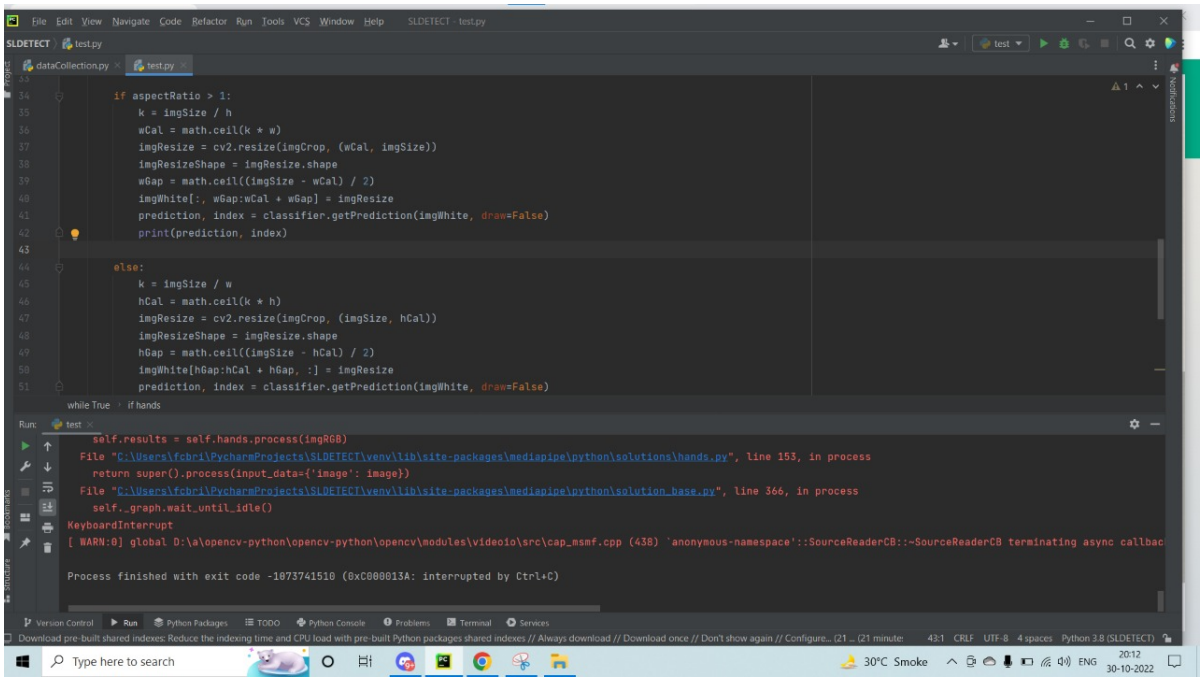


Figure 5.7: Training Model

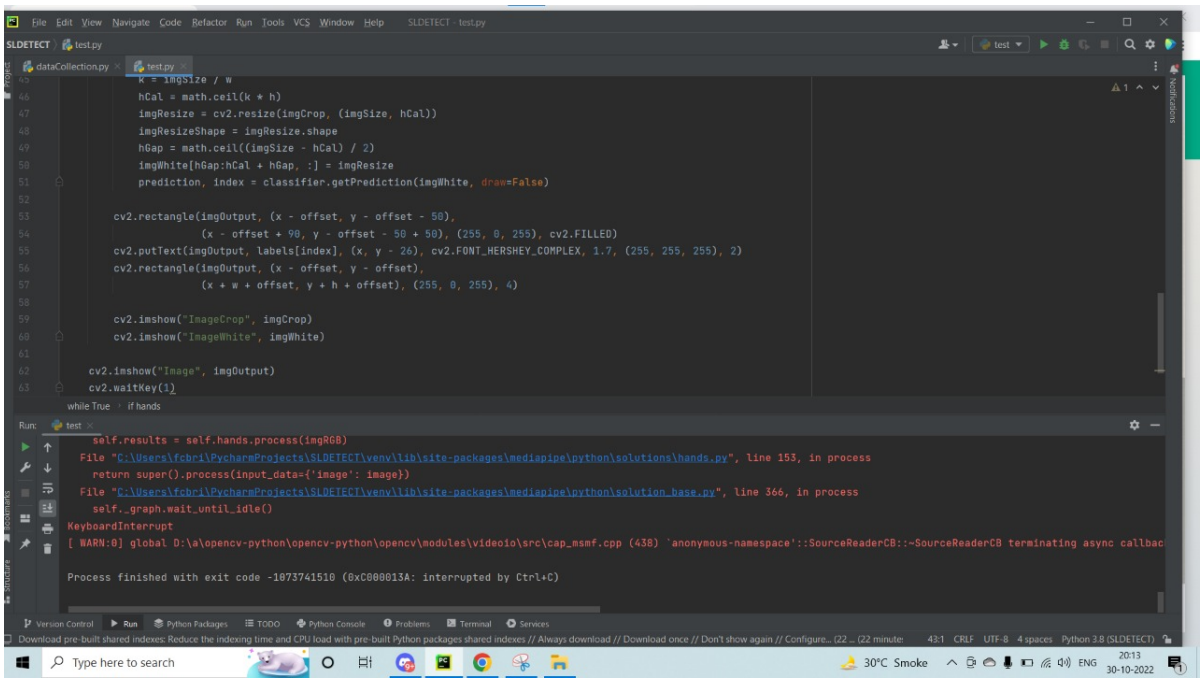


Figure 5.8: Training Model

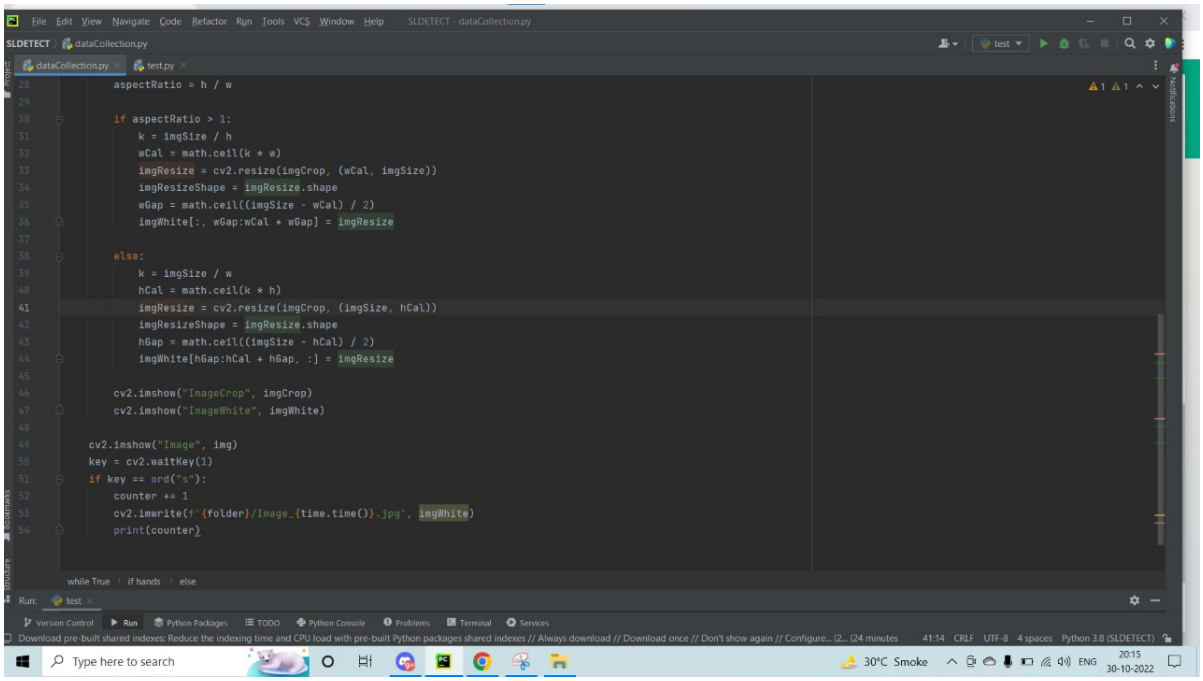


Figure 5.9: Testing Model

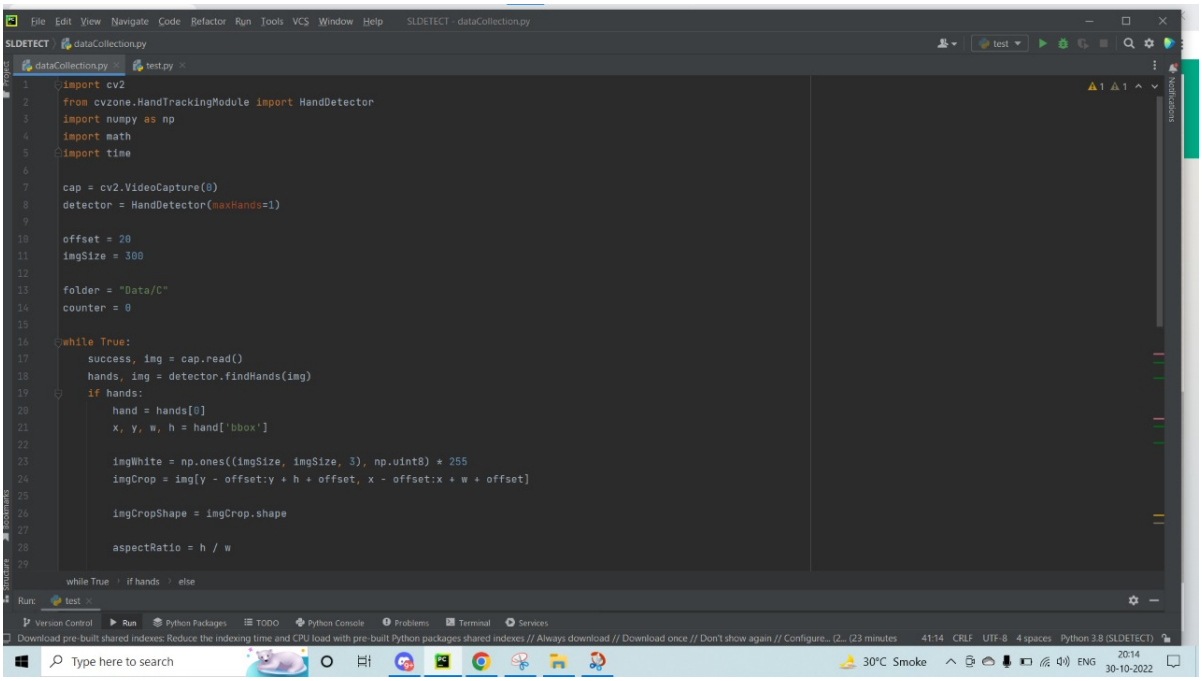


Figure 5.10: Testing Model

Chapter 6

Conclusion

6.1 Conclusion:

In this report, different sign languages have been detected using TensorFlow. The model was trained using google's Teachable Machine Learning. Approximately 300 images for each field was used to train the data. Overall the model gave high accuracy on training as well as testing. The accuracy for training and testing was almost similar. This project implemented a hand gesture recognition system using deep learning, which can detect and classify various hand gestures in real-time. The system used a hand detector module to locate the hand in the frame and a classification module to predict the gesture. The model was trained using a dataset of images for different hand gestures, and it achieved an impressive accuracy of 83%. The code is efficient, easy to understand, and can run on any system with a webcam. Moreover, it offers the possibility to train and test different models by changing the dataset and can be adapted to different applications, including gaming, sign language recognition, and human-robot interaction. In summary, this hand gesture recognition system offers a practical solution for detecting and recognizing hand gestures in real-time, with a high accuracy of 83%. Its simplicity,speed, and versatility make it an essential tool for researchers,developers, and enthusiasts interested in exploring the potential of hand gesture recognition.

6.2 Future Scope:

Future work may involve improving the accuracy of the system by collecting more data, augmenting the existing dataset, and exploring other deep learning architectures. Additionally, integrating the system with other computer vision techniques, such as object detection or facial recognition, may enhance its capabilities.

References

[1] P. Bao, A. I. Maqueda, C. R. del Blanco, and N. García, "Tiny hand gesture recognition without localization via a deep convolutional network," *IEEE Transactions on Consumer Electronics*, vol. 63, no. 3, pp. 251–257, 2017.

[2] J. Molina, J. A. Pajuelo, and J. M. Martínez, "Real-time motion-based hand gestures recognition from time-of-flight video," *Journal of Signal Processing Systems*, vol. 86, no. 1, pp. 17–25, 2017.

[3] A. Licsár and T. Szirányi, "User-adaptive hand gesture recognition system with interactive training," *Image and Vision Computing*, vol. 23, no. 12, pp. 1102–1114, 2005.

[4] H. Gunawan, N. Thiracitta, A. Nugroho, *et al.*, "Sign language recognition using modified convolutional neural network model," in *2018 Indonesian Association for Pattern Recognition International Conference (INAPR)*, pp. 1–5, IEEE, 2018.

[5] J. Krejsa and S. Vechet, "Czech sign language single hand alphabet letters classification," in *2020 19th International Conference on Mechatronics-Mechatronika (ME)*, pp. 1–5, IEEE, 2020.

[6] G. Li, H. Tang, Y. Sun, J. Kong, G. Jiang, D. Jiang, B. Tao, S. Xu, and H. Liu, "Hand gesture recognition based on convolution neural network," *Cluster Computing*, vol. 22, no. 2, pp. 2719–2729, 2019.

[7] K. Bantupalli and Y. Xie, "American sign language recognition using deep learning and computer vision," in *2018 IEEE International Conference on Big Data (Big Data)*, pp. 4896–4899, IEEE, 2018.

[8] S. M. Mankar and S. A. Chhabria, "Review on hand gesture based mobile control application," in *2015 International Conference on Pervasive Computing (ICPC)*, pp. 1–2, IEEE, 2015.

[9] M. Bilgin and K. Mutludoğan, "American sign language character recognition with capsule networks," in *2019 3rd International Sympo-*

sium on Multidisciplinary Studies and Innovative Technologies (ISM-SIT), pp. 1–6, IEEE, 2019.

[10] "`https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-tensorflow`".

[11] R. Devika, S. Vairavasundaram, C. S. J. Mahenthar, V. Varadarajan, and K. Kotecha, "A deep learning model based on bert and sentence transformer for semantic keyphrase extraction on big social data," *IEEE Access*, vol. 9, pp. 165252–165261, 2021.

Acknowledgement

Success of a project like this involving high technical expertise, patience and massive support of guides, is possible when team members work together. We take this opportunity to express our gratitude to those who have been instrumental in the successful completion of this project. We would like to show our appreciation to **Ms. Shagufta Rajguru** for their tremendous support and help, without them this project would have reached nowhere. We would also like to thank our project coordinator **Dr. Pravin Rahate** for providing us with regular inputs about documentation and project timeline. A big thanks to our HOD **Dr. Lata Ragha** for all the encouragement given to our team. We would also like to thank our principal, **Dr. S. M. Khot**, and our college, **Fr. C. Rodrigues Institute of Technology, Vashi**, for giving us the opportunity and the environment to learn and grow.

Project Group Members:

1. Joel Thomas, 1019133

2. Aditya Jadhav, 1019131

3. Tushar Gupta, 1019127

Appendix A : Timeline Chart

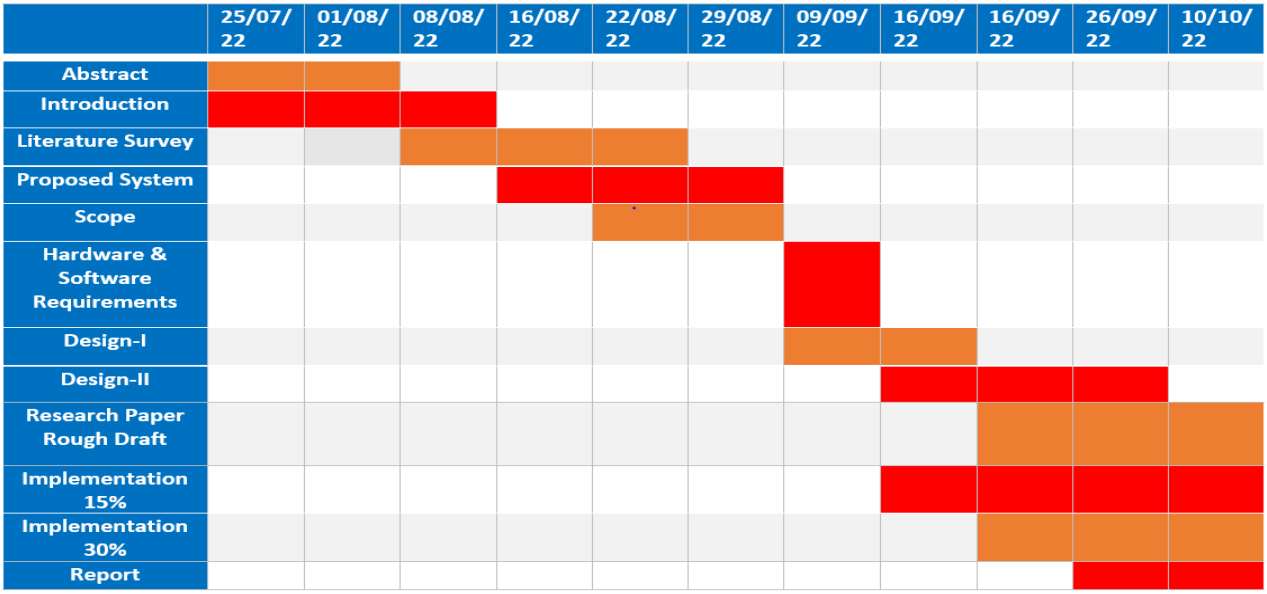


Figure 6.1: Timeline Chart for Semester VII

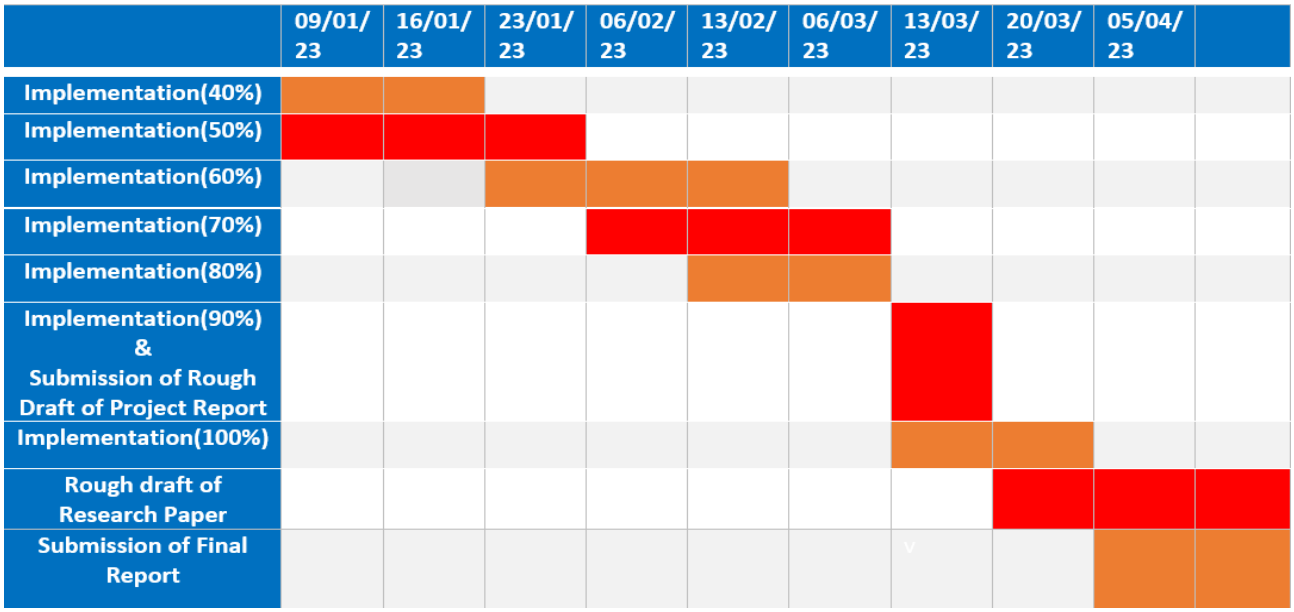


Figure 6.2: Timeline Chart for Semester VIII