HAND GESTURE RECOGNITION USING MACHINE LEARNING AND COMPUTER VISION

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

In

Computer Science and Engineering/Information Technology

By

Arnav Sharma (171222)

Under the supervision of

Dr. Yugal Kumar

To

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report entitled "**HAND GESTURE RECOGNITION USING MACHINE LEARNING AND COMPUTER VISION**" in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering and Information Technology submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from **January 2021 upto May 2021** under the supervision of **Dr. Yugal Kumar, Assistant Professor (Senior Grade),** Computer Science and Engineering/Information Technology.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Arnav Sharma, 171222

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Dr. Yugal Kumar

Assistant Professor (Senior Grade)

Computer Science & Engineering and Information Technology

Dated: 17/05/2021

ACKNOWLEDGEMENT

We would like to express our deep sense of gratitude to **Dr. Yugal Kumar (Department of Computer Science & Engineering and Information Technology)** our supervisor for this work, for his continual guidance and motivation from the inception of this project (*Hand Gesture Recognition Using Machine Learning and Computer Vision*) till the end.

We are also thankful to **Dr. Samir Dev Gupta, Head of Dept. of CSE and IT** for providing us with this opportunity and his support during our work. Also, we'd like to thank all the professors and other staff members of the Department of Computer Science Engineering and Information Technology for their generous assistance in various ways that helped in the successful completion of this work.

We are also thankful to **Mr. Ravi Raina**, for his support and guidance throughout the process.

Finally, we would also like to appreciate our fellow mates who willing helped us out with their abilities whenever we faced issues.

ABSTRACT

A major shortcoming in our society is a social barrier between the differently abled members of the society and the abled folks. One of the most important aspects of human beings, being regarded as social animals, is in fact communication. Communication is also a major obstacle faced by the hearing and vocal disabilities people. This inability to communicate leads to frequent problems and hinders the daily activities of a person with hearing and vocal disabilities.

The underlying reason for this disparity is that abled folks don't learn and aren't taught Sign Language which is the main means of communication for a person with hearing and vocal disabilities. Thus, abled folks are incapable of having a normally fluent conversation with these different sections of the society. Consequently, in a verbal exchange among hearing and speech impaired individuals and an able person the convenience of communique and consequently the consolation degree is hampered.

So, in our project, we have proposed a cost-efficient solution to overcome this communication barrier. This solution can be easily used by everyone and can also be, with some modifications, made to work on most platforms which have a camera module. Our approach uses the integrated camera module to capture real time hand gestures based on hand key points or landmarks and the algorithm using machine learning techniques, displays the alphabet that the gesture is representing.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

CNN– Convolutional Neural Networks

ANN– Artificial Neural Networks

SVM – Support Vector Machine

k-NN – k-Nearest Neighbors

DT – Decision Tree

RMSE – Root Mean Square Error

MAE – Mean Absolute Error

MSE – Mean Squared Error

RGB – Red Green Blue

HSV – Hue, Saturation and Value

SURF – Speed Up Robust Features

LIST OF FIGURES

LIST OF TABLES

Chapter 1
INTRODUCTION

1.1 Introduction

In the past few years, huge advancements have been made in the fields of science and technology. Not only this, technology has got much cheaper and its availability has widened as it is now available to the common man. So, it is vital to no longer overlook the duty of our generation to make use of this accessibility to technology to contribute to the progress and improvement of society at large.

Human beings have, since the beginning of time, been described as a social animal. As a social being, one of the principal aspects of our life is communication. Social interaction or simply communication has always been regarded as one of the major aspects of living a happy life. For an individual to live a normal lifestyle, communication is necessary and is required for almost all of our daily tasks. But there is a not so blessed segment of society which faces hearing and vocal disabilities. A hearing-impaired individual is one who either can't hear at all or is able to hear sounds which are above a certain frequency, or what we'd generally call as 'can only hear when spoken to loudly'. An individual with the inability to speak due to any reason whatsoever is considered as a mute or silent person.

In an enormous research conducted in diverse domain names, it turned into determination that impairments such as hearing-impairment, vocal-impairment or the ineptitude to express oneself causes loss of opportunities for such people when compared to able people. Not only does it lead to this, but also hinders day-to-day activity of an individual such as normal conversations.

According to MoSPI, Govt. of India [1], in 2002 about 30.62 lakh of the then population were suffering from hearing disorder and 21.55 lakh of the then population were suffering from speech disorder.

Another 2001 Census [2] states that around 21 million Indian citizens (which constituted 12.6 million males and 9.3 million females approximately), that is, about 2.1 per cent of the then population of India, were facing certain disabilities. People with speech disability accounted for the 7.5 per cent while those with hearing disability accounted for 5.8 per cent of these 21 million people in total.

These statistics also show evidence of the problems and discrimination faced by these people. Additionally, they also provide us with a wealth of facts about specific kinds of disabilities, the number of humans tormented by these disabilities and the barriers they face in their life. One of the foremost boundaries a disabled Individual faces in his existence is incapable of talking with an everyday man or woman.

So with our knowledge of technology, we hope to help such people through our project so that they are able to communicate normally with others.

1.2 Problem Statement

A major shortcoming in our society is a social barrier between the differently abled members of the society and the abled folks. One of the most important aspects of human beings, being regarded as social animals, is in fact communication. Communication is also a major obstacle faced by the hearing and vocal disabilities people. This inability to communicate leads to frequent problems and hinders the daily activities of a person with hearing and vocal disabilities.

The underlying reason for this disparity is that abled folks don't learn and aren't taught Sign Language which is the main means of communication for a person with hearing and vocal disabilities. Thus, abled folks are incapable of having a normally fluent conversation with these different sections of the society. Consequently, in a verbal exchange among hearing and speech impaired individuals and an able person the convenience of communique and consequently the consolation degree is hampered.

So, in our project, we have proposed a cost-efficient solution to overcome this communication barrier. This solution can be easily used by everyone and can also be, with some modifications, made to work on most platforms which have a camera module. Our approach uses the integrated camera module to capture real time hand gestures based on hand key points or landmarks and the algorithm using machine learning techniques, displays the alphabet that the gesture is representing.

1.3 Existing Solutions:

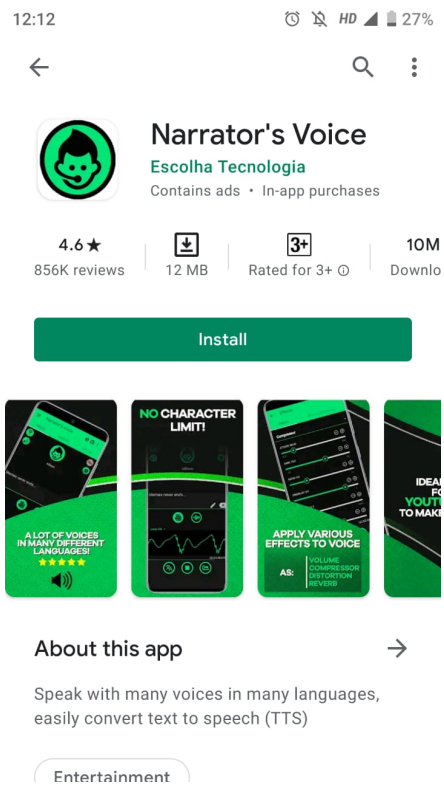
1.3.1 Applications in Apple Store and Google Play Store:

Prior to us, various organizations and individual developers have all attempted to solve this problem faced by the people with hearing and vocal disabilities using various different techniques. Some of the attempts are as follows:

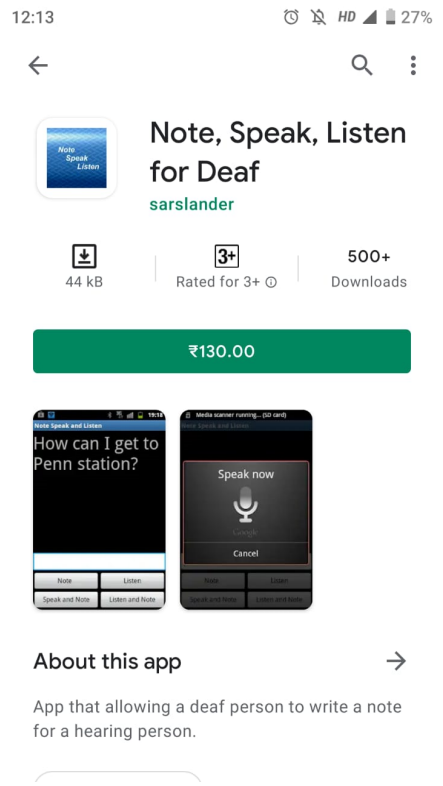
- Audio to text conversion programs,
- Applications to interpret sign language, and
- Various standard american sign language guides.

But none of these approaches were able to completely solve the problem as these applications are not accessible by all. Also each of these techniques had some shortcomings and weren't totally foolproof, like taking input as text from the user which is a tedious task for long sentences and then generating the audio as output. Some others display the corresponding sign for the entered alphabet. Not only this, but these applications are constrained to the english language only, which is not readable by everyone.

Some of the applications which are available in Google Play Store are- Virtual Voice, Note Speak Listen for Deaf, Sign Language Interpreter, Sign Short Message Service etc.



(a) Narrator Voice App



(b) Note, Speak, Listen for deaf

Figure 1.1 Some Sample Applications

1.3.2 Gestures Gloves:

To convert the gestures into command, the gesture gloves were made. Which can change the gestures into the signals. In this gloves flex sensor is used , the flex sensor measures the bending or deflection of the finger and maps it to the corresponding signal. The drawback of these gloves are that they are costly and everyone cannot afford them. Also additional hardware like lcd display, buzzer etc are required to change these signals into the corresponding gestures.

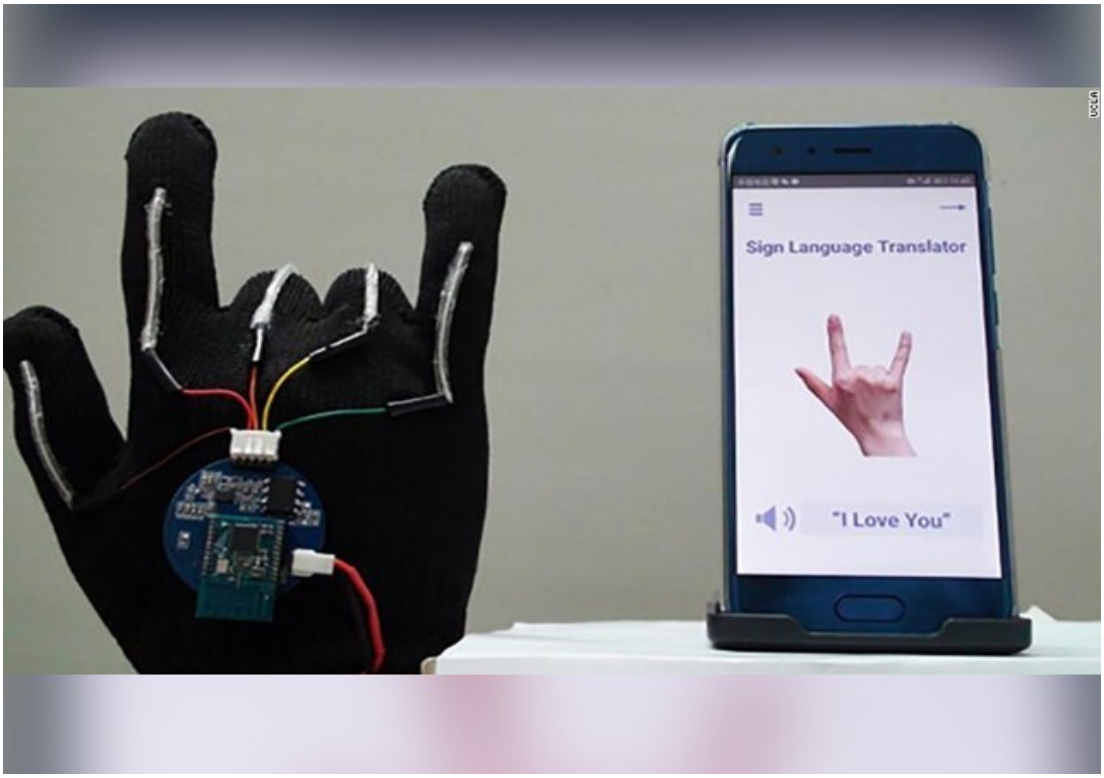


Figure 1.2: Gesture gloves

1.4 Proposed Solution

In this project, a cost efficient solution is proposed to overcome the communication barrier. This solution can not only be just just for recognizing the American Sign Language hand gestures but can also be modified for various other purposes. Our solution is an easy to use one as it uses your device's camera (be it the webcamera of the laptop or the camera of a smartphone) and by applying a few algorithms of Machine Learning and Computer Vision, it recognizes what hand gesture is being shown and displays it in textual form that is legible to any individual who knows the english alphabet.

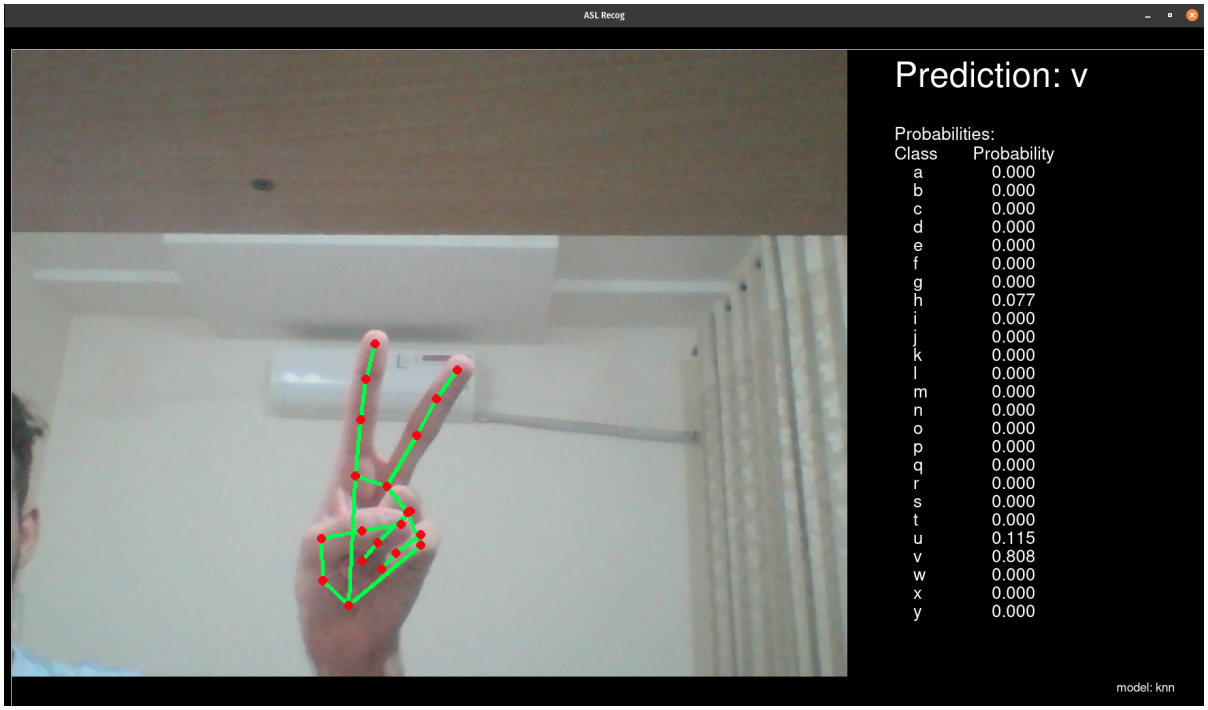


Figure 1.3: Our proposed application screenshot

The solution provided here is cheap, easily available and is easy to use by a common man. All one needs to do is run the program and do the gesture in front of the camera and the algorithms will do their work in the backend and convert those gestures into readable english alphabets.

1.5 Methodology

Any machine learning based application can be summed up to have at least three phases - data collection and preprocessing phase, training phase and visualization.

Our program also follows these steps in order. At first, the data is collected and a base dataset is prepared. This dataset is then divided into training data and testing data which in our case is a multi-label classification data as we have to predict 26 gestures. To generate our dataset, we've collected hand keypoints from images for each gesture using the laptop's web camera. Features are then selected and extracted from the training data. The next step is to decide which machine learning models to use. Since ours is a multi-class classification problem, the models used were K-Nearest Neighbours, Support Vector Machine and Decision Tree. These models are then trained on the training set. Then they are made to make predictions on the test set based on which

their performance is evaluated and changes are made to the parameters so as to squeeze out the best results from these models.

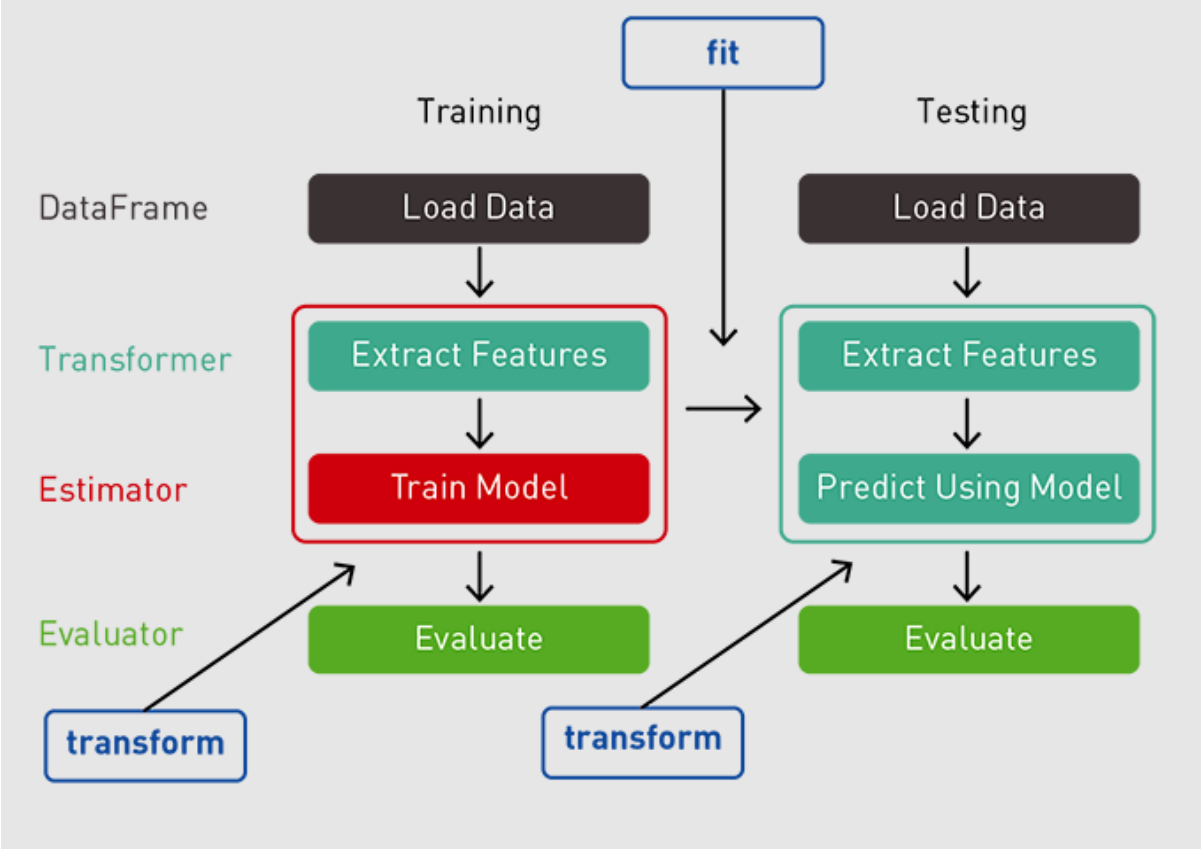


Figure 1.4: Machine Learning pipeline

1.5.1 OpenCV

OpenCV is short for Open Source Computer Vision Library which is a library for achieving actual-time programs. OpenCV is written in C and C++ and is cross-platform, that is, it works on all machines regardless of the operating system that is installed on that machine. And it is available as a library for languages such as Java, Python, C++, etc. As it is an open source project, it is freely available at http://sourceforge.net/projects/opencvlibrary/.

The creation of OpenCV is credited to Grad Bradsky of Intel. It was implemented in 1999, with only one mission in mind, that is, to encourage research in the field of computer vision and also to make computer vision available freely even for commercial applications.

Being an open source project, OpenCV also has its documentation available on the web residing at http://opencv.willowgarage.com/documentation/index.html.

A digital image is an array of discrete values or a matrix of light intensities taken or capture by a device like camera and are organized into a two-dimensional matrix of pixels, in such a way where each of the pixels is represented by a number, generally ranging from 0 - 255 (255 due to it being 8-bit). All of this may be stored in the picture formats like jpg and gif [8].
OpenCV uses its own custom data structure, *IplImage* to represent an image. This data structures has various accessible fields such as:

- `width` – an integer showing the width of the image in pixels
- `height` – an integer showing the height of the image in pixels
- `imageData` – a pointer to an array of pixel values
- `nChannels` – an integer showing the number of colors per pixel
- `depth` – an integer showing the number of bits per pixel
- `widthStep` – an integer showing the number of bytes per image row
- `imageSize` – an integer showing the size of in bytes

1.5.2 MediaPipe Hands

MediaPipe Hands employs machine learning to deduce 21 3-dimensional landmarks of the hand using just a single frame. Thus, it is regarded as a hi-fi and fairly accurate hand and finger detection and tracking solution as compared to current state-of-the-art models which generally rely on high performing machines. MediaPipe is available on various platforms even on the web and smartphones. MediaPipe Hands is also capable of inferring landmarks of both hands simultaneously.

1.5.3 k-Nearest Neighbours

k-Nearest Neighbours or simply k-NN, is a machine learning model that can be used for both classification and regression problems. The k-NN algorithm is based on the similarities between the dataset and the sample to be predicted and it puts the sample into the category that is the

closest to the categories present in the dataset. It stores the entire dataset and classifies the sample data based on similarity.

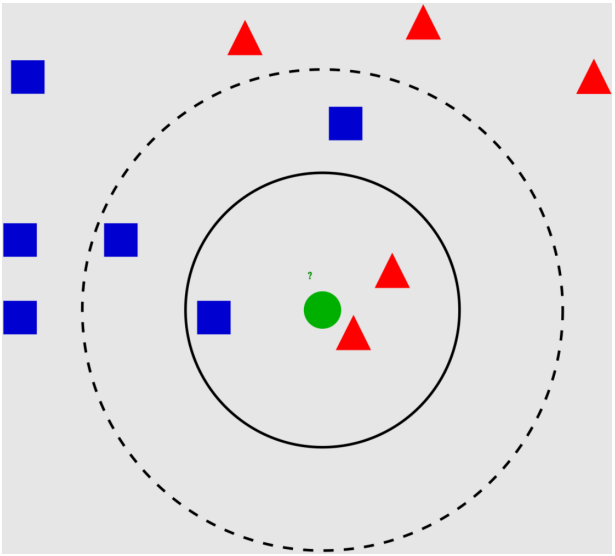


Figure 1.5: k-NN example (green dot is the sample that is to be classified)

1.5.4 Support Vector Machines

Support Vector Machines or simply SVM, is placed under the category of supervised machine learning algorithm. It is generally used for classification problems. In this model, each data item is plotted as a point in a n-D (here, n denotes the number of features in the training dataset) space and then it performs classification by identifying hyper-plane(s) that separates the two classes with the maximal distances.

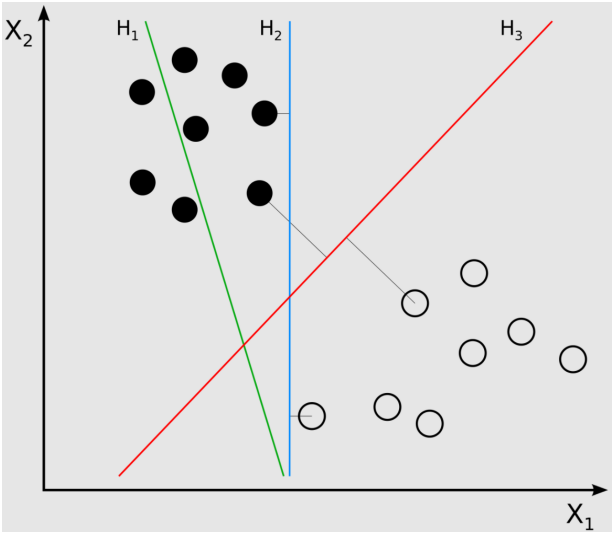


Figure 1.6: SVM example (H_1 doesn't separate the classes, H_2 does but the distances are not even whereas H_3 separates the classes with maximum distances either side)

1.5.5 Decision Tree

A decision tree is a tree-like model that supports in making decisions. It consists of a tree with decisions and possible outcomes, etc. A decision tree model uses a decision tree to reach the conclusions about an item using a set of decisions from the decision tree.

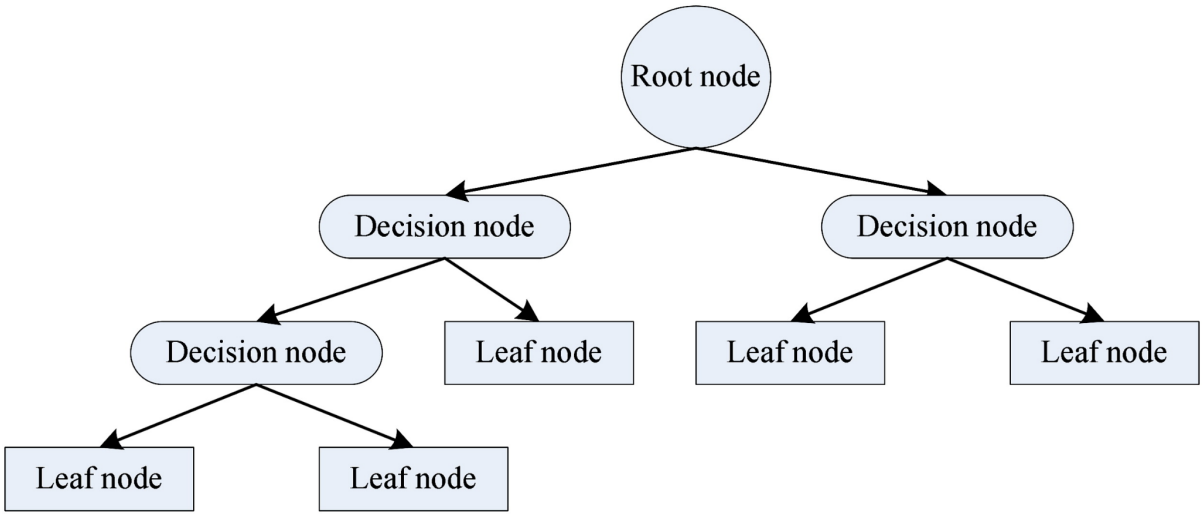


Figure 1.7: Decision Tree model representation

1.6 Organization

In Chapter 1, we have discussed the problem that is being faced by the persons with disability, how can communication be a barrier to the persons with disability. And proposed the solution to overcome this barrier.

In Chapter 2, we have discussed the research papers we have referred to in order to get a better understanding of our project. The papers mainly focus on techniques used in machine learning and other researches carried out in this field.

In Chapter 3, we have cited the possible requirements that are the hardware and software system, what language we will be using and where we are going to implement it along with the libraries required along with details about the platform used.

In Chapter 4, we have discussed the algorithms in details and also the approaches used to predict the outcome and effectiveness of our result. Implementations and the results of the outputs have been discussed.

In Chapter 5,we have given the conclusions that have been derived from this study and the future scope of this project.

Chapter 2
LITERATURE REVIEW

2.1 Sign Language Recognition System to aid Deaf-dumb People Using PCA [8]

2.1.1 Author: Shreyashi Narayan Sawant

2.1.2 Publication: International Journal of Computer Science & Engineering Technology, 2014

Summary:

1. 26 gestures from Indian Sign Language, using MATLAB with WebCam live capture.
2. 260 images in total, 10 of each sign. Captured with white background so as to avoid illumination effects.
3. Image segmentation using Otsu's method (a method for quantization), noise removal & contour smoothing.
4. Centroid is calculated (using image moment) to separate the hand part from the arm.
5. Skin detection using HSV color model.
6. Feature extraction using PCA (eigenvector matrix).
7. Finally, subject gesture is normalized with respect to the average gesture and then projected onto gesture space using the eigenvector matrix. Finally, Euclidean distance is computed between this projection and all known projections. The minimum value of these comparisons is selected for recognition during the training phase.

2.2 Hand Gesture Recognition for Deaf People Interfacing [9]

2.2.1 Author: Isaac Garcia Incertis, Jaime Gomez Garcia-Bermejo, Eduardo Zalama Casanova

2.2.2 Publication: International Journal of Computer Science & Engineering Technology, 2014

Summary:

1. The work is carried on the static gesture case corresponding to the alphabet letters in Spanish Sign Language(LSE).
2. On captured images, hand regions and corresponding contours are extracted through color segmentation.
3. Contours are Sampled at every arc distance.
4. The resulting points are compared to those of a target gesture in the dictionary.
5. The comparison is performed on the basis of four Distance Criteria- L0 norm, L1 norm , L2 Norm , L inf norm.
6. A positive identification is assumed for the closest model.

2.3 An American Sign Language Detection System using HSV Color Model and Edge Detection [10]

2.3.1 Author: A. Sharmila Konwar, B. Sagarika Borah, C. Dr.T.Tuithung

2.3.2 Publication: International Conference on Communication and Signal Processing, April 3-5, 2014, India

Summary:

1. The aim is to build a user-friendly human-computer interface in which the computer can understand the human language. .
2. The system is described into two phases- training phase and testing phase.

3. In the first phase , a database is created by capturing the images by web camera or any camera followed by preprocessing, feature extraction and training.
4. In the second phase , image acquisition, preprocessing, feature extraction and classification which was based on the testing phase is done
5. Canny edge detection algorithm is used to detect hand gestures.
6. PCA and ANN were used for the feature extraction and recognition part respectively.
7. The project was able to detect five alphabets successfully.
8. Due to the geometric variation and uneven background and lighting conditions, some images were not detected successfully.

2.4 Real Time Hand Gesture Recognition Using Different Algorithms Based on American Sign Language [11]

2.4.1 Author: Md. Mohiminul Islam,Sarah Siddiqua,and Jawata Afnan3

2.4.2 Publication: IEEE

Summary:

1. Different algorithms are used for the feature detection. Some of the algorithms are K convex hull for fingertip detection, pixel segmentation, eccentricity, elongatedness of object etc.
2. Apart from the K-convex hull algorithm, many other algorithms were also used to get better accuracy.
3. Images are taken using a mobile phone, over that dataset ANN was used.
4. The ANN is trained on 1850 sample images.
5. The model was trained on the real time environment.
6. The proposed model was able to detect ASL alphabets and numbers with the accuracy of 94.32%.
7. Further, the model is improved for the movement detection of the hand for word

recognition.

Chapter 3

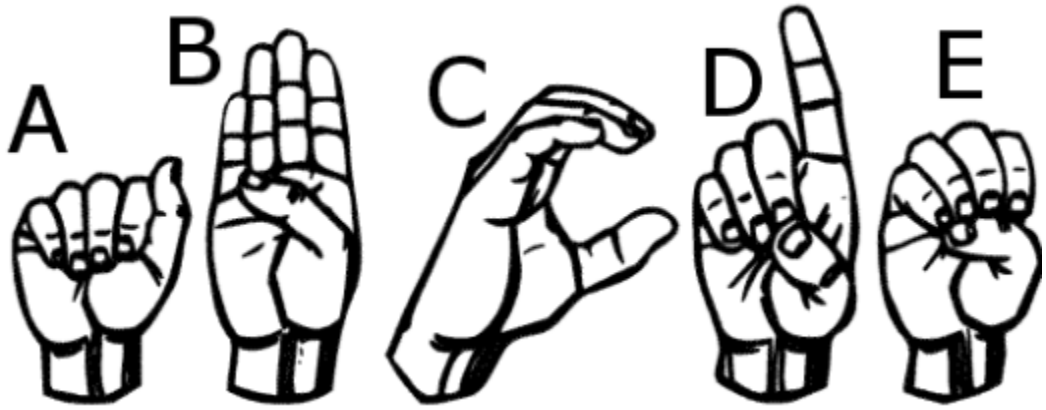
SYSTEM DEVELOPMENT

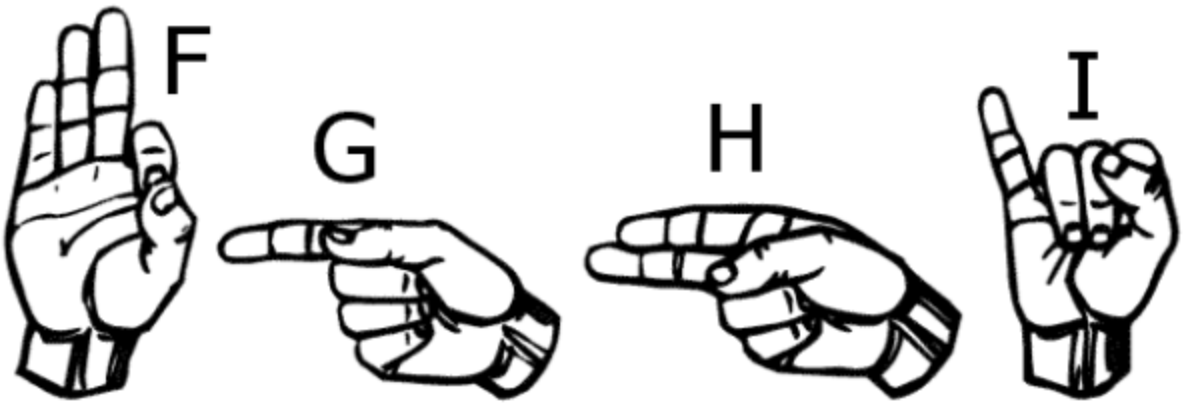
3.1 Dataset

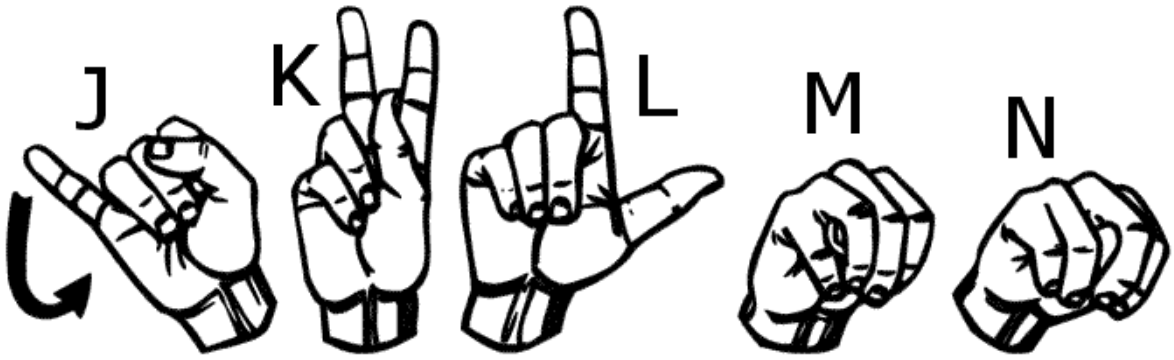
The first step in any machine learning problem is to gather the data. The data can either be taken from some open source datasets from websites such as Kaggle or you can prepare your own dataset. In our case, we created our own dataset from scratch. For the data gathering process, we took x- and y-coordinates of 21 hand keypoints using the MediaPipe and OpenCV libraries. For each gesture, the following x and y keypoints were collected:

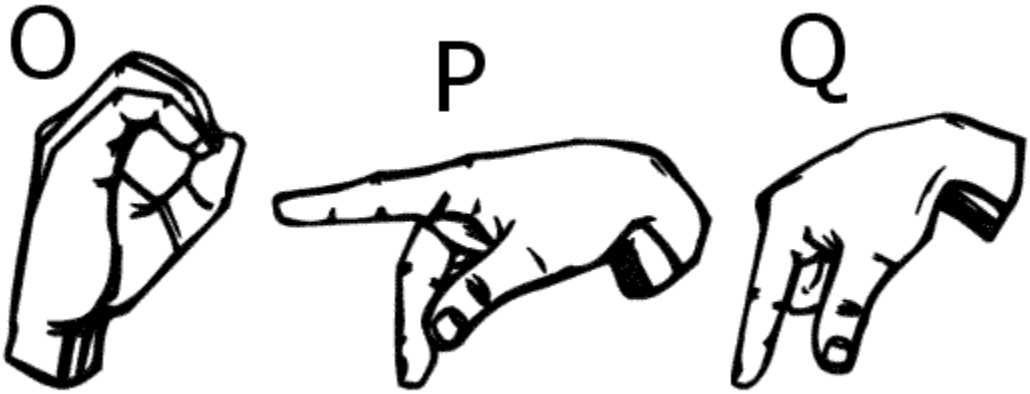
- Wrist
- Thumb
- Index finger
- Middle finger
- Ring finger
- Pinky finger

Below are some sample items from the dataset for each gesture: [12]




F
G
H
I


J
K
L
M
N


O
P
Q

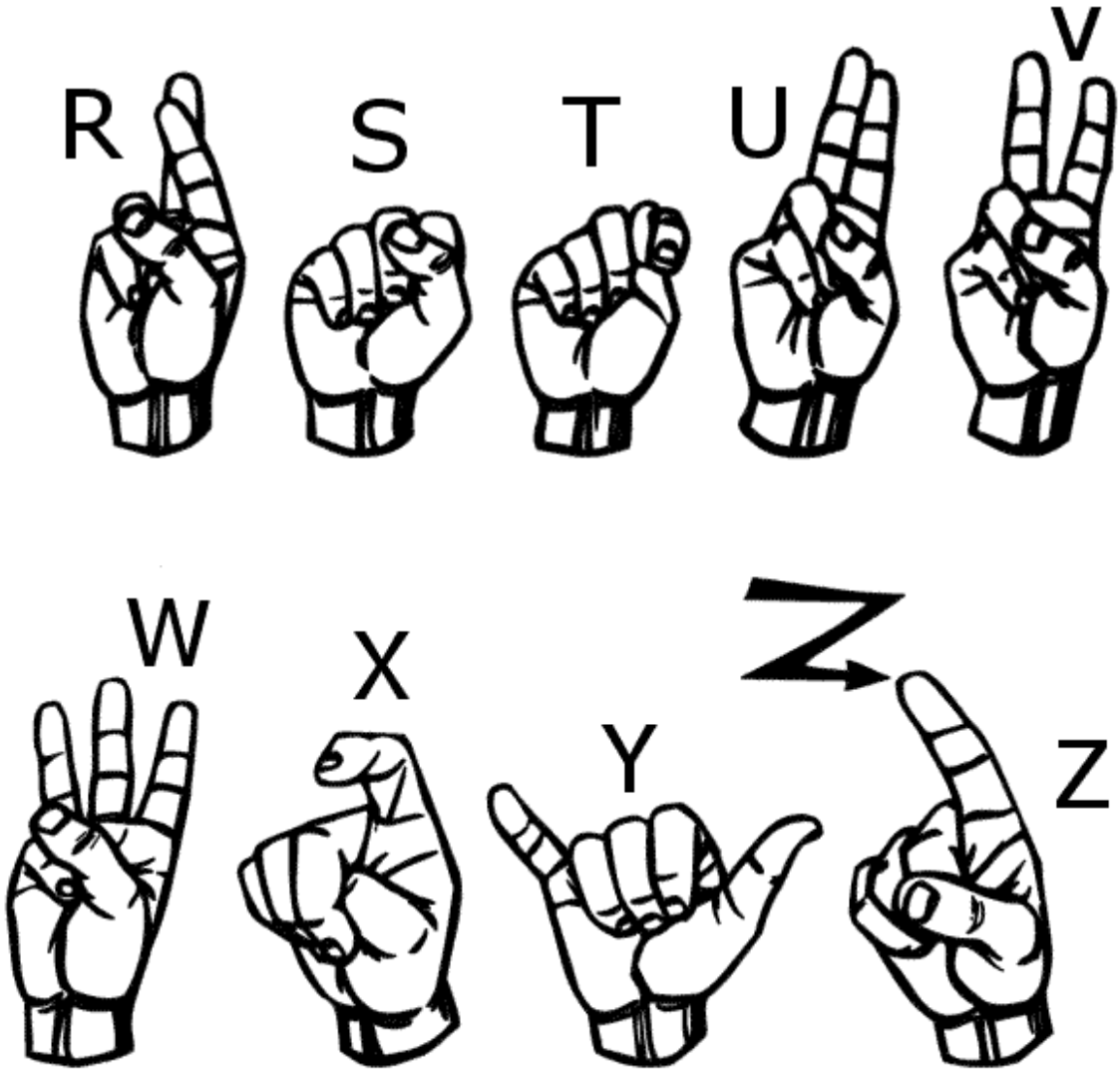


Figure 3.1: American Sign Language Gestures

CLASS	a	m	s
WRIST.x	0.519844353	0.466303408	0.632597446
WRIST.y	0.631177843	0.742914438	0.764358461
THUMB_CMC.x	0.438432842	0.391019225	0.526382744
THUMB_CMC.y	0.598431528	0.71738255	0.696385562
THUMB_MCP.x	0.379219174	0.341336727	0.457116485
THUMB_MCP.y	0.494120657	0.585350513	0.532129169
THUMB_IP.x	0.365754008	0.353561819	0.512052238
THUMB_IP.y	0.388703167	0.473028243	0.414178878
THUMB_TIP.x	0.350170732	0.378024191	0.611743271
THUMB_TIP.y	0.305573732	0.390994787	0.470294088
INDEX_FINGER_MCP.x	0.416050792	0.348028541	0.510687172
INDEX_FINGER_MCP.y	0.393216163	0.476258487	0.45892635
INDEX_FINGER_PIP.x	0.390594125	0.32041958	0.488550484
INDEX_FINGER_PIP.y	0.370393634	0.45601368	0.361100793
INDEX_FINGER_DIP.x	0.402980924	0.336573243	0.496570438
INDEX_FINGER_DIP.y	0.449733049	0.531668127	0.462251693
INDEX_FINGER_TIP.x	0.416765749	0.350674719	0.505773246
INDEX_FINGER_TIP.y	0.509701431	0.578799188	0.497640997
MIDDLE_FINGER_MCP.x	0.468968391	0.40837127	0.572409093
MIDDLE_FINGER_MCP.y	0.378663152	0.46288839	0.449965298
MIDDLE_FINGER_PIP.x	0.44774133	0.39647153	0.555936098
MIDDLE_FINGER_PIP.y	0.376232386	0.470796227	0.35556528
MIDDLE_FINGER_DIP.x	0.466828942	0.403575242	0.555517077
MIDDLE_FINGER_DIP.y	0.481974155	0.570864975	0.473611593
MIDDLE_FINGER_TIP.x	0.484939307	0.41103974	0.559561849
MIDDLE_FINGER_TIP.y	0.554076195	0.593323171	0.50836122
RING_FINGER_MCP.x	0.527718306	0.472055554	0.637236536
RING_FINGER_MCP.y	0.382022113	0.472447336	0.459994316
RING_FINGER_PIP.x	0.505761683	0.463115811	0.631087601
RING_FINGER_PIP.y	0.3780846	0.50056237	0.36356926
RING_FINGER_DIP.x	0.518674433	0.456842184	0.615692437
RING_FINGER_DIP.y	0.482942075	0.599292696	0.485224545
RING_FINGER_TIP.x	0.530949831	0.456284553	0.615781188
RING_FINGER_TIP.y	0.554197907	0.611558676	0.527433932
PINKY_MCP.x	0.588687122	0.535190165	0.706843376
PINKY_MCP.y	0.397633314	0.490768403	0.485003144
PINKY_PIP.x	0.566689491	0.513676822	0.704544365
PINKY_PIP.y	0.379502952	0.520563185	0.39477092
PINKY_DIP.x	0.560907245	0.498765528	0.681206346
PINKY_DIP.y	0.45397675	0.593908489	0.468196809
PINKY_TIP.x	0.562171102	0.498457283	0.676354527
PINKY_TIP.y	0.509734929	0.59618032	0.50912416

Table 3.1: Keypoints for gestures

3.2 Algorithms

This project uses several algorithms which are commonly used in the field of computer vision and machine learning, namely, colour segmentation, labelling, feature extraction, and convolutional neural networks for recognizing the gesture in real time.

3.2.1 Capture Live Video

The first step was to capture live video from the device webcam. For this purpose OpenCV library for Python was used. Frames from a live video can be captured in the following way: [2]

```
import numpy as np
import cv2 as cv

cap = cv.VideoCapture(0)
while True:
    _, frame = cap.read()
    cv.imshow('frame', frame)
    if cv.waitKey(1) == ord('q'):
        break

cap.release()
cv.destroyAllWindows()
```

3.2.2 Creating the Training Data

For creating the training data, we took x- and y-coordinates of 21 hand keypoints using the MediaPipe and OpenCV libraries. For each gesture, the following x and y keypoints were collected: wrist (WRIST), thumb (THUMB_CMC, THUMB_MCP, THUMB_IP, THUMB_TIP), index finger (INDEX_FINGER_MCP, INDEX_FINGER_PIP, INDEX_FINGER_DIP, INDEX_FINGER_TIP), middle finger (MIDDLE_FINGER_MCP, MIDDLE_FINGER_PIP, MIDDLE_FINGER_DIP, MIDDLE_FINGER_TIP), ring finger (RING_FINGER_MCP, RING_FINGER_PIP, RING_FINGER_DIP, RING_FINGER_TIP) and pinky (PINKY_MCP, PINKY_PIP, PINKY_DIP, PINKY_TIP). These values were then stored in *keypoints.csv* which was later loaded into a pandas dataframe and used for training models from

scikit-learn. While capturing the keypoints, the gesture was automatically rotated and slightly varied such as to have better data for a robust model.

	0	3400	2600	4200
CLASS	a	r	n	v
WRIST.x	0.519844	0.648427	0.410579	0.614857
WRIST.y	0.631178	0.940012	0.724817	0.878059
THUMB_CMC.x	0.438433	0.568895	0.329412	0.53715
THUMB_CMC.y	0.598432	0.936431	0.714727	0.847619
THUMB_MCP.x	0.379219	0.509991	0.282891	0.502831
THUMB_MCP.y	0.494121	0.836858	0.588538	0.772382
THUMB_IP.x	0.365754	0.550817	0.330221	0.596222
THUMB_IP.y	0.388703	0.742219	0.46443	0.770321
THUMB_TIP.x	0.350171	0.603992	0.367002	0.678162
THUMB_TIP.y	0.305574	0.6626	0.379596	0.737245
INDEX_FINGER_MCP.x	0.416051	0.523074	0.259804	0.528556
INDEX_FINGER_MCP.y	0.393216	0.664789	0.47045	0.560293
INDEX_FINGER_PIP.x	0.390594	0.513809	0.212547	0.494616
INDEX_FINGER_PIP.y	0.370394	0.537262	0.450001	0.411368
INDEX_FINGER_DIP.x	0.402981	0.506091	0.230641	0.472861
INDEX_FINGER_DIP.y	0.449733	0.445233	0.551565	0.314119
INDEX_FINGER_TIP.x	0.416766	0.501942	0.258426	0.458727
INDEX_FINGER_TIP.y	0.509701	0.365516	0.619552	0.218974
MIDDLE_FINGER_MCP.x	0.468968	0.578839	0.322361	0.597762
MIDDLE_FINGER_MCP.y	0.378663	0.63971	0.438229	0.561901
MIDDLE_FINGER_PIP.x	0.447741	0.538673	0.278005	0.622411
MIDDLE_FINGER_PIP.y	0.376232	0.524995	0.387824	0.417274

MIDDLE_FINGER_DIP.x	0.466829	0.506799	0.28388	0.64093
MIDDLE_FINGER_DIP.y	0.481974	0.436372	0.501866	0.31755
MIDDLE_FINGER_TIP.x	0.484939	0.487269	0.307072	0.659745
MIDDLE_FINGER_TIP.y	0.554076	0.362243	0.569292	0.22418
RING_FINGER_MCP.x	0.527718	0.639073	0.395642	0.651466
RING_FINGER_MCP.y	0.382022	0.651393	0.430524	0.598375
RING_FINGER_PIP.x	0.505762	0.623854	0.374539	0.684657
RING_FINGER_PIP.y	0.378085	0.652502	0.406852	0.638069
RING_FINGER_DIP.x	0.518674	0.616305	0.369548	0.665789
RING_FINGER_DIP.y	0.482942	0.768619	0.54245	0.730897
RING_FINGER_TIP.x	0.53095	0.614971	0.378459	0.647558
RING_FINGER_TIP.y	0.554198	0.842028	0.600908	0.772569
PINKY_MCP.x	0.588687	0.695562	0.466656	0.687371
PINKY_MCP.y	0.397633	0.683408	0.440557	0.652676
PINKY_PIP.x	0.566689	0.667793	0.443367	0.703334
PINKY_PIP.y	0.379503	0.702079	0.44557	0.690174
PINKY_DIP.x	0.560907	0.650071	0.43254	0.684527
PINKY_DIP.y	0.453977	0.793078	0.533916	0.755706
PINKY_TIP.x	0.562171	0.641488	0.435452	0.665548
PINKY_TIP.y	0.509735	0.856231	0.555947	0.795826

Table 3.2: Sample values from the dataset

3.2.3 Feature Extraction *(Hand keypoints)*

For our dataset, we needed to get the relative x- and y-coordinates of hand keypoints. The keypoints here are - wrist (WRIST), thumb (THUMB_CMC, THUMB_MCP, THUMB_IP, THUMB_TIP), index finger (INDEX_FINGER_MCP, INDEX_FINGER_PIP, INDEX_FINGER_DIP, INDEX_FINGER_TIP), middle finger (MIDDLE_FINGER_MCP, MIDDLE_FINGER_PIP, MIDDLE_FINGER_DIP, MIDDLE_FINGER_TIP), ring finger (RING_FINGER_MCP, RING_FINGER_PIP, RING_FINGER_DIP, RING_FINGER_TIP) and pinky (PINKY_MCP, PINKY_PIP, PINKY_DIP, PINKY_TIP). [3]

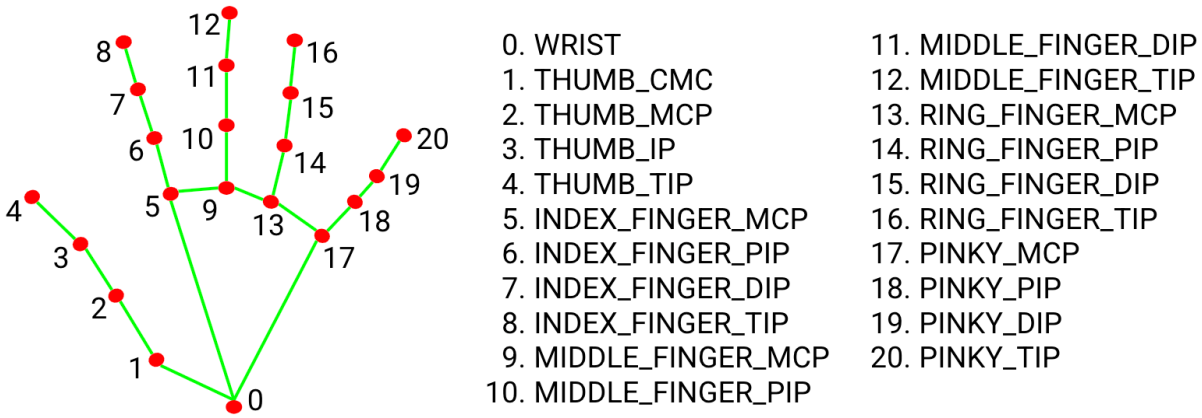


Figure 3.2: Hand Landmarks

The above figure shows the 21 hand landmarks or keypoints as we are considering them in our project. Each of the points above represent a keypoint which is a key factor in deciding the gesture.

Sample keypoint for the gesture ***a***:

```
[     0.5198443531990051,     0.6311778426170349,     0.43843284249305725,
0.5984315276145935,         0.3792191743850708,         0.49412065744400024,
0.36575400829315186,         0.3887031674385071,         0.35017073154449463,
0.3055737316608429,         0.4160507917404175,         0.39321616291999817,
0.39059412479400635,         0.37039363384246826,         0.4029809236526489,
0.44973304867744446,         0.4167657494544983,         0.5097014307975769,
0.46896839141845703,         0.37866315245628357,         0.4477413296699524,
0.376232385635376,         0.46682894229888916,         0.48197415471076965,
0.484939306974411,         0.5540761947631836,         0.5277183055877686,
0.38202211260795593,         0.5057616829872131,         0.37808459997177124,
0.5186744332313538,         0.4829420745372772,         0.5309498310089111,
0.5541979074478149,         0.5886871218681335,         0.39763331413269043,
0.5666894912719727,         0.37950295209884644,          0.560907244682312,
0.45397675037384033,         0.5621711015701294,    0.5097349286079407 ]
```

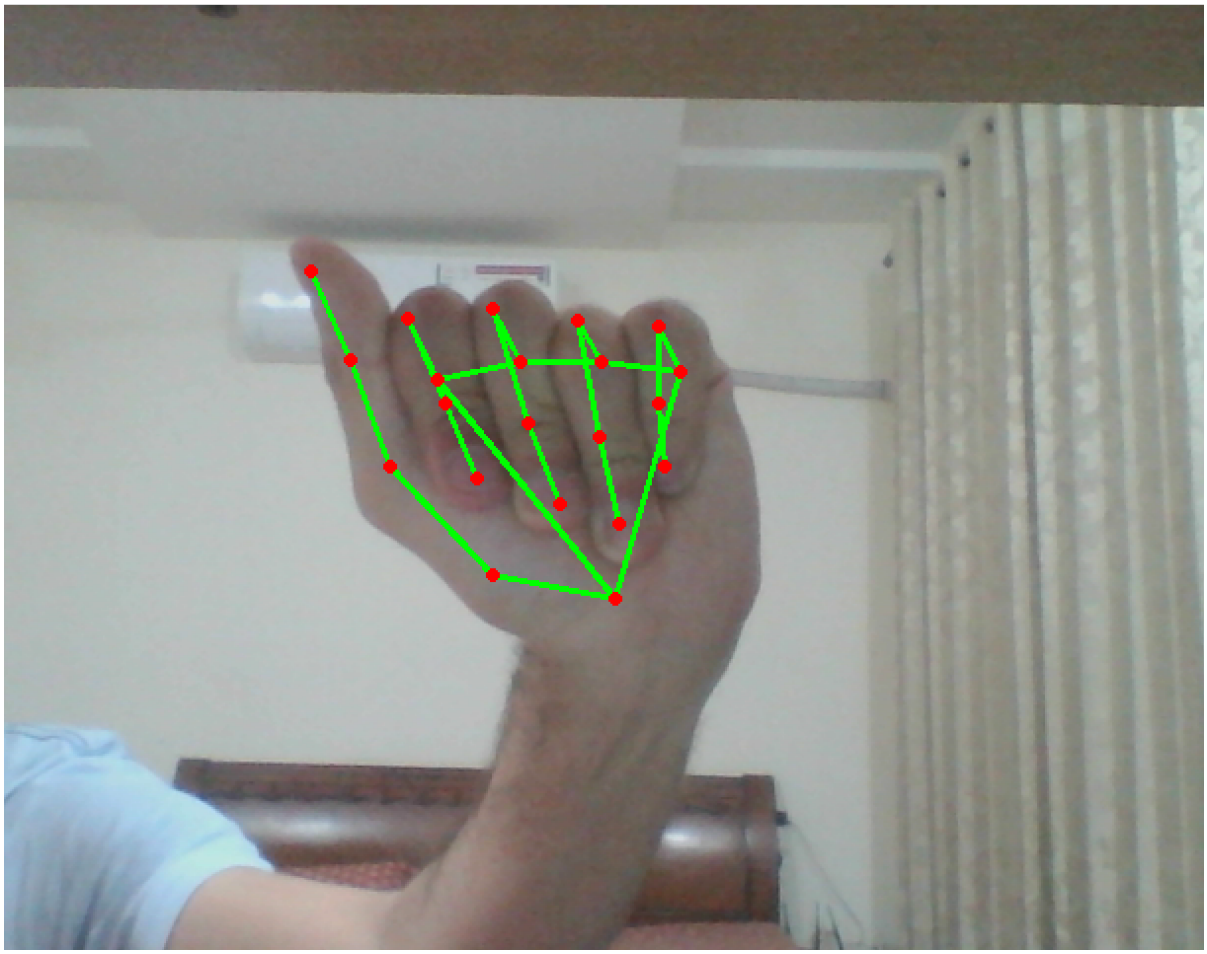


Figure 3.3: Keypoints over hand for the gesture 'a'

The following statement can be used to draw and connect the keypoints on the hand image:

```
self.MP_DRAWING.draw_landmarks(image, hand_landmarks, self.MP_HANDS.HAND_CONNECTIONS)
```

Finding the keypoints using MediaPipe is quite simple by using the code snippet as follows:

```
keypoints = [[  l[self.MP_HANDS.HandLandmark.WRIST].x,
                l[self.MP_HANDS.HandLandmark.WRIST].y,
                l[self.MP_HANDS.HandLandmark.THUMB_CMC].x,
                l[self.MP_HANDS.HandLandmark.THUMB_CMC].y,
                l[self.MP_HANDS.HandLandmark.THUMB_MCP].x,
                l[self.MP_HANDS.HandLandmark.THUMB_MCP].y,
                l[self.MP_HANDS.HandLandmark.THUMB_IP].x,
                l[self.MP_HANDS.HandLandmark.THUMB_IP].y,
                l[self.MP_HANDS.HandLandmark.THUMB_TIP].x,
                l[self.MP_HANDS.HandLandmark.THUMB_TIP].y,
                l[self.MP_HANDS.HandLandmark.INDEX_FINGER_MCP].x,
                l[self.MP_HANDS.HandLandmark.INDEX_FINGER_MCP].y,
                l[self.MP_HANDS.HandLandmark.INDEX_FINGER_PIP].x,
                l[self.MP_HANDS.HandLandmark.INDEX_FINGER_PIP].y,
                l[self.MP_HANDS.HandLandmark.INDEX_FINGER_DIP].x,
                l[self.MP_HANDS.HandLandmark.INDEX_FINGER_DIP].y,
                l[self.MP_HANDS.HandLandmark.INDEX_FINGER_TIP].x,
                l[self.MP_HANDS.HandLandmark.INDEX_FINGER_TIP].y,
                l[self.MP_HANDS.HandLandmark.MIDDLE_FINGER_MCP].x,
                l[self.MP_HANDS.HandLandmark.MIDDLE_FINGER_MCP].y,
                l[self.MP_HANDS.HandLandmark.MIDDLE_FINGER_PIP].x,
                l[self.MP_HANDS.HandLandmark.MIDDLE_FINGER_PIP].y,
                l[self.MP_HANDS.HandLandmark.MIDDLE_FINGER_DIP].x,
                l[self.MP_HANDS.HandLandmark.MIDDLE_FINGER_DIP].y,
                l[self.MP_HANDS.HandLandmark.MIDDLE_FINGER_TIP].x,
                l[self.MP_HANDS.HandLandmark.MIDDLE_FINGER_TIP].y,
                l[self.MP_HANDS.HandLandmark.RING_FINGER_MCP].x,
                l[self.MP_HANDS.HandLandmark.RING_FINGER_MCP].y,
                l[self.MP_HANDS.HandLandmark.RING_FINGER_PIP].x,
                l[self.MP_HANDS.HandLandmark.RING_FINGER_PIP].y,
                l[self.MP_HANDS.HandLandmark.RING_FINGER_DIP].x,
                l[self.MP_HANDS.HandLandmark.RING_FINGER_DIP].y,
                l[self.MP_HANDS.HandLandmark.RING_FINGER_TIP].x,
                l[self.MP_HANDS.HandLandmark.RING_FINGER_TIP].y,
                l[self.MP_HANDS.HandLandmark.PINKY_MCP].x,
                l[self.MP_HANDS.HandLandmark.PINKY_MCP].y,
                l[self.MP_HANDS.HandLandmark.PINKY_PIP].x,
                l[self.MP_HANDS.HandLandmark.PINKY_PIP].y,
                l[self.MP_HANDS.HandLandmark.PINKY_DIP].x,
                l[self.MP_HANDS.HandLandmark.PINKY_DIP].y,
                l[self.MP_HANDS.HandLandmark.PINKY_TIP].x,
                l[self.MP_HANDS.HandLandmark.PINKY_TIP].y
                ]]
```

3.2.4 k-Nearest Neighbours

k-Nearest Neighbours or simply k-NN, is a machine learning model that can be used for both classification and regression problems. The k-NN algorithm is based on the similarities between the dataset and the sample to be predicted and it puts the sample into the category that is the closest to the categories present in the dataset. It stores the entire dataset and classifies the sample data based on similarity.

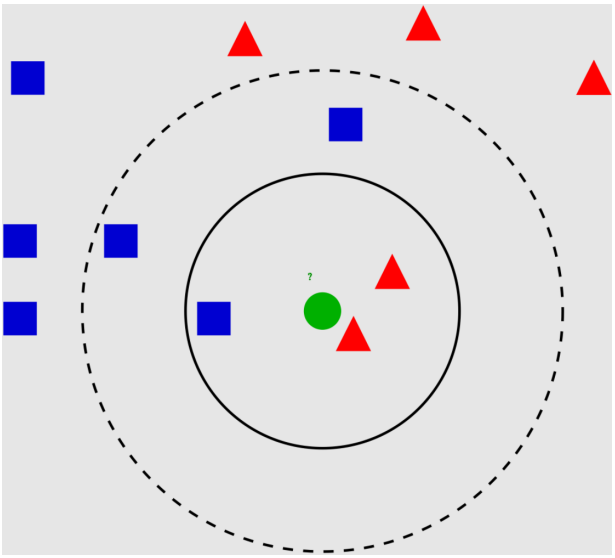


Figure 3.4: k-Nearest Neighbours example

3.2.5 Support Vector Machines

Support Vector Machines or simply SVM, is placed under the category of supervised machine learning algorithm. It is generally used for classification problems. In this model, each data item is plotted as a point in a n-D (here, n denotes the number of features in the training dataset) space and then it performs classification by identifying hyper-plane(s) that separates the two classes with the maximal distances.

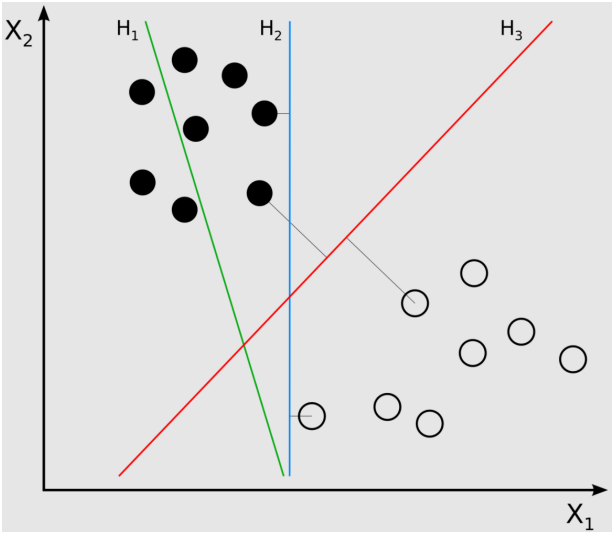


Figure 3.5: Support Vector Machine example

3.2.6 Decision Tree

A decision tree is a tree-like model that supports in making decisions. It consists of a tree with decisions and possible outcomes, etc. A decision tree model uses a decision tree to reach the conclusions about an item using a set of decisions from the decision tree.

3.3 Code and Application Screenshots

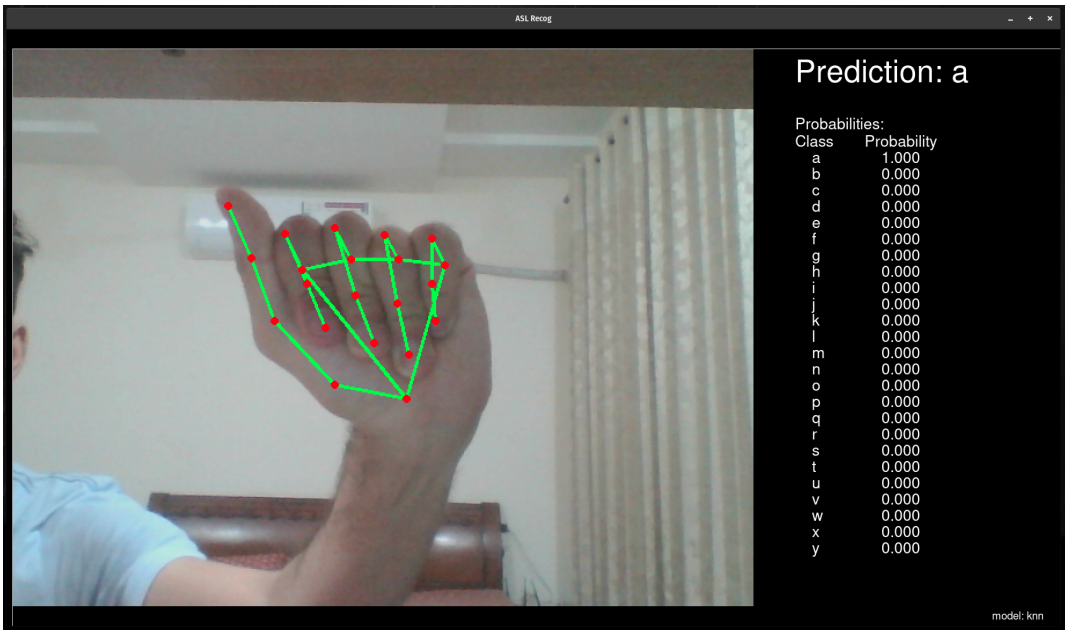


Figure 3.6: Application working on gesture 'a'

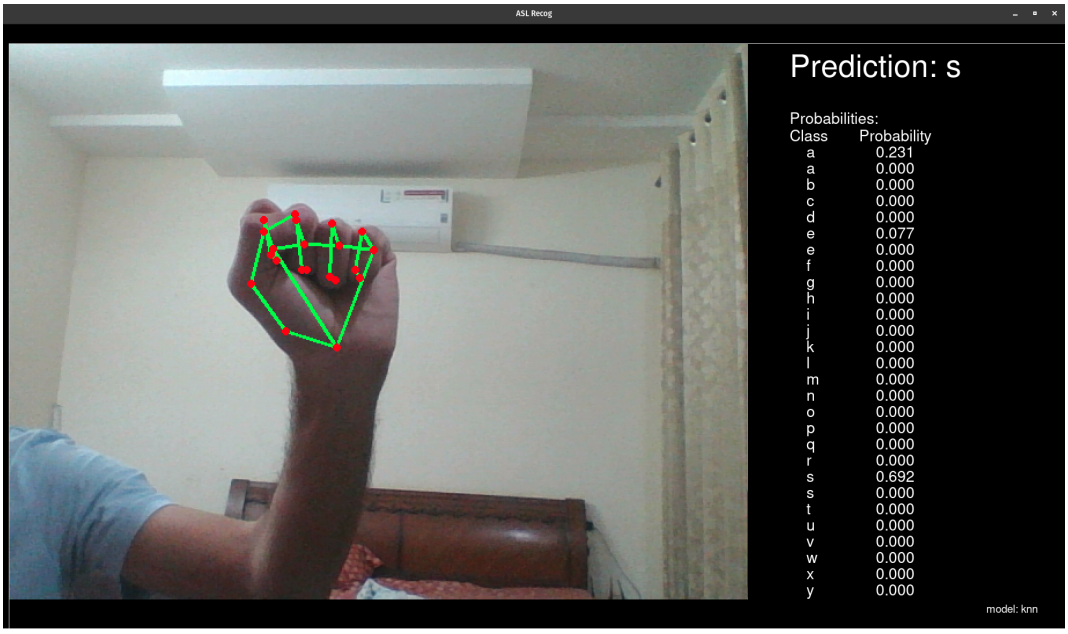


Figure 3.7: Application working on gesture 's'

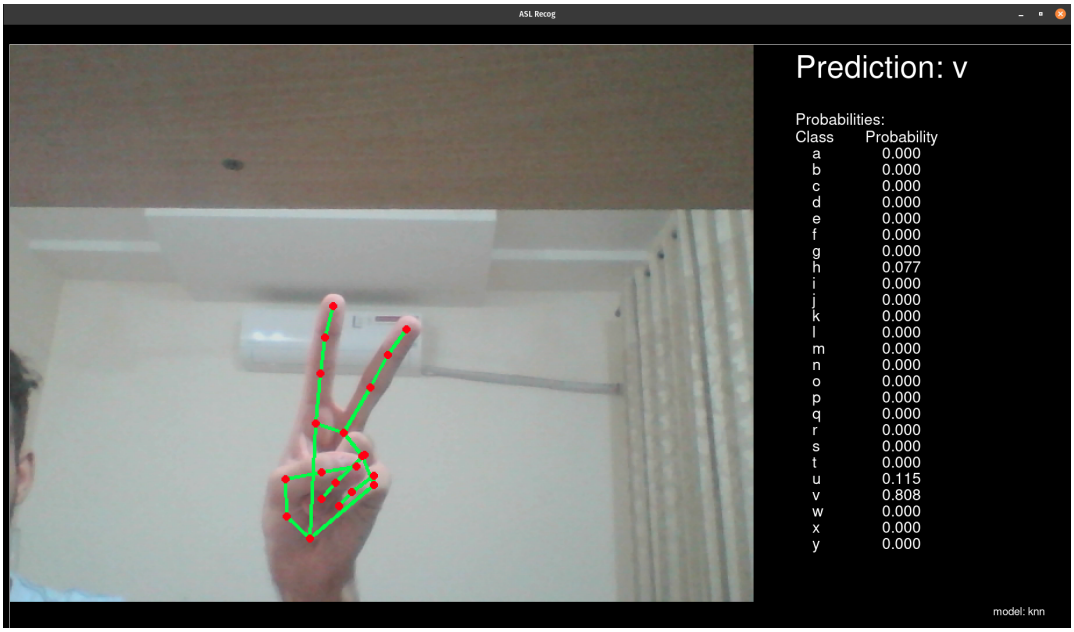


Figure 3.8: Application working on gesture 'v'

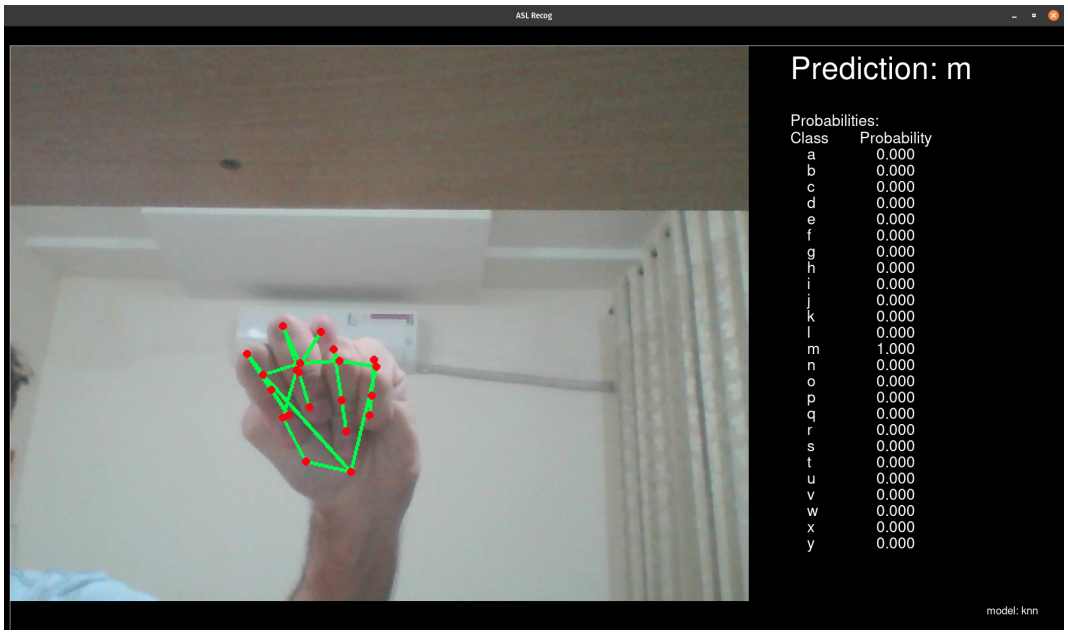


Figure 3.9: Application working on gesture 'm'

3.4 ML Pipeline

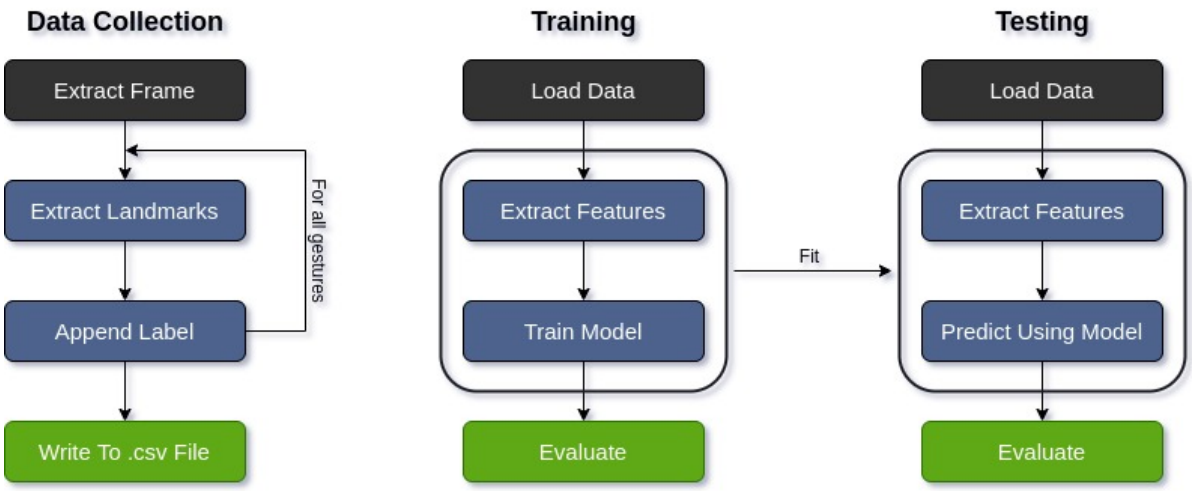


Figure 3.10: Model Pipeline

Chapter 4
PERFORMANCE ANALYSIS

4.1 Performance Analysis

The results show that the gesture recognition model is quite robust and precise for static images. However, it is not the same story in the case of video streams. The prediction on video streams is greatly affected by the illumination of the surroundings. Simply said, the models proved to be susceptible to noise (here noise refers to the objects in the background which have a texture or colour similar to the hand) in the live video stream. But, if the hand is kept steady for some time, the outputs were seen to be quite accurate. Slight hand movements were able to affect the process of gesture recognition and thus, resulting in inaccurate predictions.

Metrics are used to measure and assess the performance of the machine learning models. Some of the most common metrics used are - accuracy, precision, recall and f1-score. The performance metrics for the models used are shown as below:

4.1.1 k-Nearest Neighbours

The metrics for k-nearest neighbours model had the following values:

Accuracy: 41.84%

Precision: 0.35

Recall: 0.41

f1-score: 0.35

```
##### KNN REPORT #####
Score:
0.4184

Classification Report
              precision    recall  f1-score   support

           0       0.16      0.27      0.20        60
           1       1.00      1.00      1.00        55
           2       0.00      0.00      0.00        57
           3       0.00      0.00      0.00        49
           4       0.00      0.00      0.00        45
           5       1.00      1.00      1.00        54
           6       1.00      1.00      1.00        44
           7       1.00      1.00      1.00        54
           8       0.00      0.00      0.00        48
           9       0.00      0.00      0.00        42
          10       0.00      0.00      0.00        45
          11       1.00      1.00      1.00        48
          12       0.37      1.00      0.54        46
          13       0.00      0.00      0.00        48
          14       0.29      0.61      0.40        49
          15       0.00      0.00      0.00        53
          16       0.50      1.00      0.66        52
          17       0.00      0.00      0.00        43
          18       1.00      0.37      0.54        60
          19       0.00      0.00      0.00        56
          20       0.00      0.00      0.00        42
          21       0.26      1.00      0.41        42
          22       0.00      0.00      0.00        43
          23       0.07      0.12      0.09        57
          24       1.00      1.00      1.00        58

    accuracy                           0.42      1250
   macro avg       0.35      0.41      0.35      1250
weighted avg       0.36      0.42      0.37      1250
```

Table 4.1: k-NN evaluation report

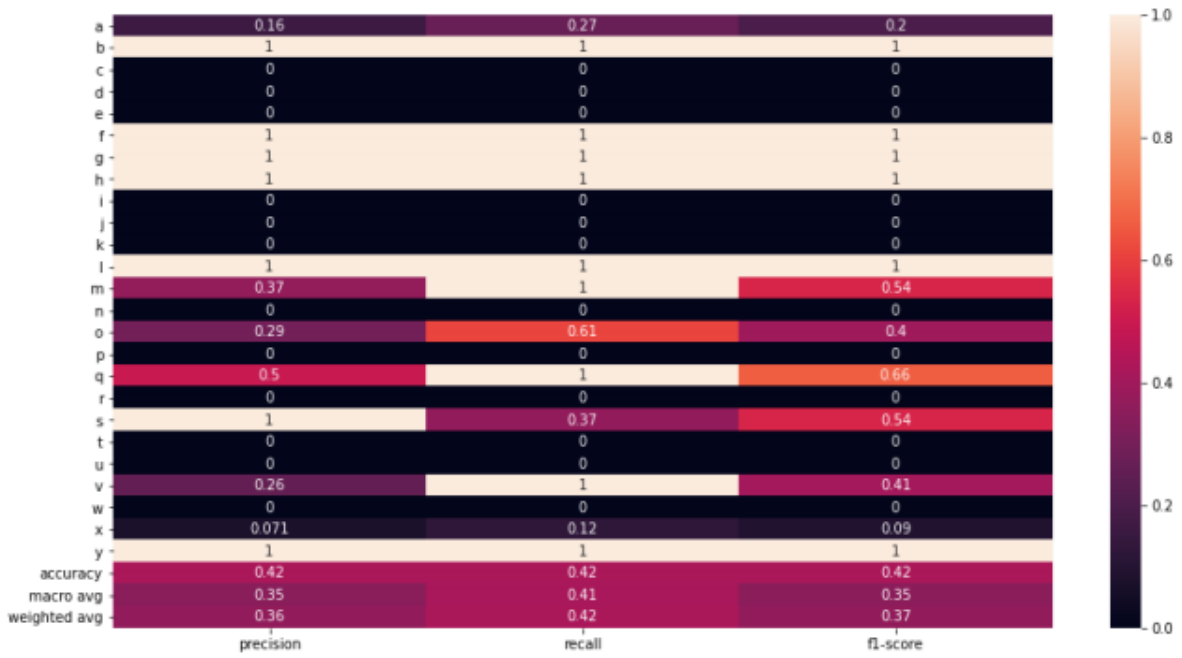


Figure 4.1: k-NN evaluation report

4.1.2 Support Vector Machine

The metrics for support vector machine model had the following values:

Accuracy: 30.08%

Precision: 0.27

Recall: 0.30

f1-score: 0.25

```
##### SVM REPORT #####
Score:
0.3008

Classification Report
              precision    recall  f1-score   support

           0       1.00      0.37      0.54        60
           1       0.00      0.00      0.00        55
           2       0.00      0.00      0.00        57
           3       0.00      0.00      0.00        49
           4       0.00      0.00      0.00        45
           5       0.00      0.00      0.00        54
           6       0.46      1.00      0.63        44
           7       0.01      0.04      0.01        54
           8       0.85      0.71      0.77        48
           9       0.00      0.00      0.00        42
          10       0.00      0.00      0.00        45
          11       1.00      1.00      1.00        48
          12       0.57      1.00      0.72        46
          13       0.00      0.00      0.00        48
          14       0.00      0.00      0.00        49
          15       0.00      0.00      0.00        53
          16       0.50      1.00      0.67        52
          17       0.00      0.00      0.00        43
          18       1.00      0.37      0.54        60
          19       0.00      0.00      0.00        56
          20       0.00      0.00      0.00        42
          21       0.19      1.00      0.32        42
          22       0.00      0.00      0.00        43
          23       0.11      0.11      0.11        57
          24       1.00      1.00      1.00        58

    accuracy                           0.30      1250
   macro avg       0.27      0.30      0.25      1250
weighted avg       0.28      0.30      0.26      1250
```

Table 4.2: SVM evaluation report

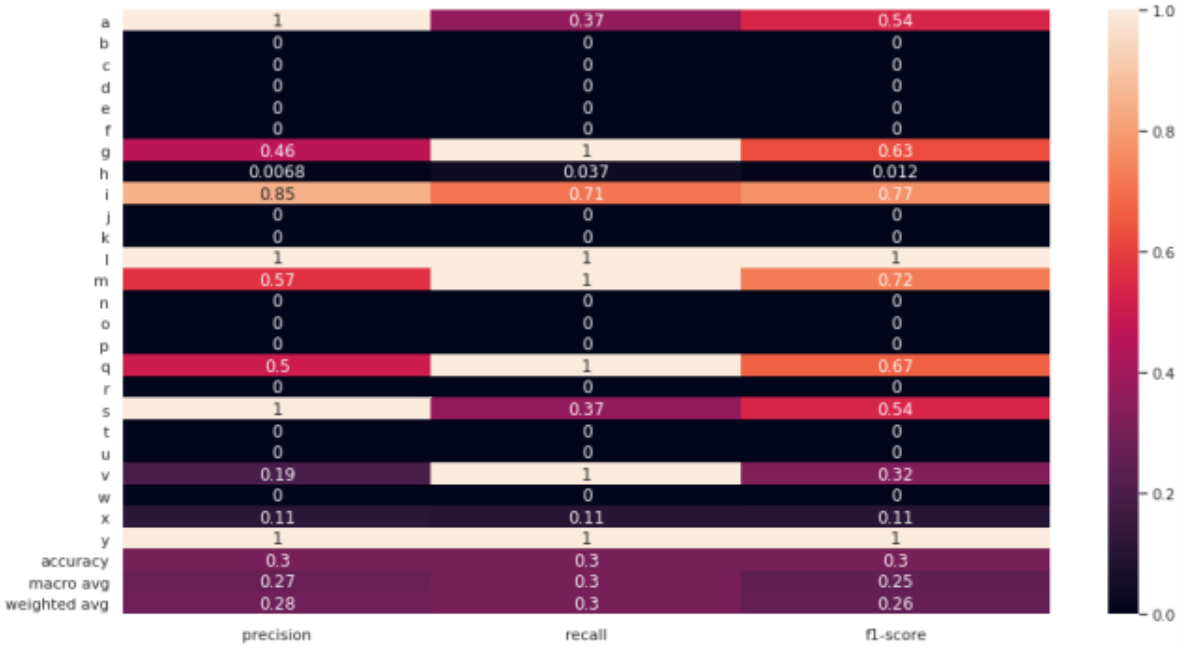


Figure 4.2: SVM evaluation report

4.1.3 Decision Tree

The metrics for decision tree model had the following values:

Accuracy: 41.84%

Precision: 0.35

Recall: 0.41

f1-score: 0.35

```
##### DT REPORT #####
Score:
0.3

Classification Report
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        60
           1       0.00      0.00      0.00        55
           2       0.00      0.00      0.00        57
           3       0.00      0.00      0.00        49
           4       0.00      0.00      0.00        45
           5       0.00      0.00      0.00        54
           6       0.22      0.93      0.36        44
           7       0.94      0.93      0.93        54
           8       0.00      0.00      0.00        48
           9       0.00      0.00      0.00        42
          10       0.00      0.00      0.00        45
          11       1.00      0.96      0.98        48
          12       0.00      0.00      0.00        46
          13       0.00      0.00      0.00        48
          14       0.17      1.00      0.29        49
          15       0.00      0.00      0.00        53
          16       0.50      1.00      0.66        52
          17       0.00      0.00      0.00        43
          18       1.00      0.38      0.55        60
          19       0.00      0.00      0.00        56
          20       0.54      1.00      0.70        42
          21       0.59      0.69      0.64        42
          22       0.00      0.00      0.00        43
          23       0.00      0.00      0.00        57
          24       0.96      0.74      0.83        58

    accuracy                           0.30      1250
   macro avg       0.24      0.31      0.24      1250
weighted avg       0.24      0.30      0.24      1250
```

Table 4.3: Decision tree evaluation report

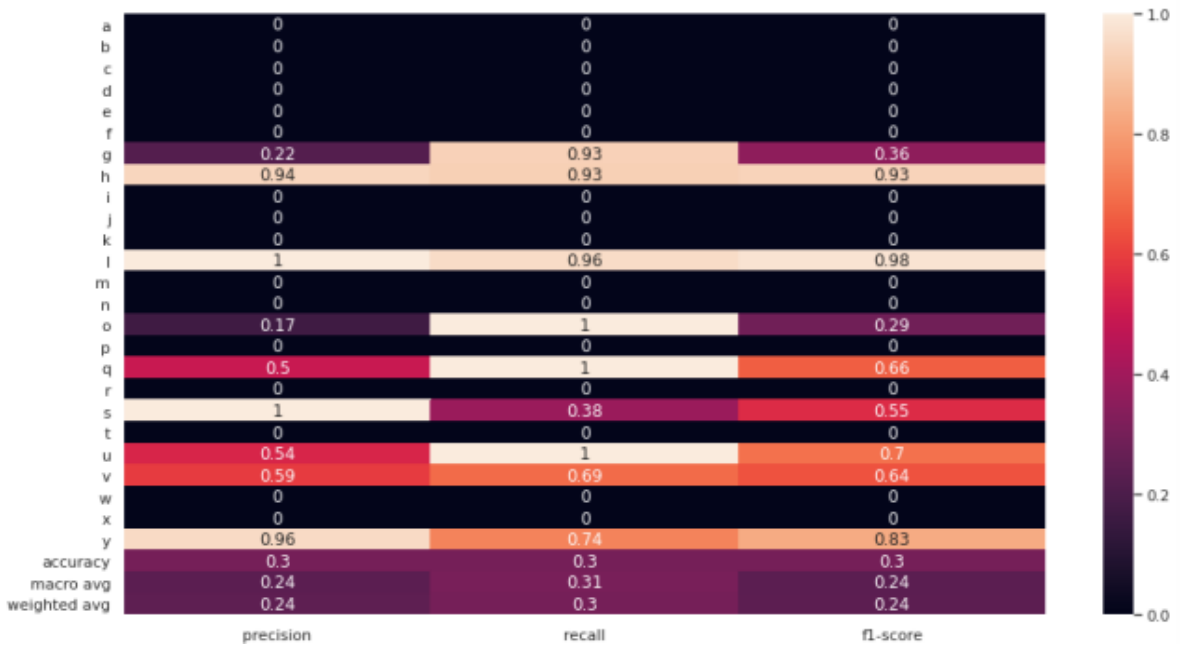


Figure 4.3: Decision tree evaluation report

4.2 Constraints

1. It is advised to use a dark background for best results.
2. Currently, only a limited number of gestures are recognized.
3. Sudden movement of hands or not making the accurate gestures resulted in inaccurate predictions.

Chapter 5
CONCLUSIONS

5.1 Future Scope

The future work that can improve our proposed work is listed as follows:

1. Understanding and applying a better algorithm to account for background noise and better separation of the foreground from the background.
2. Achieving better performance by using a customized Neural Network.
3. Comparing the results when Transfer Learning is used instead of basic Neural Networks.
4. Using just computer vision techniques such as convex hull, contour centroid, fingertip detection and distances to achieve faster results.

5.2 Applications

1. Sign Language Communication

   Our project can be used to help in communication with sign language and remove the barrier between people with speech disabilities and others.

2. Gesture Control

   It can be used to control various other products and software applications using such gestures.

   An example will be controlling your music, skipping tracks, changing volume or even playing games using gesture control.

REFERENCES

[1] Census India, Disabled Population.
https://censusindia.gov.in/census_and_you/disabled_population.aspx
[2] OpenCV Documentation, https://docs.opencv.org/master/
[3] MediaPipe Hands, https://google.github.io/mediapipe/solutions/hands.html
[4] Model Card - MediaPipe Hands,
https://drive.google.com/file/d/1yiPfkhb4hSbXJZaSq9vDmhz24XVZmxpL/preview
[5] Altman, Naomi S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression
[6] Cortes, Corinna; Vapnik, Vladimir N. (1995). Support-vector networks
[7] Shalev-Shwartz, Shai; Ben-David, Shai (2014). Decision Trees
[8] Sawant, S. (2014). Sign Language Recognition System to aid Deaf-dumb People Using PCA.
[9] I. G. Incertis (2006). Hand Gesture Recognition for Deaf People Interfacing
[10] A. S. Konwar (2014). An American Sign Language detection system using HSV color model and edge detection
[11] M. M. Islam (2017). Real time Hand Gesture Recognition using different algorithms based on American Sign Language
[12] Patil, Purushottam & Patil, Neel & Tewar, Chandru & Bhoi, Rohit & Yadav, Deepak. (2019). Deaf Communicator.
[13] Machine Learning Pipeline. https://algorithmia.com/blog/ml-pipeline

Arnav_Plag

ORIGINALITY REPORT

17%	9%	9%	11%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Jaypee University of Information Technology Student Paper	4%
2	www.ir.juit.ac.in:8080 Internet Source	2%
3	Jieun Lee. "Human hand adaptation using sweeps: generating animatable hand models", Computer Animation and Virtual Worlds, 09/2007 Publication	2%
4	ijsret.org Internet Source	2%
5	www.ijcset.com Internet Source	1%
6	A. Sharmila Konwar, B. Sagarika Borah, C. T. Tuithung. "An American Sign Language detection system using HSV color model and edge detection", 2014 International Conference on Communication and Signal Processing, 2014 Publication	1%

7	Md. Mohiminul Islam, Sarah Siddiqua, Jawata Afnan. "Real time Hand Gesture Recognition using different algorithms based on American Sign Language", 2017 IEEE International Conference on Imaging, Vision & Pattern Recognition (icIVPR), 2017 Publication	1 %
8	Submitted to Universiti Malaysia Perlis Student Paper	1 %
9	tutorsonspot.com Internet Source	1 %
10	Submitted to Universiti Malaysia Pahang Student Paper	1 %
11	I.G. Incertis, J.G. Garcia-Bermejo, E.Z. Casanova. "Hand Gesture Recognition for Deaf People Interfacing", 18th International Conference on Pattern Recognition (ICPR'06), 2006 Publication	1 %
12	www.coursehero.com Internet Source	<1 %
13	Submitted to Dr. Pillai Global Academy Student Paper	<1 %
14	myfik.unisza.edu.my Internet Source	<1 %
	www.computer.org	

15	Internet Source	<1%
16	zombiedoc.com Internet Source	<1%
17	www.ijimai.org Internet Source	<1%
18	docplayer.net Internet Source	<1%

Exclude quotes	On	Exclude matches	< 15 words
Exclude bibliography	On		