# CHAPTER 1

# INTRODUCTION

## 1.1 OVERVIEW

Our project to translate sign language into spoken language aims to improve communication between the hearing and Deaf communities. We can translate sign language motions into spoken English in real-time by employing cutting-edge computer vision and machine learning technologies. The system aims to promote accessibility and inclusivity across numerous areas, has an intuitive user interface, and can adapt to different sign language variants. Future improvements will attempt to improve bidirectional communication and broaden language support, promoting a more inclusive society for all.

## 1.2 SCOPE

- **Enhanced Communication Accessibility:** The main goal of the project is to remove communication barriers for the Deaf and Hard of Hearing community in a variety of contexts, including education, healthcare, customer service, and everyday interactions.
- **Language Expansion:** The project may be expanded to include new sign languages and dialects, which will increase its adaptability and ability to serve a wider range of users.
- **Bidirectional conversation:** In the future, improvements may include voice recognition, allowing the system to translate between spoken and sign language as well as recognize both, promoting more organic and inclusive two-way conversation.
- **Accessibility in Digital Spaces:** The scope can also include incorporating this technology into online meetings, social media, and other digital activities, making it easier for sign language users to take part.
- **Accessibility in Digital Environments:** This technology may also be incorporated into online meetings, social media, and other digital activities to facilitate participation by sign language users.

## 1.3 MOTIVATION

- **Equality and Inclusivity:** The project's main driving force is the ambition to build a more equitable and inclusive society. By giving them the same possibilities for successful communication as those without hearing impairments, it seeks to empower those with hearing loss.
- **Elimination of Communication Barriers:** The initiative aims to remove the obstacles to communication that frequently keep Deaf and Hard of Hearing people apart from the rest of society. We hope to enhance their engagement in all facets of society and to improve their quality of life by giving them a way to communicate with the hearing population.
- **Opportunities for Education:** Improving access to education for people with hearing impairments is one of the main drivers of this initiative. With the use of this technology, communication between professors and students may be facilitated, improving the effectiveness and accessibility of learning.
- **Healthcare Access:** Communication problems frequently make it difficult to access healthcare. The goal of the initiative is to improve the community's access to healthcare services for the Deaf and Hard of Hearing, enabling them to interact with doctors and receive the treatment they require.
- **Promotion of Sign Language:** The project intends to promote and maintain the cultural and linguistic value of sign languages, which are a crucial component of the identity of the Deaf community, by creating technology that can convert sign language into spoken language.

# CHAPTER 2

# LITERATURE SURWAY

| SL. NO | AUTHOR | TITLE OF THE PAPER | YEAR | JOURNAL / CONFERENCE | METHODOLOGY. |
|---|---|---|---|---|---|
| 1 | Vijay mane, tejas adsarre, Tejas dharmik, Neeraj Agrawal, Dijasmit patil, Prajwal atram | Sign language translator for speech impaired people | 2023 | International conference on nascent technologies in engineering (JCNTE 2023) | •Deep learning using CNN algorithm as well as technologies like TensorFlow, Keras to train the model. |
| 2 | Om kumar C U, K.P.K devan, Renukadevi P, Balaji V AdarshSrinivas, Krithiga R | Real time detection and conversion of gestures to text and speech to sign system (2022) | 2022 | Proceedings of the 3rd international conference on electronics and communication system (ICESC 2022) IEEE Xplore | •An artificial neural network RNN (recurrent neural network) with LSTM (recurrent neural network_ long short-term memory). |

| | | | | | |
|---|---|---|---|---|---|
| **3** | **Gokulakrishnan K, Akash C, Sagaya elvaraj A, Arockiya rayal ruffus. M, Cesario de cruz E** | **Sign language to voice translator using TensorFlow and TTS algorithm** | **2021** | **IEEE international conference on mobile networks And wireless communications (ICMNWC) (2021)** | **•CNN (convolution neural network) •TTS algorithm (text to speech)** |
| **4** | **Mr. G. Sekhar reddy, A.sahithi, P.Harsha Vardhan, P.Ushasri** | **Conversion of sign language video to text and speech (2022)** | **2022** | **International journal for research (IJRASET)** | **•Convolutional neural network (CNN) to train. •Recurrent neural network (RNN)** |
| **5** | **Ankit ojha, Ayush pandey, Shubham maurya** | **Sign language to text and speech translation In real time using convolutional network** | | **International journal of engineering research & technology** | **•CNN (convolution neural network)** |
| **6** | **S. Jothimani, S.Shruthi** | **Sign and machine learning language recognition for physically impaired individuals (2022)** | **2022** | **Proceedings of the third international conference on electronic and sustainable communication systems** | **•CNN (convolution neural network) •SVM (support vector machine)** |

| | | | | | |
|---|---|---|---|---|---|
| **7** | **Mahender reddy chilakala, Vishwa vedalia** | **A report on translating sign language to English language** | | **Proceedings of the international conference on electronics and renewable systems** | **It's a study report where researcher uses some of the algorithms which used to convert sign language into text and speech** |
| **8** | **Wanbo Li, Hang pu, ruijuan wang** | **Sign language recognition based on computer vision** | | **International conference on artificial intelligence and computer application** | **•CNN (convolution neural network).**<br>**•Long short-term memory (LSTM).** |
| **9** | **Shubham thakar, Samveg shah, bhavya shah and anant v. nimkar** | **Sign language to text conversion in real time using transfer learning** | | | **•CNN (convolution neural network) on VGG16 architecture** |

| | | | | | |
|---|---|---|---|---|---|
| **10** | **Suthagar S., K.S tamilselvan, P balakumar** | **Translation of sign language for deaf and dumb people** | | **International journal of recent technology and engineering** | **•Feature extraction process by otsu's algorithm and classification by using SVM (support vector machine)** |

## 2.1 EXISTING SYSTEM

- The existing system uses gloves and cameras to convert sign language into words that are displayed on the screen.
- The gloves should be in the hand of the speaker to recognize the words by the algorithm.
- The receiver should know the formal language to read the converted words and response.

### 2.1.1 LIMITATIONS OF EXISTING SYSTEM

- Dependency on Hardware
- Obstacle to Free-Flowing Communication
- Gloves based system
- Single letter recognition system

## 2.2 PROPOSED SYSTEM

- A platform which converts sign language into words and speech Through the help of machine learning tools.
- Allowing normal peoples to understand the sign language without any miss-leading and mid-man.
- Monitoring the way of communication.
- A platform which helps both normal people and different peoples in understanding and communicating through sign language.

### 2.2.1 ADVANTAGES OF PROPOSED SYSTEM

- Bigger word recognition
- Easy to access and use
- Accurate

### 2.2.2 The Stages Of The Proposed System

1. **Data Preprocessing:**

   •**Data Augmentation:** To increase the diversity of your training data, apply techniques like rotation, scaling, and translation.

   •**Feature Extraction:** Extract relevant features from sign language images or videos, such as key points, hand gestures, or motion features. Algorithms like Scale-Invariant Feature Transform (SIFT) or Histogram of Oriented Gradients (HOG) can be useful.

   •**Normalization:** Ensure that the data is in a consistent format and scale for accurate recognition.

   •**Noise Reduction:** Use techniques like filtering (e.g., median filtering, Gaussian soothing to remove noise from the input video or image data.

**2. Sign Language Recognition:**

•**Deep Learning Models:** Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), the transformer-based models can be used to recognize sign

•**Training**

•**Classification into classes**

**3. Text and Speech Conversion:**

- **Text Processing:** Utilize NLP techniques for text processing, including tokenization, part-of-speech tagging, and Named Entity Recognition (NER).
- **Concatenative TTS:** This approach assembles pre-recorded speech segments to generate speech. Tools like Festival and FestVox can be useful.
- **Deep Learning TTS:** Implement neural TTS models like Tacotron, WaveNet, or Transformer-TTS for more natural and expressive speech synthesis.

# CHAPTER 3
# SYSTEM DESIGN

System design is a crucial phase in the software development life cycle where the high-level architecture and structure of a software system or application are planned and documented. This phase follows the completion of requirements analysis and typically precedes the actual implementation (coding) phase. The primary goal of system design is to create a blueprint for the construction of the software, ensuring that it meets the specified requirements and is scalable, maintainable, and efficient

## 3.1 SYSTEM ARCHITECTURE

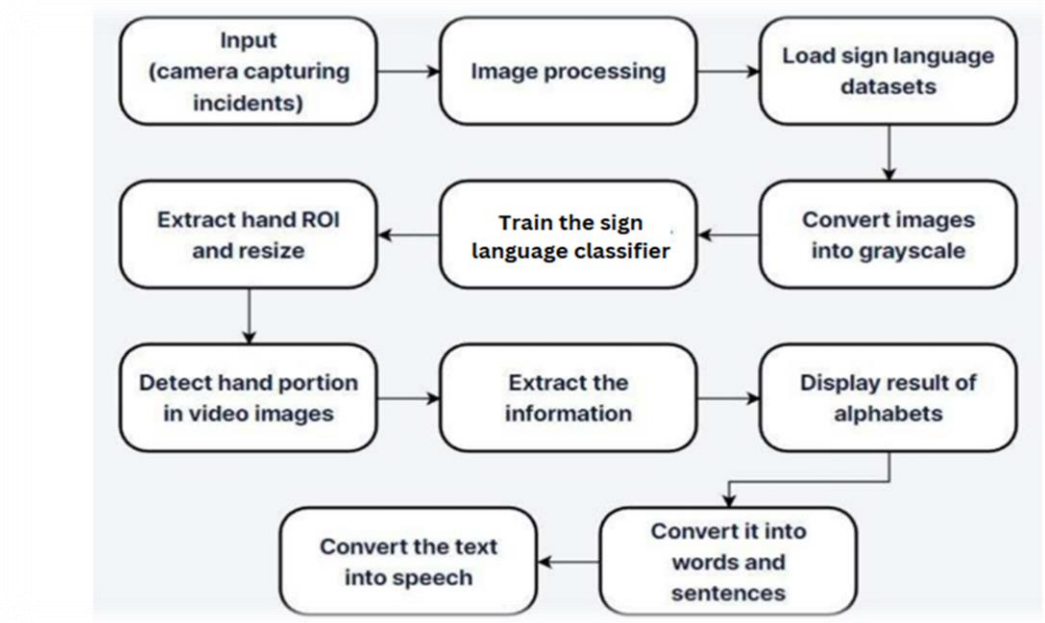


**Fig 3.1.1 system design**

**Step 1:** Data entry (collecting incidences on camera) In this stage, video frames are extracted from a video stream that was captured by a camera. The frames are then reduced in size to a standard size so that the model can process them.

**Step 2:** Resize and extract the hand ROI from the area of the frame that comprises the hands, which is known as the hand region of interest (ROI). Various techniques, including skin segmentation, hand identification, and object tracking, are used to extract this ROI. The ROI is then enlarged to a uniform size after being removed.

**Step 3:** Spot the hand in the video pictures. Identifying the hand component in the video pictures is the task at hand in this stage. A number of techniques, including backdrop removal, hand detection, and object tracking, can be used to accomplish this. Once the hand part has been identified, it is removed and sent into the classifier for sign language.

**Step 4:** Grayscale picture conversion to make the photos less noisy and more computationally efficient to analyze, they are converted to grayscale.

**Step 5:** Educate the classifier for sign language. A picture collection of various sign language motions is used to train the sign language classifier. The classifier gains the ability to distinguish between various gestures and group them according to the associated sign language letters.

**Step 6:** Retrieve the data. The information from the input photos is extracted using the trained sign language classifier. In order to do this, sign language motions in the photos must be recognized and categorized into the appropriate sign language characters.

**Step 7:** Put it into phrases and words. Then, words and phrases are created using the sign language characters that were retrieved. A statistical language model or a sign language dictionary can be used for this.

**Step 8:** Show the outcome The user is shown the final outcome of the sign language recognition system. Various techniques, such as text, audio, or animation in sign language, can be used to accomplish this.

## 3.2 ALGORITHM USED

### 1. RANDOMFOREST ALGORITHM.

Random Forest Algorithm Random Forest is a supervised learning algorithm that uses ensemble earning method for classification. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model. A Random Forest model is powerful and accurate. It usually performs great on many problems, including features with non-linear relationships. Disadvantages, however, include the following: there is no interpretability, over fitting may easily occur, we must choose the number of trees to include in the mode

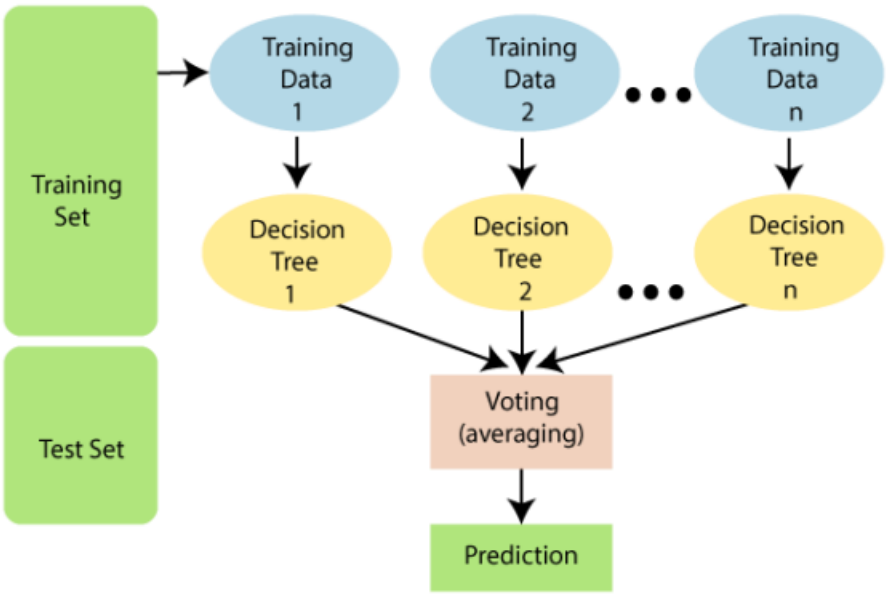


**Fig 3.2.1 random forest algorithm**

## 3.3 FLOW OF ALGORITHM

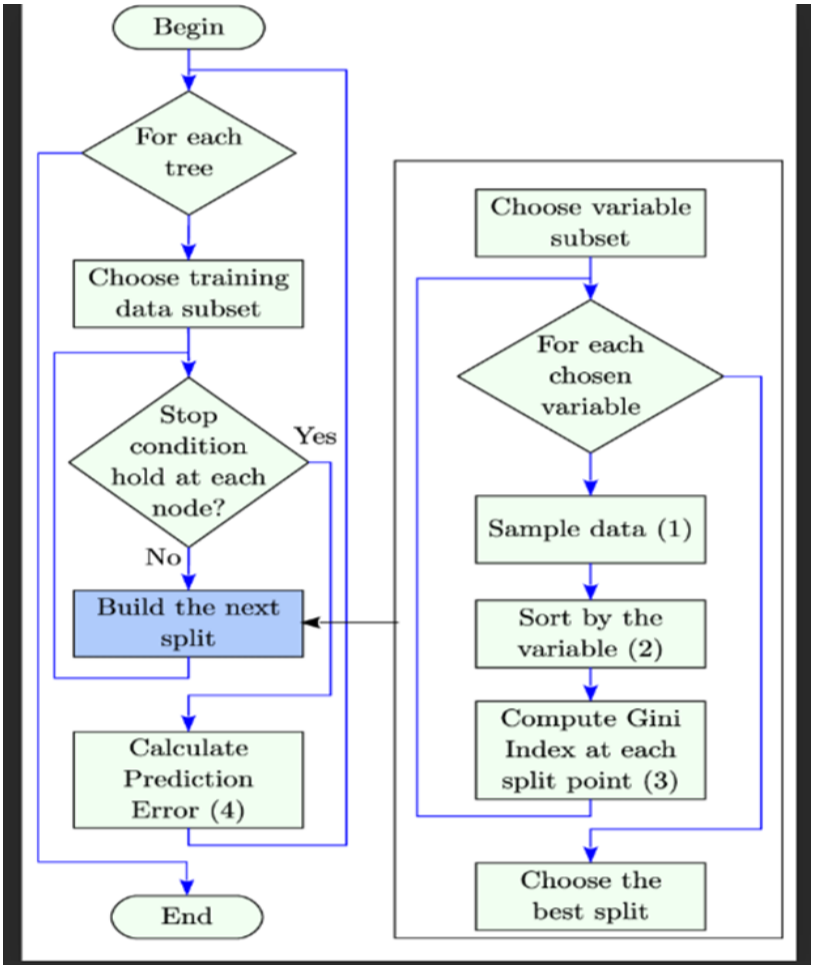


Fig 3.3.1 flow diagram of algorithm

# CHAPTER 4
# CODING

## 4.1 IMAGE COLLECTION

```
import os
import cv2
DATA_DIR = './data'
if not os.path.exists(DATA_DIR):
    os.makedirs(DATA_DIR)
number_of_classes = 60
dataset_size = 5
class_count = 0
cap = cv2.VideoCapture(0)
if not cap.isOpened():
    print("Error: Camera not opened. Check the camera index.")
    exit(1)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
cv2.namedWindow('Capturing the image', cv2.WINDOW_NORMAL)
cv2.resizeWindow('Capturing the image', 800, 600)
while class_count < number_of_classes:
    if not os.path.exists(os.path.join(DATA_DIR, str(class_count))):
        os.makedirs(os.path.join(DATA_DIR,str(class_count)))
    print(f'Collecting data for class {class_count}')
    done = False
    while True:
        ret, frame = cap.read()
```

```
        cv2.putText(frame, 'Ready? \n Press "y"-> start \n "Esc"-> end ', (75, 50), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 3, cv2.LINE_AA)
        cv2.imshow('Capturing the image', frame)
        key = cv2.waitKey(25)
        if key == ord('y'):
            break
        elif key == 27:
            cv2.destroyAllWindows()
            exit(0)
    counter = 0
    while counter < dataset_size:
        ret, frame = cap.read()
        cv2.imshow('Capturing the image', frame)
        key = cv2.waitKey(25)
        if key == ord('y'):
            break
        elif key == 27:
            cv2.destroyAllWindows()
            exit(0)
    cv2.imwrite(os.path.join(DATA_DIR, str(class_count), '
            {}.jpg'.format(counter)), frame)
        counter += 1
    class_count += 1
cap.release()
cv2.destroyAllWindows()
```

## 4.2 DATA SET CREATION:

```
import os
import pickle
import mediapipe as mp
import cv2
import matplotlib.pyplot as plt
mp_hands = mp.solutions.hands # type: ignore
mp_drawing = mp.solutions.drawing_utils # type: ignore
mp_drawing_styles = mp.solutions.drawing_styles # type: ignore
hands = mp_hands.Hands(static_image_mode=True, min_detection_confidence=0.3)
DATA_DIR = './data'
data = []
labels = []
for dir_ in os.listdir(DATA_DIR):
    for img_path in os.listdir(os.path.join(DATA_DIR, dir_)):
        data_aux = []
        x_ = []
        y_ = []

        img = cv2.imread(os.path.join(DATA_DIR, dir_, img_path))
        img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        results = hands.process(img_rgb)
        if results.multi_hand_landmarks:
            for hand_landmarks in results.multi_hand_landmarks:
                for i in range(len(hand_landmarks.landmark)):
                    x = hand_landmarks.landmark[i].x
                    y = hand_landmarks.landmark[i].y
                    x_.append(x)
                    y_.append(y)
```

```
            for i in range(len(hand_landmarks.landmark)):
                x = hand_landmarks.landmark[i].x
                y = hand_landmarks.landmark[i].y
                data_aux.append(x - min(x_))
                data_aux.append(y - min(y_))
        data.append(data_aux)
        labels.append(dir_)
f = open('data.pickle', 'wb')
pickle.dump({'data': data, 'labels': labels}, f)
f.close()
```

## 4.3 TRAIN THE ALGORITHM:

```
from sklearn.feature_extraction.text import TfidfVectorizer
import pickle
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
data_dict = pickle.load(open('./data.pickle', 'rb'))
data = data_dict['data']
labels = data_dict['labels']
# Ensure all elements in 'data' are treated as strings
data = [str(doc) if not isinstance(doc, str) else doc for doc in data]
# Create a TfidfVectorizer and transform your data
vectorizer = TfidfVectorizer(lowercase=True)
X = vectorizer.fit_transform(data)
# Split the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(X, labels, test_size=0.2, shuffle=True,
stratify=labels)
model = RandomForestClassifier()
model.fit(x_train, y_train)
y_predict = model.predict(x_test)
score = accuracy_score(y_predict, y_test)
print(f"{score * 100}% of samples were classified correctly!")
f = open('model.p', 'wb')
pickle.dump({'model': model, 'vectorizer': vectorizer}, f)
f.close()
```

## 4.4 INFERENCE_CLASSIFIER

```
import time
import pickle
import cv2
import mediapipe as mp
import numpy as np
import pyttsx3
import threading
# Load the model
model_dict = pickle.load(open('./model.p', 'rb'))
model = model_dict['model']
#mp_hands = mp.solutions.hands # type: ignore
mp_drawing = mp.solutions.drawing_utils # type: ignore
mp_drawing_styles = mp.solutions.drawing_styles # type: ignore
hands = mp_hands.Hands(static_image_mode=True, min_detection_confidence=0.3)
# Define labels for recognition
labels_dict = {
    0: 'A', 1: 'B', 2: 'C', 3: 'D', 4: 'E', 5: 'F', 6: 'G',
    7: 'H', 8: 'I', 9: 'J', 10: 'K', 11: 'L', 12: 'M', 13: 'N',
    14: 'O', 15: 'P', 16: 'Q', 17: 'R', 18: 'S', 19: 'T', 20: 'U',
    21: 'V', 22: 'W', 23: 'X', 24: 'Y', 25: 'Z',
    26: '0', 27: '1', 28: '2', 29: '3', 30: '4',
    31: '5', 32: '6', 33: '7', 34: '8', 35: '9',
    36: ' i love you'
}
cap = cv2.VideoCapture(0)
# Create a text-to-speech engine
engine = pyttsx3.init()
```

```
# Initialize variables for character recognition
recognized_characters = []  # Initialize a list to store characters
current_character = ''
last_recognition_time = time.time()
max_character_duration = 1.0
word = ''
local_dictionary = {'a': 'a', 'b': 'b', 'c': 'c', 'd': 'd', 'e': 'e', 'f': 'f', 'g': 'g', 'h': 'h', 'i': 'i', 'j': 'j', 'k': 'k', 'l': 'l', 'm': 'm', 'n': 'n', 'o': 'o', 'p': 'p', 'q': 'q', 'r': 'r', 's': 's', 't': 't', 'u': 'u', 'v': 'v', 'w': 'w', 'x': 'x', 'y': 'y', 'z': 'z', '1': '1', '2': '2', '3': '3', '4': '4', '5': '5', '6': '6', '7': '7', '8': '8', '9': '9', 'i love you':'i love you'}
    # Add more words and meanings as needed
def speak_recognized_word(word):
    meaning = local_dictionary.get(word.lower())
    if meaning:
        engine.say(word)
        engine.say(meaning)
        engine.runAndWait()
    else:
        engine.say(f"No word found for '{word}'")
        engine.runAndWait()
while True:
        current_time = time.time()
            time_difference = current_time - last_recognition_time
        ret, frame = cap.read()
    if not ret:
        print("Error: Camera not found or could not be opened.")
        break
    H, W, _ = frame.shape
    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    results = hands.process(frame_rgb)
    if results.multi_hand_landmarks:
```

```
        for hand_landmarks in results.multi_hand_landmarks:
            data_aux = []
            x_ = []
            y_ = []
            for i in range(len(hand_landmarks.landmark)):
                x = hand_landmarks.landmark[i].x
                y = hand_landmarks.landmark[i].y
                x_.append(x)
                y_.append(y)
            for i in range(len(hand_landmarks.landmark)):
                x = hand_landmarks.landmark[i].x
                y = hand_landmarks.landmark[i].y
                data_aux.append(x - min(x_))
                data_aux.append(y - min(y_))
            x1 = int(min(x_) * W) - 10
            y1 = int(min(y_) * H) - 10
            x2 = int(max(x_) * W) - 10
            y2 = int(max(y_) * H) - 10
            prediction = model.predict(np.asarray(data_aux).reshape(1, -1))
            predicted_character = labels_dict[int(prediction[0])]                print(f" predicetd
char {predicted_character}")                current_time = time.time()
            time_difference = current_time - last_recognition_time
            if time_difference >= max_character_duration:
                if current_character:
                    recognized_characters.append(current_character)
                    current_character = ''
                                print(f" current char {current_character}")
            if predicted_character != ' ':
                current_character = predicted_character
```

```
            last_recognition_time = current_time
        cv2.putText(frame, current_character, (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, 1.3, (0, 0, 0), 3, cv2.LINE_AA)
        if time_difference >= 2.0:
            if recognized_characters:
                print(f" list  {recognized_characters}")
                full_word = ''.join(recognized_characters)                print(f" full  Word {full_word}")
                recognized_characters = []
                if full_word:
                    threading.Thread(target=speak_recognized_word, args=(full_word,)).start()
    key = cv2.waitKey(2)
    if key == 27:  # Press Esc key to exit
        break
    cv2.imshow('Sign Language Recognition', frame)
cap.release()
cv2.destroyAllWindows()
```

## 4.5 SNAPSHOTS


Fig 4.5.1 start page for image collection

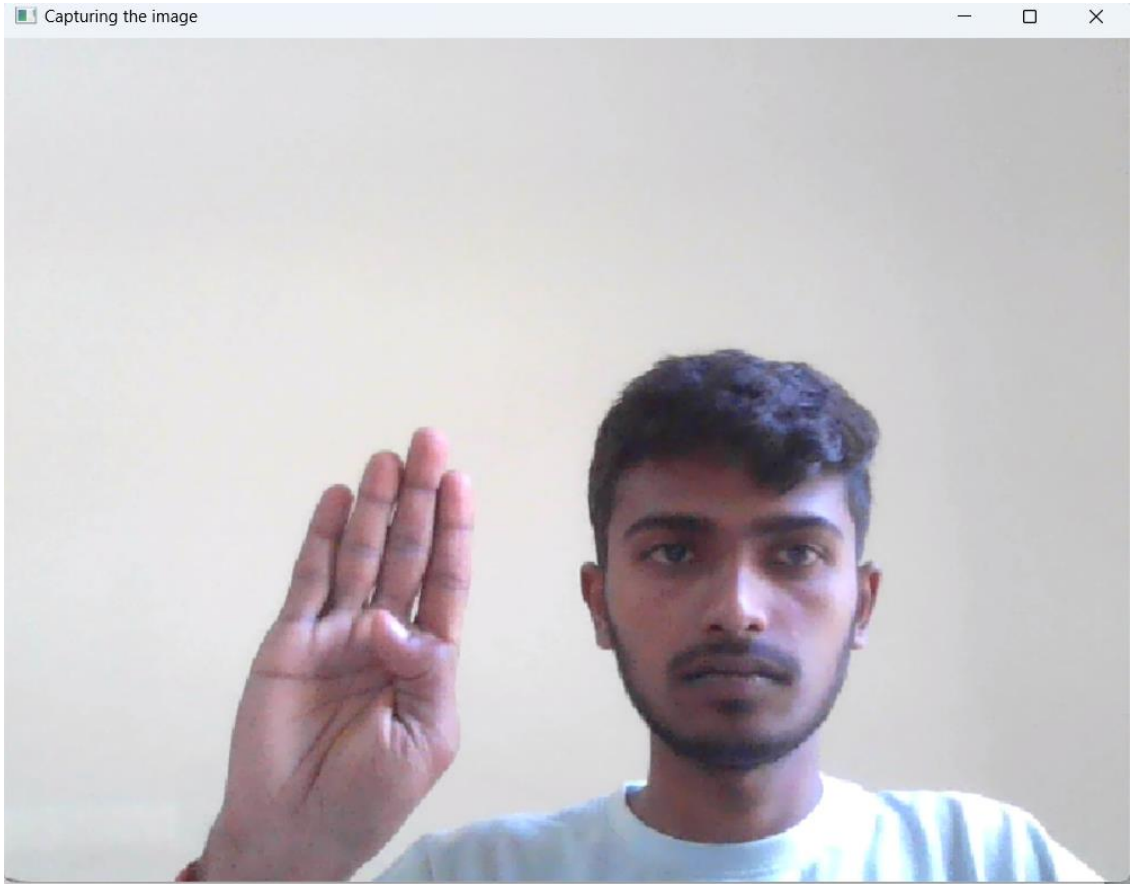


Fig 4.5.2 data collection

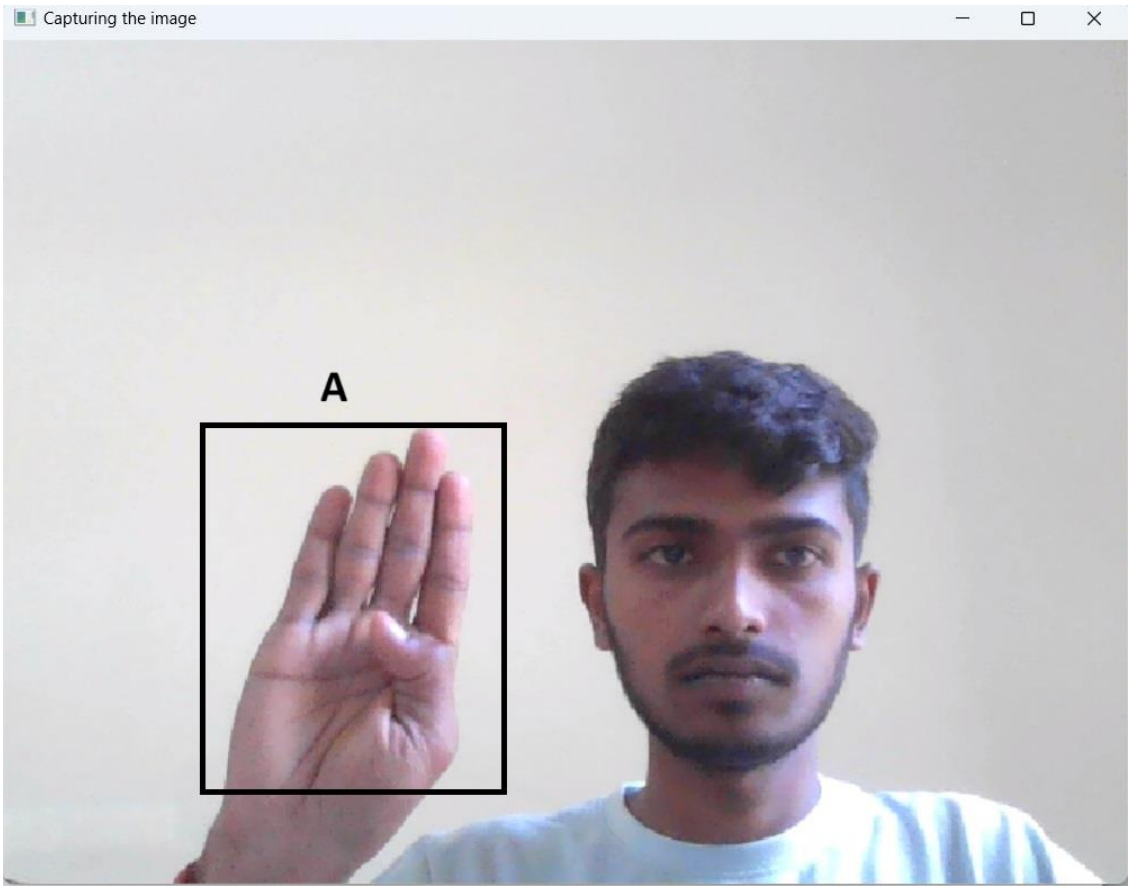


Fig 4.5.3 classification

## 4.6 RESULT AND ANALYSIS

This project harness the power of Random Forest Algorithm for the classification of the signs into text and then it converts this text into the speech by using real time hand detection. By doing this it is easy to communicate with the dumb and deaf people for the normal people and visa-versa. This minimize the communication gap between peoples and maximize the overall improvements of communication.

# CHAPTER 5

# SOFTWARE TESTING

Software testing is a crucial phase in the software development life cycle aimed at identifying and rectifying defects or issues in a software application. It involves the systematic and controlled execution of the software to evaluate its functionality and quality. The primary goals of software testing are to ensure that the software meets its requirements, performs as expected, and is free from critical defects.

**Functional Testing**

Verify that the models built for crop prediction are correctly implemented. Test the system with various user inputs representing different input parameters to ensure it produces accurate predictions. Ensure that the system correctly handles different combinations values of N,P, K and other relevant factors. Check for accurate model selection and prediction based on the user's input data.

**Non-Functional Testing Performance Testing:**

Evaluate how the system handles large volumes of user queries and whether it responds within an acceptable time frame.

**Security Testing:**

Assess the system's security measures to protect sensitive data. Verify that it is protected against common security threats.

**Usability Testing:**

Involve users or stakeholders to assess the system's user-friendliness and ensure it provides a good user experience.

**Boundary Testing:**

Test the system with extreme boundary values for N,P,K and other input parameters to confirm that it behaves correctly at the limits of its range

**Error Handling and Validation**

Test the system's error-handling capabilities, ensuring that it provides meaningful error messages for invalid or incomplete inputs.

**Ethical Considerations**

Check that the system adheres to ethical considerations in handling data, including privacy and fairness.

**Model Accuracy**

Validate the accuracy of the regression models by comparing their predictions with expected results using a set of test cases.

**Load and Stress Testing**

Assess the system's performance under a heavy load of simultaneous user requests. Verify that it maintains accuracy and responsiveness.

**Regression Testing**

Perform regression testing after any code changes or updates to ensure that the new modifications do not introduce new issues or negatively impact the model's accuracy

# Functional test cases

| Test case no | Positive scenario | Required input | Expected output | Actual output | Test pass/fail |
|---|---|---|---|---|---|
| 1 | Sign recognized and speech | Hand signal | Text identification, | Text identified | pass |
| 2 | Sign recognized and speech | Hand signal | Text identification and speech | Text identified and speech | pass |

| Test case no | Negative scenario | Required input | Expected output | Actual output | Test pass/fail |
|---|---|---|---|---|---|
| 1 | Sign doesn't recognized and no speech | Hand signal | Error no hand recognized | Error no hand recognized | pass |
| 2 | Sign doesn't recognized and no speech | Hand signal | Error no word matched | Error no word matched | pass |

# CHAPTER 6

# CONCLUSION

The project, titled **"SIGN LANGUAGE TRASLATION INTO SPEECH FOR IMPAIRED PEOPLE"** focuses on those who have speech and hearing problems, our initiative has set the groundwork for a sign language translation system that can considerably improve their quality of life. The system has the ability to reduce communication obstacles, empower people with impairments, and encourage a stronger sense of empathy and understanding among all societal members. By using the model machine learning algorithms we made this possible, thus eliminating the middle man between two peoples, and hence it increases the privacy, security for the peoples who share things with the peoples who speech and hearing problem.

# CHAPTER 7

# FUTURE ENHANCEMENT

Future work on this subject may include increasing the speed and accuracy of sign language identification, broadening the languages and sign dialects it can comprehend, and making the technology more approachable through wearable technologies and mobile apps. We can introduce a cloud storage based a web system or application for better, easy, fast, secure and lite system so that everyone can use this facility and live happily. And also till now it is recognizing the sign and convert it into the speech and text, in future we can develop this to form the words by using the characters recognized by the classifier and can give the formatted word and its meaning.

# CHAPTER 8

# REFERENCES


1. Vijay Mane, Tejas Adsare, Tejas Dharmik, Neeraj Agrawal, Dijasmit Patil, Prajwal Atram "**Sign Language Translator for Speech Impaired People**" 2023 International Conference on Nascent Technologies in Engineering (ICNTE 2023)IEEE | DOI: 10.1109/ICNTE56631.2023.10146708

2. Om Kumar C.U, K.P.K.Devan, Renukadevi .P, Balaji V, Adarsh Srinivas, Krithiga .R, " **" Real Time Detection and Conversion of Gestures to Text and Speech to Sign"** System2022 3rd International Conference on Electronics and Sustainable Communication Systems (ICESC) | 978-1-6654-7971-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICESC54411.2022.9885562

3. Gokulakrishnan. K, Akkash. C, Sagaya Selvaraj. A, Arockiya Rayal Ruffus. M, Cesario De Cruz. E, **"Sign Language to Voice Translator Using Tensorflow and TTS Algorithm"** 2021 IEEE International Conference on Mobile Networks and Wireless Communications (ICMNWC) | 978-1-6654-3883-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICMNWC52512.2021.9688545.

4. Mr. G. Sekhar Reddy, A. Sahithi , P. Harsha Vardhan, P. Ushasri, **"Conversion of Sign Language Video to Text and Speech"** International Journal for Research in Applied Science & Engineering Technology (IJRASET) ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538 Volume 10 Issue V May 2022- Available at www.ijraset.com DOI:https://doi.org/10.22214/ijraset.2022.42078

5. Ankit Ojha, Ayush Pandey, Shubham Maurya, Abhishek Thakur, Dr. Dayananda P, **"Sign Language to Text and Speech Translation in Real Time Using Convolutional Neural Network"** International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181 Published by, www.ijert.org NCAIT - 2020 Conference Proceedings.

6. S. Jothimani, S. Shruthi, E.D. Tharzanya, S. Hemalatha **"Sign and Machine Language Recognition for Physically Impaired Individuals"** 2022 3rd International Conference on Electronics and Sustainable Communication Systems (ICESC) | 978-1-6654-7971-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICESC54411.2022.9885433 Proceedings of the Third International Conference on Electronics and Sustainable Communication Systems (ICESC 2022), IEEE Xplore Part Number: CFP22V66-ART; ISBN: 978-1-6654-7971-4

7. Mahender Reddy Chilukala, Vishwa Vadalia **" A Report on Translating Sign Language English Language"** Proceedings of the International Conference on Electronics and Renewable Systems (ICEARS 2022) IEEE Xplore Part Number: CFP22AV8-ART; ISBN: 978-1-6654-8425-1 2022 International Conference on Electronics and Renewable Systems (ICEARS) | 978-1-6654-8425-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICEARS53579.2022.9751846

8. Wanbo Li, Hang Pu, Ruijuan Wang* **"Sign Language Recognition Based on Computer Vision"** 2021 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA) 2021 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA) | 978-1-6654-1867-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICAICA52286.2021.9498024

9. Shubham Thakar, Samveg Shah, Bhavya Shah and Anant V. Nimkar **"Sign Language to Text Conversion in Real Time using Transfer Learning"** arXiv:2211.14446v2 [cs.CV] 7 Dec 2022

10. Suthagar S., K. S. Tamilselvan, P. Balakumar, B. Rajalakshmi, C. Roshini **"Translation of Sign Language for Deaf and Dumb People"** International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-8 Issue-5, January 2020 DOI:10.35940/ijrte.E6555.018520 Journal Website: www.ijrte.org