

# **Hand Gesture Recognition**

A major project report submitted in partial fulfillment of the  
requirement for the award of degree of

**Bachelor of Technology**

in

**Computer Science & Engineering / Information Technology**

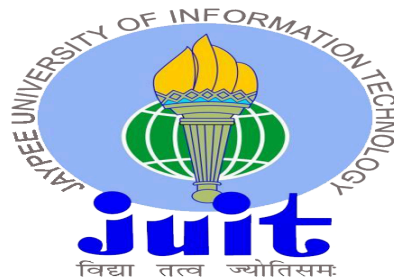
*Submitted by*

**Aditya (201216)**

**Sanidhya Kapoor (201363)**

*Under the guidance & supervision of*

**Dr. Amit Kumar**



**Department of Computer Science & Engineering and  
Information Technology**

**Jaypee University of Information Technology,**

**Waknaghat, Solan - 173234 (India)**

# CERTIFICATE

I hereby declare that the work presented in this report entitled **Hand Gesture Recognition** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering/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 August 2023 to May 2024 under the supervision of Mr. Amit Kumar (Assistant Professor, Department of CSE).

I also authenticate that I have carried out the above mentioned project work under the proficiency stream **Machine Learning**.

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

Aditya 201216

Sanidhya Kapoor 201363

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

Assistant Professor

Department of Computer Science and Engineering

Dated:

# Candidate's Declaration

I hereby declare that the work presented in this report entitled '**Hand Gesture Recognition**' in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science & Engineering / 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 August 2023 to May 2024 under the supervision of **Mr. Amit Kumar** (Assistant Professor, Department of Computer Science & Engineering and Information Technology).

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

(Student Signature with Date)

Student Name: Aditya

Roll No.: 201216

(Student Signature with Date)

Student Name: Sanidhya Kapoor

Roll No.: 201363

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

(Supervisor Signature with Date)

Supervisor Name: Mr. Amit Kumar

Designation: Assistant Professor

Department: Computer Science & Engineering

Dated:

# Acknowledgement

Firstly, I express my heartiest thanks and gratefulness to almighty God for his divine blessing making it possible to complete the project work successfully.

I am really grateful and wish my profound indebtedness to my supervisor **Mr. Amit Kumar, Assistant Professor (SG)**, Department of CSE, Jaypee University of Information Technology, Waknaghat. His deep knowledge & keen interest in the field of "**Machine Learning**" helped us to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

I would like to express my heartiest gratitude to **Mr. Amit Kumar** , Department of CSE, for his kind help to finish my project.

I would also generously welcome each one of those individuals who have helped me straightforwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patience of my parents.

Aditya 201216

Sanidhya Kapoor 201363

# TABLE OF CONTENTS

<b>TITLE NO.</b>	<b>PAGE</b>
CERTIFICATE	i
DECLARATION	ii
ACKNOWLEDGEMENT	iii
TABLE OF CONTENTS	iv
LIST OF TABLES	v
LIST OF FIGURES	vi
LIST OF ABBREVIATIONS	vii
ABSTRACT	viii
CHAPTER1:INTRODUCTION	1-5
CHAPTER2:LITERATURE SURVEY	6-20
CHAPTER3:SYSTEM DEVELOPMENT	21-47
CHAPTER4:TESTING	48-49
CHAPTER5:RESULT AND EVALUATION	50-57
CHAPTER6:CONCLUSION AND FUTURE SCOPE	58-60
REFERENCES	

# LIST OF TABLES

<b>Table-1</b>	<b>List of References</b>
<b>Table-2</b>	<b>Comparisons</b>

# LIST OF FIGURES

Fig1MediapipeImport	22
Fig 2 Basic flow of the project using simple keras model	23
Fig3 Detailed flow of the project using LSTM	24
Fig4 Detailed flow of the project using CNN	24
Fig5 Data Collection	26
Fig6 Hand detected	27
Fig7 Hand Cropped	27
Fig8 Architecture of the Hand Gesture Recognition System	28
Fig9 Images for data collected on different signs	29
Fig10 Image for showing different landmarks using mediapipe	29
Fig11 Hand landmarks	31
Fig12 Pooling Matrix	32
Fig13-16CollectionCode	36-37
Fig17-20TestCode	38-39
Fig21-25LSTMCode	40-43
Fig 26-29 CNN Code	43-45
Fig30 Confusion Matrix	49
Fig 31-35 Results and Limitations	50-53
Fig36 .npy file for 30 frames	53
Fig37 Input given after running the model.	54
Fig38&39 CNN input and output	54-55

# LIST OF ABBREVIATIONS

<b>BGR</b>	Blue Green Red
<b>CV</b>	Computer Vision
<b>CNN</b>	Convolutional Neural Network
<b>HGR</b>	Hand Gesture Recognition
<b>KNN</b>	K-Nearest Neighbour
<b>LSTM</b>	Long Short Term Memory
<b>RGB</b>	Red Green Blue
<b>RNN</b>	Recurrent Neural Network
<b>SVM</b>	Support Vector Machine



# ABSTRACT

It is necessary for humans to have a way of communicating with one another. People who are "specially abled," have speech or hearing impairments, are "mute," or are "deaf," always depend on visual communication. Individuals without visual or auditory impairments could find it challenging to communicate with those who do.

The development of a system that can convert hand gestures into text is necessary to enable two-way communication between the general public and individuals with impairments.

Finding interpreters is extremely challenging because most individuals do not know sign language, despite the fact that sign language is one of the most natural and ancient modes of communication. Given this, we have created a neural network-based fingerspelling method for American sign language that operates in real-time.

Deep learning techniques can help lower obstacles to communication. The primary phases in system design are gesture acquisition, tracking, segmentation, feature extraction, and gesture recognition. The dynamic dataset of common gestures that the author produced is used to train the sign language recognition system. The gesture is accurately recognised by the trained model, which then projects it as text onto the screen.

# CHAPTER 1 : INTRODUCTION

## 1.1 INTRODUCTION

The power to converse with technology using natural gestures in the constantly adapting arena of human-computer interaction is a centerpiece for ingenuity. Computer vision sub-field of hand gestures detection has become popular due to its role in eradicating boundaries between humans and computers. The transformational nature intended to decode human hand trajectories and configurations in a manner that is concurrent with them as a natural interface supporting broad spectrum applications.

Hand gesture detection is centrally based on modern technologies such as depth-sensing cameras and complex machine learning algorithms. This comprises various elements that work jointly, making a comprehensible rendering into meaningful interaction with the three-dimensional features of hand gestures. However, it is difficult to overcome challenges like different gestures, hand shape and also high-speed performance. Every day, researchers and designers are working on new approaches to machine learning models and CV techniques to develop robust systems that can manage this complexity.

Hand gesture detection is significant because it has the power of transforming the way we experience things within different sectors. Users are able to enter into an alternative reality when gaming and manoeuvre events through natural hand gestures. Interactive virtual and augmented reality apps work because they make people feel like they can interact with digital elements intuitively.

In addition, it has great potential for use in healthcare rehabilitation and sign language interpretation that improves communication access.

Going forward the trend will be to take such systems that combine wearable devices with hand gesture recognition downward the size line in order to increase the portability of personal computing. Hand gesture detection and recognition are moving towards edge computing based processing data at the device edge for reduced latency and investigation of

multi-modal interaction combinations based gestures accompanied by voice and face expression.

The central topic of this project report deals with the complexities associated with hand gesture detection and recognition, which are considered an imminent phenomenon contributing to modifying conventional paradigms governing people's interrelations with computers through technological innovations.

Hand Gesture Detection: A Comprehensive Examination Unveiling Implications for Technology-Based Communication and Interaction.

## **1.2 Problem Statement:**

- **Gesture Recognition Accuracy Improvement** : Use cutting-edge machine learning and computer vision techniques to improve accuracy.
- **Real-Time Performance Optimization** : Enhance for immediate feedback and interactive interfaces.
- **Robustness in Varied Environments** : Use noise reduction and adaptive techniques to provide dependability in various conditions.

Hand gesture detection and recognition are moving towards edge computing based processing data at the device edge for reduced latency and investigation of multi-modal interaction combinations based gestures accompanied by voice and face expression.

The central topic of this project report deals with the complexities associated with hand gesture detection and recognition, which are considered an imminent phenomenon contributing to modifying conventional paradigms governing people's interrelations with computers through technological innovations.

Hand Gesture Detection: A Comprehensive Examination Unveiling Implications for Technology-Based Communication and Interaction.

## **1.3 OBJECTIVES**

### **1. Organic Human-Computer Communication:**

Users can use common everyday gestures such as pointing using hands as the inputs that will allow them to communicate naturally with computers and other electronic devices.

### **2. Immersive Virtual and Augmented Reality Experiences:**

With that, users derive maximum utility from their virtual as well as augmented reality applications through allowing hand movements for navigation and interaction.

### **3. Easily accessible:**

Let disabled users handle a device or program via gesturing instead; they should have user interface tailored specifically for them. They are particularly important for those suffering mobility problems.

### **4. Interpretation of Sign Language:**

Translate sign language into spoken or written language to facilitate interaction between those using sign language and others.

### **5. Real-Time Gesture Analysis:**

Real time processing of hand movements assures instant and accurate detection. It becomes highly important in cases where quick response is vital, including in entertainment and emergency situations.

### **6. Improving User Experience:**

Offer an alternative and additional means of interaction distinct from traditional input devices such as keyboard and mouse for enhancing end users' enjoyment through an easy interface.

### **7. Adaptability to Diverse Users:**

Ensure that the systems accommodate a variety of users with different hand sizes, shapes, and motion patterns for inclusive purposes.

## **1.4 SIGNIFICANCE AND MOTIVATION OF THE PROJECT WORK:**

Hand gesture detection and recognition are quite relevant in the area of human-computer interaction as there exist multiple domains and applications where it holds importance. The following salient features underscore the significance of this technology:

### **1. Natural Interaction:**

The hands are a vital part of human communication, especially for language acquisition. Through this, people can interact with computers and other devices in a manner similar to face-to-face communication by simply using hand gestures. This organic interaction can reduce the learning curve associated with traditional input techniques, and improve the user experience.

### **2. Immersive Virtual and Augmented Reality:**

The hand gesture detection gives ease of handling and interaction of digital objects in a virtual/augmented environment setting. This gives consumers a more natural mode of interaction/ navigation within the virtual environment thus making it even more immersive.

### **3. Interpreting Sign Language:**

To understand sign language, one should have the capability to identify gestural signals. This technology is inclusive and makes for a mutualistic interpretation of sign languages across societies through accurate sensing and translation mechanisms which improve communication between the deaf and the non-deaf populations of society.

### **4. Human-Robot Interaction:**

Hand gestures in robotics allow for better communication between humans and robots. This is of paramount importance during group engagements where precise communication is a vital requirement for collaborative endeavours.

### **5. Innovative User Interfaces:**

Hand gesture detection takes input beyond convention and provides for an inventive interface.

The emergence of new services, apps and devices which entertain while at the same time being helpful.

## **1.5 ORGANISATION OF THE PROJECT REPORT:**

The rest of this report is organised as follows:

Chapter 2 gives an overview of literature study performed

→ An overview of the body of research on sign gesture detection, examining different approaches, technology, and datasets within the framework of deep learning.

Chapter 3 discuss the system development and overflow

→ Thorough examination of the particular specifications needed to build an Detection system for Sign Language Gestures, with a focus on the demand for an original dataset.

Chapter 4 shows the performance analysis

→ An explanation of the testing methodology used to assess the precision and dependability of our system.

Chapter 5 highlights the conclusion , future scope and application contribution

→ The results of the system's performance are presented and interpreted, demonstrating how accurately it can recognise sign language gestures.

# CHAPTER 2: LITERATURE SURVEY

## INTRODUCTION:

Researchers have in recent times focused on the creation of models that can predict sign language gestures for the benefit of the deaf and hard of hearing individuals.

The programs implemented using modern technologies and artificial intelligence (AI) demonstrate that every effort is made to deliver high-quality care to people with hearing disabilities within society at large. Despite the many SLR studies, it remains necessary to enhance communication channels and hearing-impaired accessibility while acknowledging current restrictions.

This literature review examines the status of SLR currently, focusing on research articles applying deep learning methods with sensing and vision techniques. In this context, researchers have examined multiple ways to address the intricate issues of gesture recognition in video by using different methods.

This diversification in these approaches showcases how hard the work is, also shows that the different research communities collaborated together to give complete and efficient remedies. The purpose of this review is to add to the current debate regarding the enhancement of SLRs and the production of new technologies for the interest of people with hearing challenges.

## 2.2 DIFFERENT RECOGNITION APPROACHES

### 2.21. [1]A Comprehensive Study on Deep Learning-based Methods for Sign Language Recognition

#### --INTRODUCTION

- A deep learning model for sign language detection and recognition in paper
- Recognizing signs of Indian Sign Language video frames using LSTM and GRU models.

- Data set used is dataset IISL2020 of eleven handwritten digits attaining 97% accuracy.
- Provides means for enabling speech or hearing-impaired people to express themselves.

## **--RESULTS**

The presented model yields about 97 percent of success with respect to 11 types of characters.

- Among the various models, LSTM-GRU provides the highest accuracy.
- A super server was utilized for training the model using Keras-TensorFlow libraries.
- This can further help in improving the results by having a larger dataset.
- The suggested prototype has a high recognition percentage of real time signature recognition and perception.

## **--LIMITATIONS**

- High complexity makes standard dynamic sign language datasets difficult.
- Specifically, region-based languages such as Indian Sign Language take a lot of effort.
- The data used in previous research studies was collected under ideal environments. - The proposed methodology got 97% accuracy of sign recognition on the IISL2020 dataset.
- There are problems associated with non-stable as well as angular input data frames that should be improved.
- Sign recognition poses challenges at different levels of brightness.

## **2.22. [2]Deepsign:Sign Language Detection and Recognition using Deep learning**

### **--INTRODUCTION**

- A comparative study on computer vision based methods for sign language detection and recognition.



- Recent developments in deep neural network methods.
- A complete analysis across several public data sets.
- Map the unsegmented video streams to glosses.
- Development of new sequence training guidelines and pre training plans.
- Proposing a new RGB+D dataset for sign language from Greece.
- Three-level annotations of sign language in the first sign language video database.

## **--RESULTS**

- DNN-based SLR architectures comparative overview using publicly available datasets.
- A new large-scale RGBD dataset for Greek SL.
- Comparison of two CTC variations in CSLR evaluation. - Isolated SLR is more effective using 3D CNN-based architectures.
- In CSLR, 2D CNN-based models with per gloss representation outperformed other models.
- The proposed pre-training scheme and EnStimCTC achieved state of the art results on CSLI.
- Future, incorporating depth information into it and attentional approaches.
- The role of the linking in the SLR-SL translation.

## **--CONTRIBUTION**

- Systematic review of various SLR literature DNN approaches.
- Developing a new GSL dataset.
- Presentation of findings, conclusions and arguments.

## **2.23 [3]An Exploration into Human–Computer Interaction: Hand Gesture Recognition Management in a Challenging Environment**

### **--INTRODUCTION**

- This paper is concerned with hand gesture-based HCI.
- It seeks to enhance communications for the deaf and disabled society.
- It utilizes image segmentation as well as a CNN-based model.
- With segmentation, the optimal model attained an accuracy rate of 58%, which constituted a 10% improvement compared to no segmentation.
- This paper aims at addressing the gaps associated with previous research concerning gesture recognition.

### **--RESULTS**

- This hand gesture recognition system comprises two elements.
- The best performing model attained an accuracy of 58%.
- Using image segmentation increased the accuracy rate by 10% as opposed to using it without segmentation.

### **--LIMITATIONS**

- Light and crowded scenes hinder vision-based recognition.
- Data used cannot be accessed publicly because of privacy.

## **2.24.[4]HAND GESTURE RECOGNITION FOR INDIAN SIGN LANGUAGE**

### **--INTRODUCTION**

- A hand gesture recognition system for the Indian sign language is postulated in this paper.
- System consists of 4 modules: Hand Tracking, Segmentation, Feature Extraction, and Gesture Recognition.
- Hand tracking and hand's segmentation using a camshift method and HSV color model.
- Gesture recognition using a Genetic Algorithm.
- The system can distinguish between single-handed and double-handed gestures precisely.
- The purpose of the system is to aid deaf people in communicating efficiently with others.

### **---RESULTS**

- The proposed system applies the CamShift method, HSV (Hue, Saturation, Value) color model, and Genetic algorithm.
- The system is also able to cater for varying types of hand gestures.
- Suitable for single-handed as well as doubled hand gestures.
- It is cheap, but it can be used by deaf-mute people.
- Recognizes the Indian sign language alphabet effectively.
- Many deaf people will be able to communicate among themselves.

### **--CONTRIBUTION**

- Development of an Indian gesture recognition system.
- Accurate recognition of single handed and double handed gestures.

- Single normal webcam for gesture recognition.
- With maximum accuracy and as minimum time as possible when recognizing gestures.

## **2.25.[5]COMBINING HAND DETECTION AND GESTURE RECOGNITION ALGORITHMS FOR MINIMISING COMPUTATIONAL COST**

### **--INTRODUCTION**

- Neural networks for gesture reconstructing in paper
- Human computer interaction involves gesture recognitions.
- These systems are slow and very resource consuming.
- The proposed solution involves both hand detection and gestures recognition algorithms.
- It is easier compared to hand detection, which consumes minimal computing power.
- Proposed system is evaluated using public gesture bases and videos.
- The proposed system design is confirmed based on the experiment.

### **--RESULTS**

Four real test videos using hand detection and gesture recognition.

- Hands are detected on an average of ~75 frames per second.
- To this end, the proposed algorithm improved the average processing time of one video frame such that it became 26.1.
- These results are in agreement with calculations within the margins of error.
- The combined usage of the gesture classifier and hand detector makes efficient use of computing resources.

## **--LIMITATIONS**

- Poor performance and too much consuming of computing resources not playing any part in the picture.
- Complexity of decomposition of the neural network into separate steps.
- Optimization, fusion to other algorithms and so forth.

## **2.26.[6]REDUCTION OF GESTURE FEATURE DIMENSION FOR IMPROVING THE HAND GESTURE RECOGNITION PERFORMANCE OF NUMERICAL SIGN LANGUAGE**

### **--INTRODUCTION**

The hand gesture recognition is one of the important forms of non-touch human-computer interaction.

- This research work introduces a competent hand gesture recognition system using hand attributes selection.
- Discrete wavelet transformation and singular value decomposition are used for hand feature extraction.
- Optimal features are chosen using a genetic algorithm.
- Hand gestures recognition is supported by a support vector machine.
- The model is better as compared to traditional hand recognition.

### **--RESULTS**

- This new model exceeds conventional manual approaches of hand recognition.

- Discrete wavelet transformation and singular value decomposition for hand feature extraction.
- Feature selection is implemented using genetic algorithms with effective fitness function.
- Hand gesture recognition using support vector machines.
- A constructed hand gesture dataset for validation of the proposed model.

#### **--CONTRIBUTIONS**

- User-oriented research and development, which include interfacing, are the vital area of HCI.
- A webcam based model that cuts down on cost and increases reliability is proposed.
- Discrete Wavelet Transformation (DWT) and Singular Value Decomposition (SVD) are used to obtain hand features.
- Efficient feature selection using genetic algorithms with an appropriate fitness function.
- Hand gesture recognition using a support vector machine.
- A proposed model is more effective than traditional manual hand identification systems.

## 2.27 LITERATURE SURVEYS

S. No.	Paper Title [Cite]	Journal/ Conference (Year)	Tools/ Techniques/ Dataset	Results	Limitations
1.	An Exploration into Human-Computer Interaction: Hand Gesture Recognition Management in a Challenging Environment	2023	image enhancement and segmentation using colour space conversions - Machine learning algorithms, specifically CNN, for hand gesture recognition	The optimal model achieved a performance of 58% accuracy. - Image segmentation improved accuracy by 10% compared to without segmentation.	- Vision-based recognition is impacted by light and crowded surroundings. - The data used in the study is not publicly available due to privacy concerns.

S. No.	Paper Title [Cite]	Journal/ Conference (Year)	Tools/ Techniques/ Dataset	Results	Limitations
2.	Deepsign: Sign Language Detection and Recognition Using Deep Learning	2022	LSTM and GRU, InceptionResNetv2,Dropout	<p>The proposed model has a high recognition rate for real-time sign detection and recognition.</p> <p>-The LSTM-GRU model has the highest accuracy among the different combinations. - The model was developed using KerasTensorFlow libraries and trained on a SuperServer.</p>	<p>- Standard dynamic sign language datasets are challenging due to high intricacy. - Working on specific region-based languages like Indian Sign Language is challenging. - Previous datasets used in research were designed in ideal environments.</p>



S. No.	Paper Title [Cite]	Journal/ Conference (Year)	Tools/ Techniques/ Dataset	Results	Limitations
3.	Hand Gesture Recognition with Color Descriptors	2020	SIFT,DPL algorithm	successfully recognized 36 classes of hand gestures with a high recognition accuracy of 97.18% for isolated hand signs .	Computational complexity efficiency of the proposed architecture not discussed
4.	Combining Hand Detection and Gesture Recognition Algorithms for Minimising Computational Cost	2020	CNN and ViBe algorithm	benefits in terms of performance increase and experimental results were consistent with theoretical estimates	low performance and excessive consumption computing resource when there is no hand the frame

S. No.	Paper Title [Cite]	Journal/ Conference (Year)	Tools/ Techniques/ Dataset	Results	Limitations
5.	An Improved Hand Gesture Recognition With Two-Stage Convolutional Neural Networks Using a Hand Colour Image And Its Pseudo-Depth Image	2019	Two-Step based CNN approach	Overall accuracy for hand gesture recognition was improved with pseudo-depth images: 88.45% for feature fusion and 87.4% for committee fusion, compared to 80.71% for colour image only and 85.76% for depth	Computational complexity efficiency of the proposed architecture not discussed

<b>S. No.</b>	<b>Paper Title [Cite]</b>	<b>Journal/ Conference (Year)</b>	<b>Tools/ Techniques/ Dataset</b>	<b>Results</b>	<b>Limitations</b>
6.	A Novel Dynamic Hand Gesture and Movement Trajectory Recognition model for Non-Touch HRI Interface	2019	Kinect sensor, Skin colour segmentation model, Machine learning techniques	The proposed dynamic hand gesture and movement trajectory recognition system achieved an average recognition accuracy of 94.5% for dynamic motion instruction identification	Reliance on Kinect sensor, Small sample size and Lack of detailed information
7	Research on the Hand Gesture Recognition Based on Deep Learning	2019	AdaBoost, Cam-Shift algorithm, CNN	Achieved 98.3% accuracy in recognizing 10 common digits.	The specific dataset used for training and testing the CNN is not mentioned, which could affect the generalizability of the results

<b>S. No.</b>	<b>Paper Title [Cite]</b>	<b>Journal/ Conference (Year)</b>	<b>Tools/ Techniques/ Dataset</b>	<b>Results</b>	<b>Limitations</b>
8.	[8]Dynamic hand gesture recognition based on 3D pattern assembled trajectories	2018	LIBSVM	state-of-the-art performance with 90.48% accuracy for 14 gestures and 80.48% accuracy for 28 gestures	The unsegmented gesture recognition has lower accuracy due to the challenges in gesture detection and segmentation
9.	[9]Reduction of Gesture Feature Dimension for Improving the Hand Gesture Recognition Performance of Numerical Sign Language	2018	SVM and Genetic Algorithm	Average accuracy improved from 61.15% without feature selection to 77.55% with feature selection	The paper does not discuss the computational complexity and time required for feature selection using GA.

<b>S. No.</b>	<b>Paper Title [Cite]</b>	<b>Journal/ Conference (Year)</b>	<b>Tools/ Techniques/ Dataset</b>	<b>Results</b>	<b>Limitations</b>
10.	[10]Hand Gesture Recognition Based on Improved Histograms of Oriented Gradients	2017	Improved Histograms of Oriented Gradients (HOG), Cascaded SVM, Skin Similarity	Recognition rates of over 91% were achieved for most hand gestures, with some confusion between similar gestures.	The method may not tolerate translation, rotation, and scaling well without sufficient training data for variations.
11.	[11]Real time hand gesture movements tracking and recognizing system	2016	webcam,convex hull algorithm, lucas kanade algorithm	The system can operate well, allowing users to interact with the computer using hand gestures instead of a mouse	Focused on basic hand gestures and didn't explore a wide range of gestures.

# CHAPTER 3 : SYSTEM DEVELOPMENT

## 3.1 REQUIREMENT AND ANALYSIS

In order to achieve all these preparations and high levels of accuracy, the system must be able to identify different gestures and movements, acquire enough high-quality images and videos. The system has to have the flexibility to operate at real time, capable of proper identification on the data generated from different sources and under different lighting conditions for the best version. Afterward, the resultant version should be able to generate a reliable output without losing its fidelity to the true classification.

**1. Hand detection and segmentation:** The system should specifically pinpoint and separate out the hand part in a picture of the input image or in an input video frame. In a nutshell, any part of the hand ought to have a different color to that of the scenery and its accessories.

**2. Feature extraction:** Secondly, some significant parameters critical for taking the outline of the hand, position and movements must be incorporated. Typical HGR includes contour points, edge features, as well as key points.

**3. Gesture Classification:** The system has to organize these features within their predefined groups that constitute this gesture representing the intended activity. There are various forms of machine learning methodologies that might be included in this categorisation problem, including SVMs, KNN or Neural Network.

**4. Real-Time Performance:** This should be a live working system so as to provide users with a consistent, responsive environment. Therefore, it needs suitable algorithms together with optimization strategies as ways of minimizing the processing span.

**5. Robust Against Environmental Variations:** For example, this system should remain constant to any changes in light, noise or shadows. It is, therefore, imperative for adaptive algorithms to be developed to readjust these components but should not affect discrimination ability.

One of the most important requirements for this project is a library known as MediaPipe.

## What is MediaPipe?

An open-source hybrid architecture known as MediaPipe has to be developed in order for pipelines to be able to process such perceptual data as images, video or audio. An algorithm that uses machine learning for real-time gesture recognition as well as hand tracking has been proposed. Its accuracy in identifying signs also means more hand and finger-tracking features are available.

```
# STEP 1: Import the necessary modules.
import mediapipe as mp
from mediapipe.tasks import python
from mediapipe.tasks.python import vision

# STEP 2: Create an GestureRecognizer object.
base_options = python.BaseOptions(model_asset_path='gesture_recognizer.task')
options = vision.GestureRecognizerOptions(base_options=base_options)
recognizer = vision.GestureRecognizer.create_from_options(options)

images = []
results = []
for image_file_name in IMAGE_FILENAMES:
    # STEP 3: Load the input image.
    image = mp.Image.create_from_file(image_file_name)

    # STEP 4: Recognize gestures in the input image.
    recognition_result = recognizer.recognize(image)

    # STEP 5: Process the result. In this case, visualize it.
    images.append(image)
    top_gesture = recognition_result.gestures[0][0]
    hand_landmarks = recognition_result.hand_landmarks
    results.append((top_gesture, hand_landmarks))

display_batch_of_images_with_gestures_and_hand_landmarks(images, results)
```

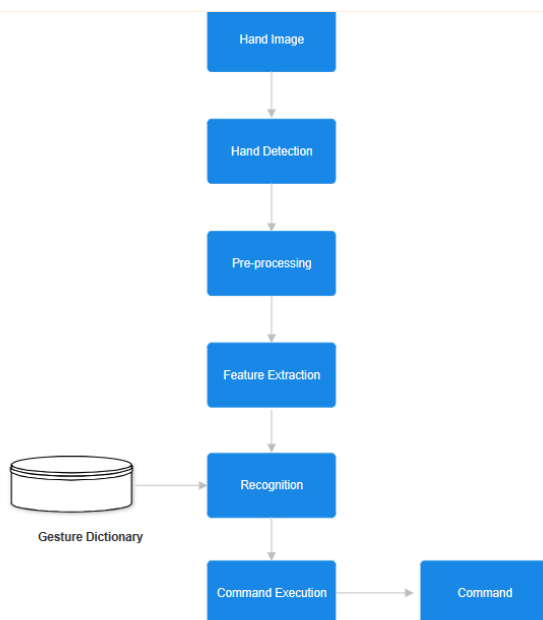
**Fig 1** Mediapipe Import (It Tells us the code required to import and use mediapipe library)

By merging the components of the pose, face, and the hand landmarks in the MediaPipe Holistic Landmarker project, we could come up with a full set marker for the human body. Actions, stances and full body motions can be evaluated using this assignment. An ML model will take in a constant stream image that it can learn from in this assignment. This process performs live output of 543 landmarks overall (33 position landmarks, 468 faces landmarks, twenty one hands landmarks per hand).

### 3.2 PROJECT DESIGN AND ARCHITECTURE

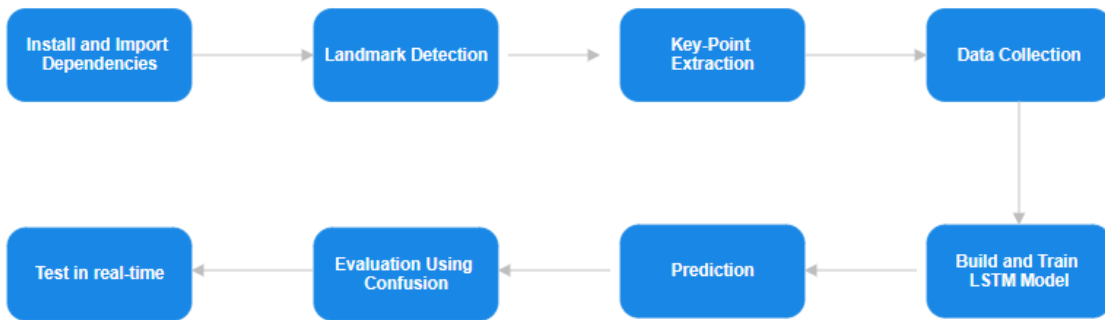
The approach is based on vision. The use of hands means that there is no need for any artificial devices to be used to facilitate interaction as all the indications are shown by the hands.

During the search for ready-to-use datasets suitable for this project, we could not find any relevant one. Hence, we decided to develop our own data set. For the generation of a dataset we used Open Computer Vision (OpenCV).

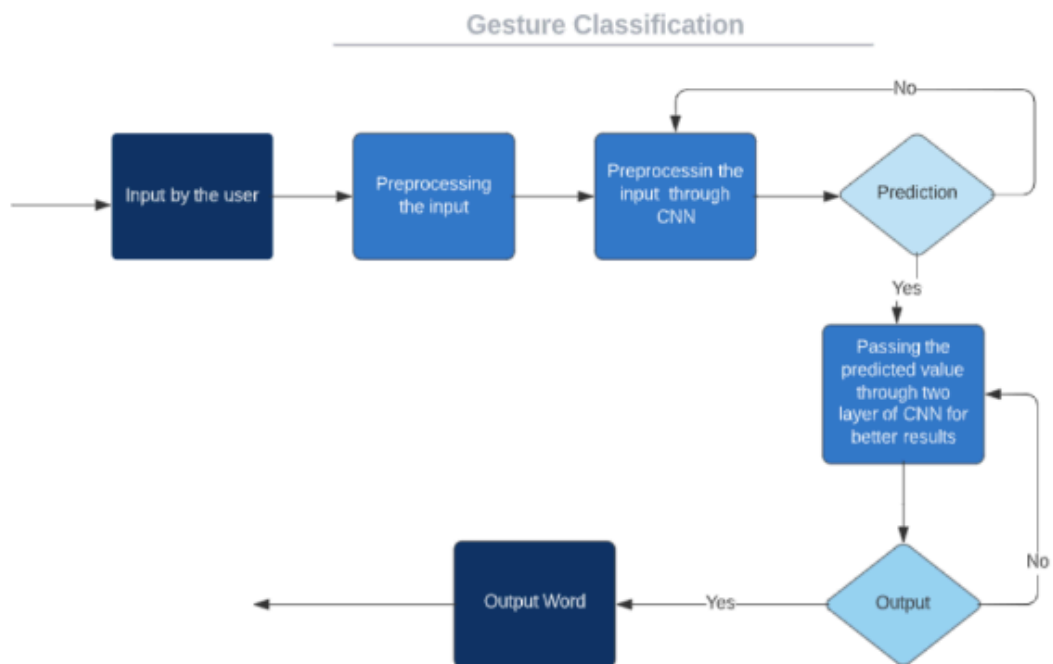


**Fig2** Basic flow of the project using simple keras model(It is the process taken into account for making the basic model )





**Fig3** Detailed flow of the project using LSTM(It is the step by step process which is followed for making the LSTM model)



**Fig4** (Flow chart of the process implementation when using the CNN model)

### **Algorithm Layer 1:**

1. To obtain the processed image following feature extraction, apply the Gaussian Blur filter and threshold to the frame captured using openCV.
2. After this image has been processed, it is fed into the CNN model for prediction. If a letter is identified in more than 50 frames, it is printed and used to build the word.
3. The blank symbol is used to represent the space between the words.

### **Algorithm Layer 2:**

1. We identify other sets of symbols that, when identified, yield comparable outcomes.
2. Next, we use classifiers designed specifically for those sets to classify between those sets.

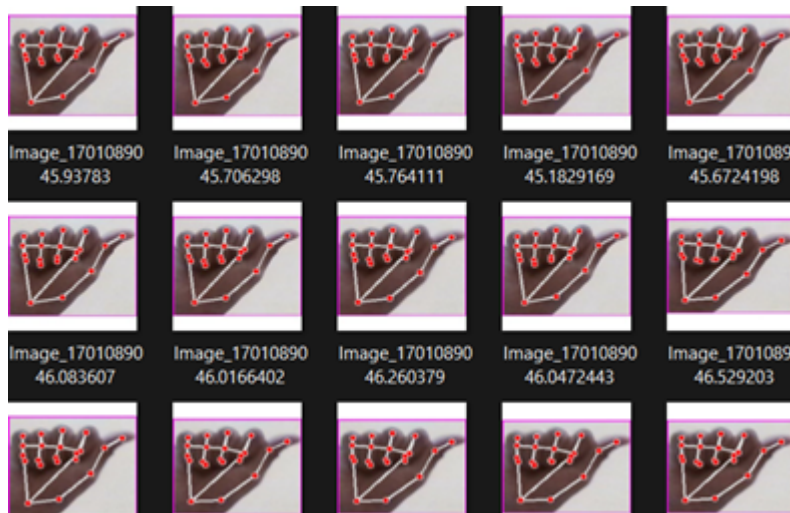
### **Collection of Hand Gesture Images:**

The first step entails getting the video frames and the images in this case with regard to the hand gestures. This can be accomplished in a number of ways, such as:

Webcam: Such real-time photographs of hand gestures can be taken by a web-enabled computer.

Hand-held devices: The use of smartphones or tablets could be used in recording hand gestures or specifically designed hand gesture recognizers.

Gesture databases: For education about hand gestures, there are gesture databases available to the public with a collection of photographs that can be used to train and test.



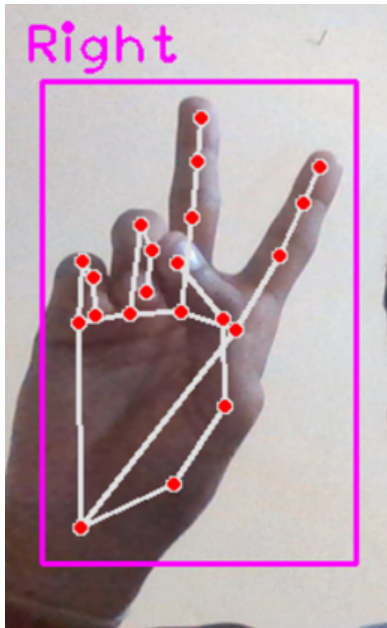
**Fig 5** Data Collection(Personal Data Collected)

### **Hand Recognition**

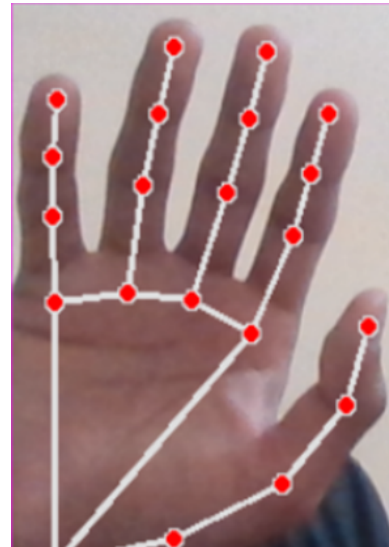
Hand detection involves localizing and differentiating the hand region from the input image or frame. This crucial step ensures extraction of relevant features that are necessary for gesture recognition. Typical methods for detecting hands include:

backdrop subtraction: It involves determining the background of the image where the hand serves as a foreground object and eliminating it.

The process of defining edges that help identify the hand area from the image is known as edge detection.



**Fig6** Hand Detected



**Fig7** Hand Cropped

### **Image Preparation**

Once the hand is detected, its segmented image is pre-processed to enhance its quality and make it ready for feature extraction. Preprocessing actions could consist of:

Noise reduction refers to removing unwanted noise in an image that can use filters or various smoothing techniques.

Normalisation: The brightness, contrast, and color balance of an image is adjusted to enhance feature extraction.

Image resizing: Resize the image to a common size for easier feature extraction and gesture classification.

### **Feature Deletion**

The relevance of features extracted for removal of irrelevant features which correctly describes the shape, positioning and movement of hand taken input pre-processed hand image. The characteristics form part of the gesture classification model. Typical methods for

extracting features are as follows:

Contour points: Obtaining perimeter points for the hand area.

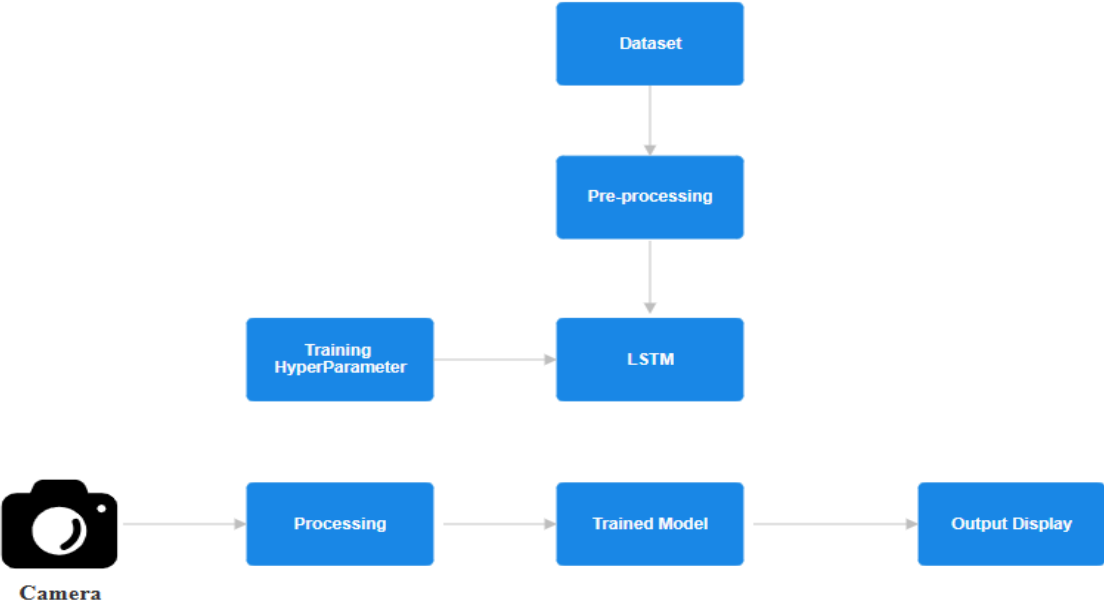
For example, one of the edge features includes assessing where the hand lies, as well as orientation and strength of the edges within the hand region.

Important points: Detection of separate landmarks in hand like knuckles or finger tips.

**Recognition of Gestures**

The features obtained are then passed to a gesture classification algorithm that places them in one of the reserved classes. Several machine learning algorithms are frequently employed for the classification of gestures, such as:

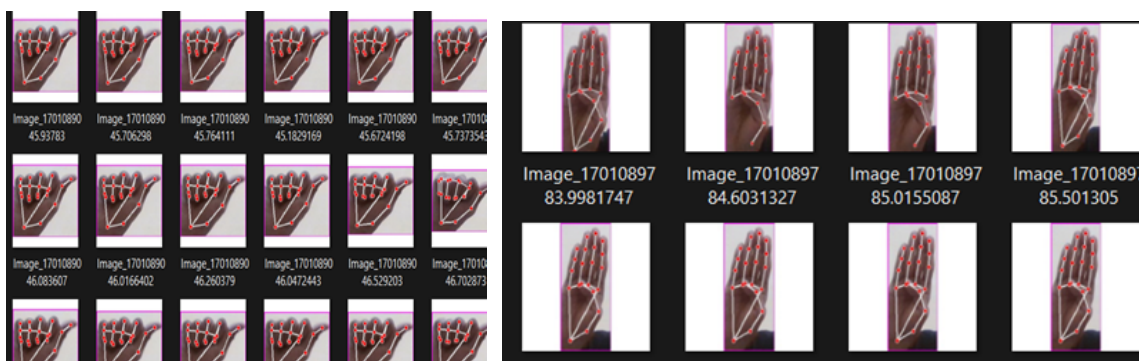
Using neural networks: Given that neural networks are capable of learning complex patterns from data, they should be suitable for classification gestures.



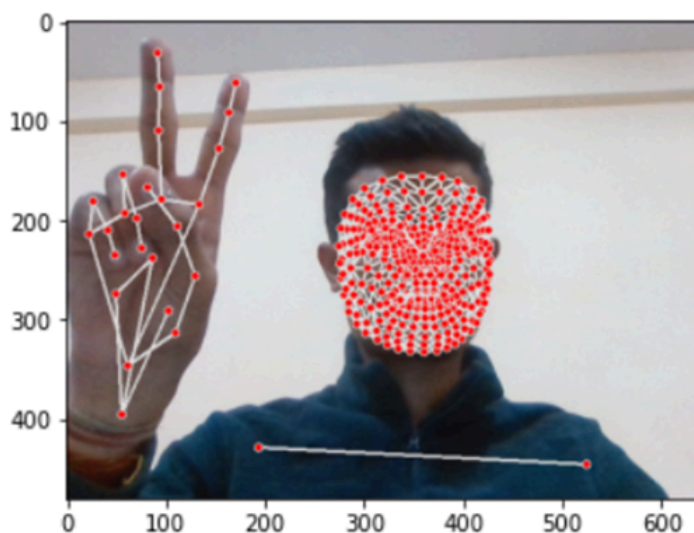
**Fig8** Architecture of the Hand Gesture Recognition System

### 3.3 DATA PREPARATION

The first step for the HGR project is to collect a quality dataset for effectively training and validating the hand gestures which comprises a wide variety of quality photos. The dataset should contain a versatile, appropriate labelling, and different types of hand postures, different angles under varied lighting conditions. To ensure the variety in the dataset it should contain images of hands of different skin tones with different hand sizes, different symbols and in different lighting conditions which bring stability in the system's decision making for hand detection.



**Fig9** Images for data collected on different signs



**Fig10** Image for showing different landmarks using mediapipe

To create this self-developed dataset, we make use of the inbuilt system camera to record the hand gesture images under different conditions of lights and background. The dataset includes images of individuals performing the hand gestures and making sure that the symbol they are making is accurate. Images were taken with consistent focus, resolution, and framing in order to preserve the quality of the data. Furthermore, proper hand gesture labels will be provided in-depth annotations for every photograph.

## **3.4 IMPLEMENTATION**

### **3.4.1 Mediapipe**

For computer vision inference, pipelines handling any kind of sensory data, such as audio and video, can be designed using the open-source MediaPipe framework. MediaPipe may construct such a perception pipeline into a graph made up of interchangeable parts.

The primary use case for the MediaPipe framework is the quick construction of perception pipelines using AI inferencing models and reusable components. Furthermore, it makes the integration of machine vision technology easier across a range of applications and demo versions of the majority of hardware platforms. Over time, teams will be able to enhance computer vision pipelines with the aid of configuration languages and assessment tools.

### **3.4.2 Holistic Model of Mediapipe**

Any kind of sensory data, including audio and video, can be processed using pipelines constructed with the open-source MediaPipe framework to carry out computer vision inference. A perception pipeline of this kind may be constructed as a graph of modular components using MediaPipe.

The MediaPipe framework's primary application is the quick creation of perceptual pipelines using AI models for other reusable parts like inferencing. Additionally, it makes it easier to incorporate computer vision technologies into applications and demos on a variety of hardware platforms. Teams may gradually enhance machine vision pipelines by utilising the evaluation tools and configuration language.

### 3.4.3 Hand Detection using Mediapipe Holistic model

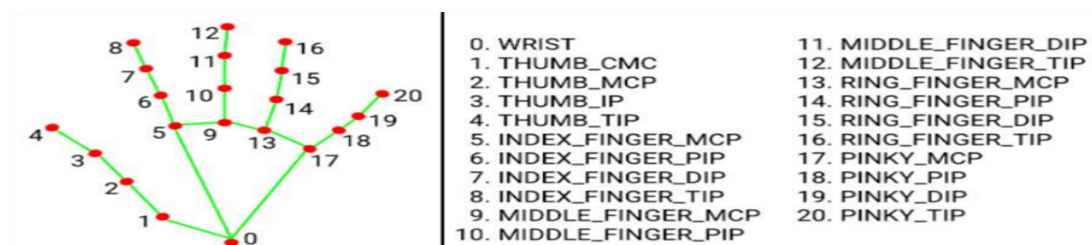
Pose, Hand, and Face are the three components of the mediapipe holistic pipeline. All in all, Mediapipe provides a wide range of choices. 543 unique landmarks are recovered using the holistic model (468-face, 33-pose, and 21-hand).

It makes use of the Mediapipe holistic model, which is included in the Mediapipe Python package. We created a function called mediapipe detection with two arguments.

1. Image: The image that serves as its detecting source.
2. Model: The model used for detection is the mediapipe ("Holistic model") model. The function first converts the image from BGR to RGB format, which is then used to feed the output into the model and store it in the process() method. The function then returns the image and the saved result after it has been transformed.

The output from the functions that came before it is passed as an argument to another function we construct, called draw styled landmarks. We utilise this feature to help visualise real-time hand detection by drawing landmarks. Extract the main ideas: The X, Y, and Z coordinates of the important points are extracted from the detection results (the stored result provided by the mediapipe detection function previously explained) once real-time hand detection has been achieved.

We create an additional method called extract key points that takes the detection results as an input and uses them to extract the coordinates. The concatenated array of all the arrays containing the key point coordinates for the holistic model is the result of this function. This function is called during the data collection process.



**Fig11 Hand landmarks**



### 3.4.3.1 Hand Detection using CNN model

The neurons of CNN layers are arranged in three dimensions: width, height, and depth, in contrast to ordinary neural networks. Only one neuron per layer will be coupled to a small area (window size) of the layer prior to it, as opposed to every neuron in a fully linked fashion. Because we will have reduced the entire image to a single vector of class scores at the end of the CNN architecture, the final output layer would also contain dimensions (number of classes).

1. **Convolutional Layer:** We use a modest window size in the convolution layer that extends to the input matrix's depth, usually measuring 5 by 5. Learnable window-sized filters make up the layer. Every time, we compute the dot product of the filter entries and input values at a particular place after sliding the window by a stride size, which is usually 1. A 2-Dimensional activation matrix that displays the response of the matrix at each spatial position will be created as we go with this process. That is, the network will pick up filters that become active upon detecting a certain kind of visual feature, such a coloured patch or an edge with a specific orientation.
2. **Pooling Layer:** In order to lower the size of the activation matrix and, eventually, the learnable parameters, we employ a pooling layer. Two categories of pooling exist:
  - a. **Max Pooling:** This technique uses a window size, such as a 2\*2 window, and only takes the highest four values. We'll close this window and carry on until we eventually get an activation matrix that is half the size it was originally.
  - b. **Average Pooling:** This method makes use of every Value within a window.

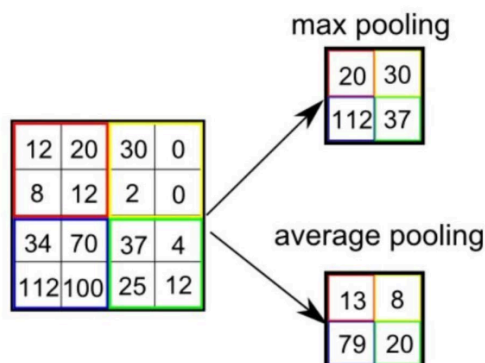


Fig12 Pooling Matrix

**3. Fully Connected Layer:** In a fully connected region, all inputs will be connected to neurons, however in a convolution layer, neurons are only connected to a small region.

**4. Final Output Layer:** We will connect the information from the completely connected layer to the last layer of neurons, whose count equals the total number of classes, so that it can forecast the likelihood that each image will belong to a different class.

## Layer 1:

### CNN Model:

**1. First Convolution Layer:** The input image has a resolution of 128 x 128 pixels. It is processed in the first convolutional layer using 32 filter weights (3 x 3 pixels each). This will produce an image of 126 x 126 pixels, one for each filter-weight.

**2. First Pooling Layer:** The images are down sampled using max pooling of 2 x 2, meaning we retain the highest value in the 2 x 2 square of array. Consequently, our image is down sampled to 63 x 63 pixels.

**3. Second Convolution Layer:** The 63 x 63 from the first pooling layer's output are now used as an input to the second convolutional layer. They are processed in the second convolutional layer using 32 filter weights (3 x 3 pixels each). This will produce a 60 x 60 pixel image.

**4. Second Pooling Layer:** Using a maximum 2 x 2 pool, the final images are down sampled once more and are lowered to a resolution of 30 x 30.

**5. First Densely Connected Layer:** The output of the second convolutional layer is reshaped into an array of  $30 \times 30 \times 32 = 28800$  values. These images are now fed into a fully connected layer with 128 neurons. This layer receives an array of 28800 values as input. The second densely connected layer receives the output from these layers. To prevent overfitting, we use a dropout layer with a value of 0.5.

**6. Second Densely Connected Layer:** A completely connected layer with 96 neurons now receives its input from the output of the first densely connected layer.

**7. Final layer:** The number of neurons in the final layer (alphabets plus a blank sign) corresponds to the number of classes we are classifying. This layer receives its input from the output of the second densely connected layer.

**The Activation Function:** Rectified Linear Units, or ReLUs, have been incorporated into every layer (including fully linked and convolutional neurons).  $\text{Max}(x,0)$  is determined for every input pixel by ReLU. This enhances the formula's nonlinearity and facilitates the learning of increasingly complex features. It assists in solving the vanishing gradient issue and shortens the computation time, which speeds up training.

**Pooling Layer:** Using ReLU activation function and a pool size of (2, 2), we apply Max pooling to the input image. This lowers the number of parameters, which lowers the cost of computing and lessens overfitting.

**Layers of Dropout:** The issue of overfitting occurs when, following training, the weights of the network become so adapted to the training instances that it becomes unresponsive to fresh examples. By setting them to zero, this layer "drops out" a randomly selected subset of activations in that layer.

**Optimizer:** In order to update the model in response to the loss function's output, we used the Adam optimizer. The adaptive gradient algorithm (ADA GRAD) and root mean square propagation (RMSProp), two extensions of stochastic gradient descent methods, are combined to create the Adam optimizer, which combines their benefits.

## **Layer 2:**

In order to get as near to detecting the displayed symbol as possible, we are employing two layers of algorithms to validate and forecast symbols that are more similar to one another. During testing, we discovered that the following symbols were not displaying correctly and were also displaying additional symbols:

1. R and U for D
2. D and R for U

3. T, D, K, and I

4. M and N for S

We therefore created three distinct classifiers for these sets in order to address the aforementioned cases:

1. {D, R, U}

2. {T, K, D, I}

3. {S, M, N}

### **Implementing Finger Spelling Sentence Formation:**

1. We print the detected letter and append it to the current string whenever the count of that letter surpasses a given number (in our code, we kept the value at 50 and the difference threshold at 20).

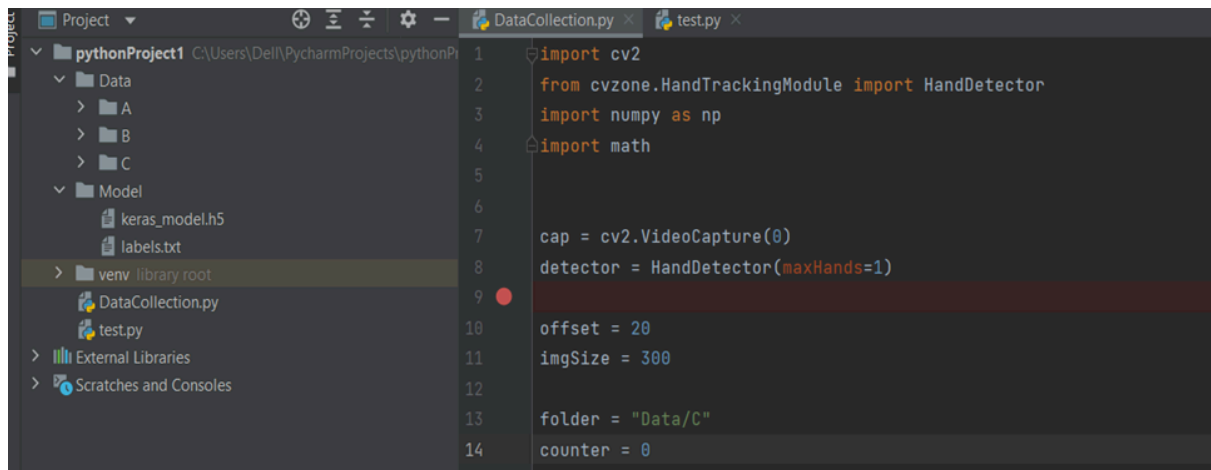
2. If not, we eliminate the current dictionary that contains the number of times the current symbol has been detected in order to reduce the possibility that the wrong letter would be predicted.

3. No spaces are recognized if the current buffer is empty or if the count of blank (plain backdrop) observed exceeds a certain value.

4. In the alternative, a space is printed to indicate that the word is about to end, and the current is appended to the sentence below.

### **3.4.4 Code for Simple Model :**

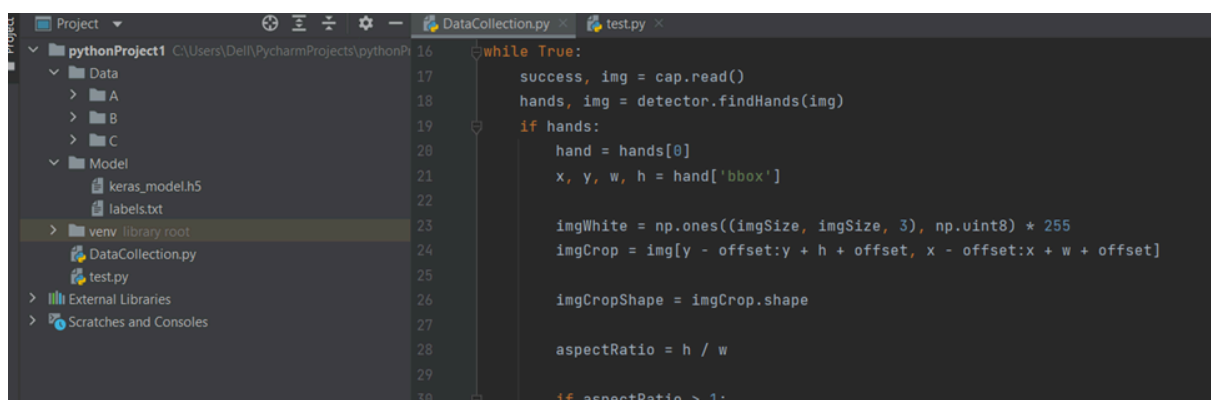
#### **Data Collection Code:**



```
1 import cv2
2 from cvzone.HandTrackingModule import HandDetector
3 import numpy as np
4 import math
5
6
7 cap = cv2.VideoCapture(0)
8 detector = HandDetector(maxHands=1)
9
10 offset = 20
11 imgSize = 300
12
13 folder = "Data/C"
14 counter = 0
```

**Fig13 Collection Code**

The project's required libraries are imported in these lines. HandDetector is a hand tracking module from the cvzone library, where cv2 is OpenCV. For numerical operations, numpy is imported as np, and for mathematical functions, math is imported. In this instance, video from the default camera (index 0) is captured by creating a video capture object (cap). Additionally, an item called the Hand Detector is created, with a maximum hand count of 1. These lines create two constants: imgSize, the size of the square image used for hand sign recognition, and offset, which adds extra space around the hand bounding box. These lines initialise a counter (counter) to track the number of collected photographs and set up a folder path (folder) where they will be saved.



```
16 while True:
17     success, img = cap.read()
18     hands, img = detector.findHands(img)
19     if hands:
20         hand = hands[0]
21         x, y, w, h = hand['bbox']
22
23         imgWhite = np.ones((imgSize, imgSize, 3), np.uint8) * 255
24         imgCrop = img[y - offset:y + h + offset, x - offset:x + w + offset]
25
26         imgCropShape = imgCrop.shape
27
28         aspectRatio = h / w
29
30         if aspectRatio > 1:
```

**Fig14 Collection Code**

```

30
31
32
33
34
35
36
37
38
39
40
41
42
43
if aspectRatio > 1:
    k = imgSize / h
    wCal = math.ceil(k * w)
    imgResize = cv2.resize(imgCrop, (wCal, imgSize))
    imgResizeShape = imgResize.shape
    wGap = math.ceil((imgSize - wCal) / 2)
    imgWhite[:, wGap:wCal + wGap] = imgResize
else:
    k = imgSize / w
    hCal = math.ceil(k * h)
    imgResize = cv2.resize(imgCrop, (imgSize, hCal))
    imgResizeShape = imgResize.shape
    hGap = math.ceil((imgSize - hCal) / 2)

```

**Fig15 Collection Code**

This loop uses `cap.read()` to take frames from the video feed. It then uses the `HandDetector`'s `findHands` function to find hands in the current frame. The `hands` variable contains the detected hands.

The function retrieves the bounding box (`x, y, w, h`) of the first hand detected if any are found. `NumPy` is used to build a white image (`imgWhite`) of size `imgSize`. This image will serve as the canvas for painting the resized hand region. The original frame's area of interest (`imgCrop`) is taken out using the hand bounding box plus the offset. For visualisation, the white canvas and the clipped hand region are shown in different windows.

```

44
45
46
47
48
49
50
51
52
53
54
hGap = math.ceil((imgSize - hCal) / 2)
imgWhite[hGap:hCal + hGap, :] = imgResize

cv2.imshow("ImageCrop", imgCrop)
cv2.imshow("ImageWhite", imgWhite)

cv2.imshow("Image", img)
key = cv2.waitKey(1)
if key == ord("s"):
    counter += 1
    cv2.imwrite(f'{folder}/Image_{time.time()}.jpg', imgWhite)
    print(counter)

```

**Fig16 Collection Code**

There is an additional window displaying the original frame. The number increases and the white canvas (`imgWhite`) is saved as a picture in the designated folder with a filename that contains the current timestamp if the 's' key is pressed.

**TEST Code:**

```
pythonProject1 C:\Users\Dell\PycharmProjects\pythonPi 1 import cv2
2 from cvzone.HandTrackingModule import HandDetector
3 from cvzone.ClassificationModule import Classifier
4 import numpy as np
5 import math
6
7 cap = cv2.VideoCapture(0)
8 detector = HandDetector(maxHands=1)
9 classifier = Classifier("Model/keras_model.h5", "Model/labels.txt")
10
11 offset = 20
12 imgSize = 300
13
14 folder = "Data/C"
15 counter = 0
```

**Fig17 Test Code**

The project's required libraries are imported in these lines. OpenCV is represented by cv2, and the cvzone library contains the HandDetector and Classifier modules for hand tracking and classification, respectively. For numerical operations, numpy is imported as np, and for mathematical functions, math is imported.

In this instance, video from the default camera (index 0) is captured by creating a video capture object (cap). The maximum number of hands that can be constructed for the HandDetector object (detector) is 1. An already-trained model (keras\_model.h5) and the label file that goes with it (labels.txt) are used to generate the Classifier object (classifier).

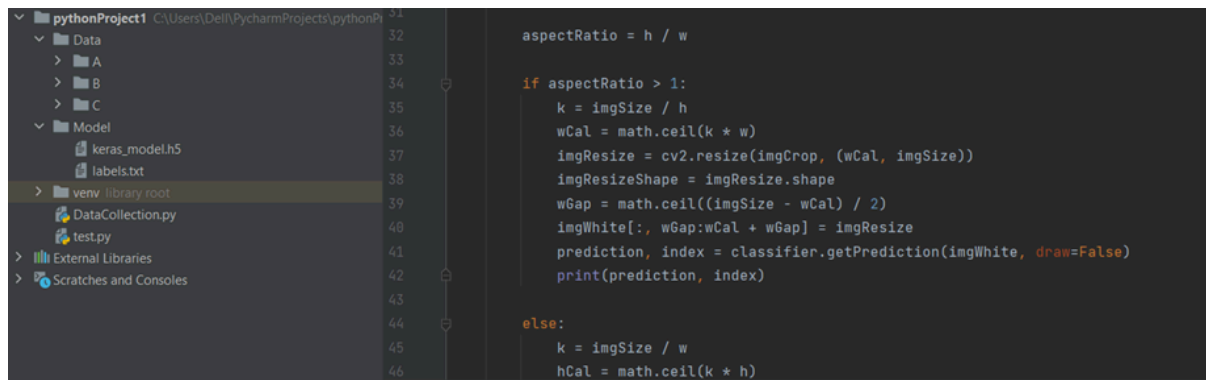
These lines define a folder path (folder) where the taken photographs will be saved, a counter (counter) to tally the number of images captured, and constants (offset, imgSize). The class labels that are used for classification are listed in the labels list.

```
Project 16
17 labels = ["A", "B", "C"]
18
19 while True:
20     success, img = cap.read()
21     imgOutput = img.copy()
22     hands, img = detector.findHands(img)
23     if hands:
24         hand = hands[0]
25         x, y, w, h = hand['bbox']
26
27         imgWhite = np.ones((imgSize, imgSize, 3), np.uint8) * 255
28         imgCrop = img[y - offset:y + h + offset, x - offset:x + w + offset]
29
30         imgCropShape = imgCrop.shape
```

**Fig18 Test Code**

This loop uses `cap.read()` to take frames from the video feed. It then uses the `HandDetector`'s `findHands` function to find hands in the current frame. The `hands` variable contains the detected hands.

The function retrieves the bounding box (`x, y, w, h`) of the first hand detected if any are found. After that, it makes a white canvas (`imgWhite`) and uses the hand bounding box with the offset added to extract a region of interest (`imgCrop`) from the original frame.



```

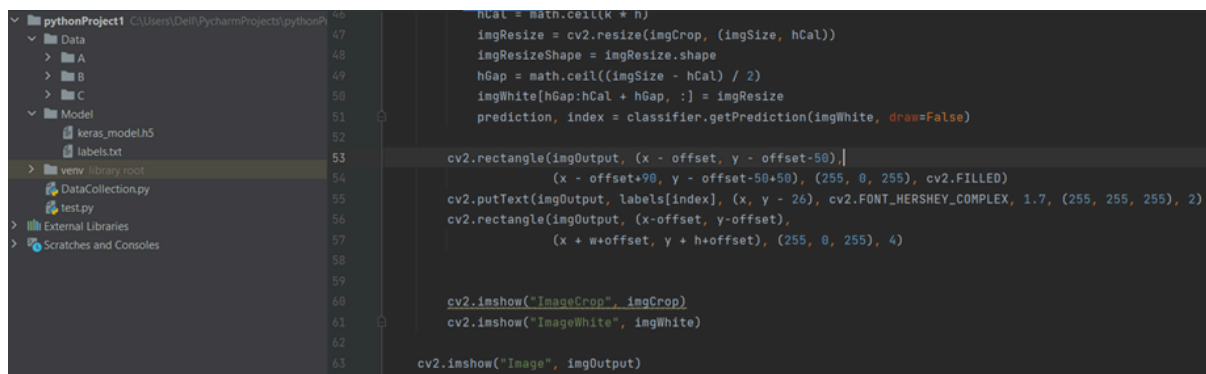
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
aspectRatio = h / w

if aspectRatio > 1:
    k = imgSize / h
    wCal = math.ceil(k * w)
    imgResize = cv2.resize(imgCrop, (wCal, imgSize))
    imgResizeShape = imgResize.shape
    wGap = math.ceil((imgSize - wCal) / 2)
    imgWhite[:, wGap:wCal + wGap] = imgResize
    prediction, index = classifier.getPrediction(imgWhite, draw=False)
    print(prediction, index)

else:
    k = imgSize / w
    hCal = math.ceil(k * h)

```

**Fig19 Test Code**



```

47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
hCal = math.ceil(k * h)
imgResize = cv2.resize(imgCrop, (imgSize, hCal))
imgResizeShape = imgResize.shape
hGap = math.ceil((imgSize - hCal) / 2)
imgWhite[hGap:hCal + hGap, :] = imgResize
prediction, index = classifier.getPrediction(imgWhite, draw=False)

cv2.rectangle(imgOutput, (x - offset, y - offset-50),
              (x - offset+90, y - offset-50+50), (255, 0, 255), cv2.FILLED)
cv2.putText(imgOutput, labels[index], (x, y - 26), cv2.FONT_HERSHEY_COMPLEX, 1.7, (255, 255, 255), 2)
cv2.rectangle(imgOutput, (x-offset, y-offset),
              (x + w+offset, y + h+offset), (255, 0, 255), 4)

cv2.imshow("ImageCrop", imgCrop)
cv2.imshow("ImageWhite", imgWhite)

cv2.imshow("Image", imgOutput)

```

**Fig20 Test Code**

The code for adjusting aspect ratio and resizing is comparable to that of the previous iteration. The prediction for the hand sign is obtained by using the `getPrediction` method from the `Classifier` object once the hand region has been resized. The expected label and index are printed together with the result. The code then uses the hand region and the classification result to create labels and rectangles on the output image (`imgOutput`). The photos that have been processed are shown for visualisation in several windows.



## Code for LSTM Model:

### 1. Import and Install Dependencies

```
In [1]: |pip install tensorflow==2.5.0 tensorflow-gpu==2.5.0 opencv-python mediapipe sklearn matplotlib
Requirement already satisfied: tensorflow==2.5.0 in c:\users\dell\anaconda3\lib\site-packages (2.5.0)
Requirement already satisfied: tensorflow-gpu==2.5.0 in c:\users\dell\anaconda3\lib\site-packages (2.5.0)
Requirement already satisfied: opencv-python in c:\users\dell\anaconda3\lib\site-packages (4.8.1.78)
Requirement already satisfied: mediapipe in c:\users\dell\anaconda3\lib\site-packages (0.8.11)
Requirement already satisfied: sklearn in c:\users\dell\anaconda3\lib\site-packages (0.0.post11)
Requirement already satisfied: matplotlib in c:\users\dell\anaconda3\lib\site-packages (3.5.1)
Requirement already satisfied: wheel~=0.35 in c:\users\dell\anaconda3\lib\site-packages (from tensorflow==2.5.0) (0.37.1)
Requirement already satisfied: tensorflow-estimator<2.6.0,>=2.5.0rc0 in c:\users\dell\anaconda3\lib\site-packages (from tensorflow==2.5.0) (2.5.0)
Requirement already satisfied: keras-preprocessing~=1.1.2 in c:\users\dell\anaconda3\lib\site-packages (from tensorflow==2.5.0) (1.1.2)
Requirement already satisfied: keras-nightly~=2.5.0.dev in c:\users\dell\anaconda3\lib\site-packages (from tensorflow==2.5.0) (2.5.0.dev2021032900)
Requirement already satisfied: typing-extensions~=3.7.4 in c:\users\dell\anaconda3\lib\site-packages (from tensorflow==2.5.0) (3.7.4.3)
Requirement already satisfied: flatbuffers~=1.12.0 in c:\users\dell\anaconda3\lib\site-packages (from tensorflow==2.5.0) (1.12)
Requirement already satisfied: termcolor~=1.1.0 in c:\users\dell\anaconda3\lib\site-packages (from tensorflow==2.5.0) (1.1.0)
Requirement already satisfied: astunparse~=1.6.3 in c:\users\dell\anaconda3\lib\site-packages (from tensorflow==2.5.0) (1.6.3)
Requirement already satisfied: gast==0.4.0 in c:\users\dell\anaconda3\lib\site-packages (from tensorflow==2.5.0) (0.4.0)
Requirement already satisfied: grpcio~=1.34.0 in c:\users\dell\anaconda3\lib\site-packages (from tensorflow==2.5.0) (1.34.1)
Requirement already satisfied: numpy~=1.19.2 in c:\users\dell\anaconda3\lib\site-packages (from tensorflow==2.5.0) (1.19.5)
Requirement already satisfied: wrapt~=1.12.1 in c:\users\dell\anaconda3\lib\site-packages (from tensorflow==2.5.0) (1.12.1)
```

## Fig21 LSTM Code

Using Python's pip package management, the programme installs certain versions of TensorFlow, TensorFlow-GPU, OpenCV-Python, MediaPipe, Scikit-learn, and Matplotlib, supplying the necessary dependencies for activities including machine learning, computer vision, and data visualisation

```
import cv2
import numpy as np
import os
from matplotlib import pyplot as plt
import time
import mediapipe as mp
```

### 2. Keypoints using MP Holistic

```
mp_holistic = mp.solutions.holistic # Holistic model
mp_drawing = mp.solutions.drawing_utils # Drawing utilities
```

```
def mediapipe_detection(image, model):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # COLOR CONVERSION BGR 2 RGB
    image.flags.writeable = False # Image is no longer writeable
    results = model.process(image) # Make prediction
    image.flags.writeable = True # Image is now writeable
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR) # COLOR CONVERSION RGB 2 BGR
    return image, results
```

```
def draw_landmarks(image, results):
    mp_drawing.draw_landmarks(image, results.face_landmarks, mp_holistic.FACEMESH_TESSELATION) # Draw face connections
    mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_holistic.POSE_CONNECTIONS) # Draw pose connections
    mp_drawing.draw_landmarks(image, results.left_hand_landmarks, mp_holistic.HAND_CONNECTIONS) # Draw left hand connections
    mp_drawing.draw_landmarks(image, results.right_hand_landmarks, mp_holistic.HAND_CONNECTIONS) # Draw right hand connections
```

## Fig22 LSTM Code

These lines import the MediaPipe library's holistic model and drawing tools. The holistic model is represented by the `mp_holistic` object, and `mp_drawing` offers tools for creating connections and markers.

Using an input image and the holistic model, this function first converts the image to RGB format (as required by MediaPipe), then uses the model to predict the outcome before returning the resultant image to its original BGR format. Both the processed image and the prediction's outcomes are returned by the function.

The function uses the drawing utilities to overlay landmarks and connections on the processed image for the face, pose, left hand, and right hand; the specific connections to be drawn for each part are defined by the constants `mp_holistic.FACEMESH_TESSELATION`, `mp_holistic.POSE_CONNECTIONS`, and `mp_holistic.HAND_CONNECTIONS`.

```
] def draw_styled_landmarks(image, results):
    # Draw face connections
    mp_drawing.draw_landmarks(image, results.face_landmarks, mp_holistic.FACEMESH_TESSELATION,
                              mp_drawing.DrawingSpec(color=(80,110,10), thickness=1, circle_radius=1),
                              mp_drawing.DrawingSpec(color=(80,256,121), thickness=1, circle_radius=1)
                              )
    # Draw pose connections
    mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_holistic.POSE_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(80,22,10), thickness=2, circle_radius=4),
                              mp_drawing.DrawingSpec(color=(80,44,121), thickness=2, circle_radius=2)
                              )
    # Draw left hand connections
    mp_drawing.draw_landmarks(image, results.left_hand_landmarks, mp_holistic.HAND_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(121,22,76), thickness=2, circle_radius=4),
                              mp_drawing.DrawingSpec(color=(121,44,250), thickness=2, circle_radius=2)
                              )
    # Draw right hand connections
    mp_drawing.draw_landmarks(image, results.right_hand_landmarks, mp_holistic.HAND_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(245,117,66), thickness=2, circle_radius=4),
                              mp_drawing.DrawingSpec(color=(245,66,230), thickness=2, circle_radius=2)
                              )
```

### Fig23 LSTM Code

This function takes the image that has been processed (`image`) and the predictions made by the holistic model (`results`). It overlays stylized landmarks and connections for various body sections on the image using the `mp_drawing.draw_landmarks` function. `mp_drawing` is used to specify the styling. `DrawingSpec` instances that let you adjust the circle radius, colour, and thickness for every kind of connection.

### Camera input Code:

```

cap = cv2.VideoCapture(0)
# Set mediapipe model
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:
    while cap.isOpened():

        # Read feed
        ret, frame = cap.read()

        # Make detections
        image, results = mediapipe_detection(frame, holistic)
        print(results)

        # Draw Landmarks
        draw_styled_landmarks(image, results)

        # Show to screen
        cv2.imshow('OpenCV Feed', image)

        # Break gracefully
        if cv2.waitKey(10) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
<class mediapipe.python.solution_base.SolutionOutputs >
<class 'mediapipe.python.solution_base.SolutionOutputs'>

```

**Fig24 LSTM Code**

In order to record video frames from the default camera (index 0), the line initialises a video capture object (cap). Here, a holistic model (mp\_holistic.Holistic) is created using a with statement. The lowest levels of confidence needed for initial detection and subsequent tracking are indicated by the min\_detection\_confidence and min\_tracking\_confidence parameters, which are both set to 0.5. The video capture (cap) is kept open as long as this starts a while loop. Using cap.read(), it reads a frame from the video capture inside the loop. The captured image is contained in the frame, and the return value ret indicates if the frame was properly read.

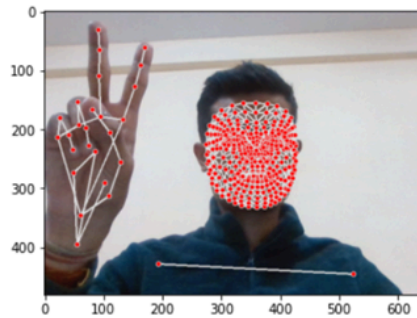
In order to process the collected frame using the holistic model (holistic), the mediapipe\_detection function is invoked. It gives back the processed image (image) and the predictions made by the holistic model (results). landmarks and linkages for the face, posture, left hand, and right hand are included in the results. For the purpose of analysis or troubleshooting, the findings are printed. Based on the outcomes of the holistic model, the draw\_styled\_landmarks function is invoked to overlay styled landmarks and connections on the image. Use of cv2.imshow displays the processed image in a window called 'OpenCV Feed'. This looks for a significant news story. The loop ends, the video capture is released, and OpenCV windows are destroyed if the key hit is 'q'.

Once the loop is exited, the video capture is released, and all OpenCV windows are closed.

```
In [8]: draw_landmarks(frame, results)
```

```
In [9]: plt.imshow(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
```

```
Out[9]: <matplotlib.image.AxesImage at 0x267eab51a30>
```



**Fig25 LSTM Code**

The `draw_landmarks` function, which is likely a bespoke function that overlays connections and landmarks on the input frame depending on the output of a pose estimate model, is called by this line. This line shows the processed frame using Matplotlib. The OpenCV frame's BGR format is converted to RGB using `cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)`, since Matplotlib requires RGB-formatted images. After that, `plt.imshow()` receives the RGB frame that was produced.

### Using the CNN Model(Code):

```
# Importing the Libraries Required
import os
import string

# Creating the directory Structure

if not os.path.exists("dataSet"):
    os.makedirs("dataSet")

if not os.path.exists("dataSet/trainingData"):
    os.makedirs("dataSet/trainingData")

if not os.path.exists("dataSet/testingData"):
    os.makedirs("dataSet/testingData")

# Making folder 0 (i.e blank) in the training and testing data folders respectively
for i in range(0):
    if not os.path.exists("dataSet/trainingData/" + str(i)):
        os.makedirs("dataSet/trainingData/" + str(i))

    if not os.path.exists("dataSet/testingData/" + str(i)):
        os.makedirs("dataSet/testingData/" + str(i))

# Making Folders from A to Z in the training and testing data folders respectively
for i in string.ascii_uppercase:
    if not os.path.exists("dataSet/trainingData/" + i):
        os.makedirs("dataSet/trainingData/" + i)
```

**Fig26** (It is used to create a folder for data storage.)

```

1 # Importing the Libraries Required
2
3 import os
4 import string
5
6 # Creating the directory Structure
7
8 if not os.path.exists("dataSet"):
9     os.makedirs("dataSet")
10
11 if not os.path.exists("dataSet/trainingData"):
12     os.makedirs("dataSet/trainingData")
13
14 if not os.path.exists("dataSet/testingData"):
15     os.makedirs("dataSet/testingData")
16
17 # Making folder 0 (i.e blank) in the training and testing data folders respectively
18 for i in range(0):
19     if not os.path.exists("dataSet/trainingData/" + str(i)):
20         os.makedirs("dataSet/trainingData/" + str(i))
21
22     if not os.path.exists("dataSet/testingData/" + str(i)):
23         os.makedirs("dataSet/testingData/" + str(i))
24
25 # Making Folders from A to Z in the training and testing data folders respectively
26
27 for i in string.ascii_uppercase:
28     if not os.path.exists("dataSet/trainingData/" + i):
29         os.makedirs("dataSet/trainingData/" + i)
30

```

Fig27(a)

```

import cv2
import numpy as np
import os

# Creating and Collecting Training Data

mode = 'testingData'
directory = 'dataSet/' + mode + '/'
minValue = 35

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

while True:
    _, frame = capture.read()

    # Simulating mirror Image

    frame = cv2.flip(frame, 1)

    # Getting count of existing images

    count = {
        'zero': len(os.listdir(directory+"0")),
        'a': len(os.listdir(directory+"A")),
        'b': len(os.listdir(directory+"B")),
        'c': len(os.listdir(directory+"C")),
    }

```

Fig27(b) (It is used to store the images as training and testing data)

```

import numpy as np
import cv2

minValue = 70

def func(path):
    frame = cv2.imread(path)

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

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

```

**Fig28** (It is used to pre process the image that is taken as input)

```

import numpy as np
import cv2
import os, sys
import time
import operator

from string import ascii_uppercase

import tkinter as tk
from PIL import Image, ImageTk

from hunspell import Hunspell
import enchant

from keras.models import model_from_json

os.environ["THEANO_FLAGS"] = "device=cuda, assert_no_cpu_op=True"

#Application :
class Application:

    def __init__(self):

        self.hs = Hunspell('en_US')
        self.vs = cv2.VideoCapture(0)
        self.current_image = None

```

**Fig29** (This code helps in providing the interface which takes input from the camera , detects the signs and provides the suggested words which can come after a particular word.)

#### 3.4.4.1 Tools Used:

**Camera:** The main visual input source for hand gesture detection systems is cameras. They record hand movements in real time, allowing the system to identify and categorise the movements based on pertinent data extracted.

**Keras:** Strong technologies for hand gesture recognition projects are TensorFlow and Keras. Building neural networks is made easier with Keras' high-level API, which makes it easier to create and train gesture detection models. The underlying computational backend, TensorFlow, provides scalable and effective performance for processing hand gesture data.

**Matplotlib:** A Python package for data visualisation is called Matplotlib. It can be used to visualise the extracted features, such as contour points, edge features, and key points, in hand gesture recognition applications. This can assist in figuring out possible trends and comprehending the distribution of the data.

**Scikit-learn:** For hand gesture detection applications, the scikit-learn library offers an extensive collection of machine learning techniques. These methods, which are excellent for classification tasks like gesture recognition, include support vector machines (SVMs), and neural networks.

**Tkinter:** Tkinter is a common Python GUI (Graphical User Interface) toolkit that may be used to create intuitive user interfaces for data visualisation, model interaction, and result presentation in a variety of machine learning applications.

**[27]Hunspell\_suggest:** It's used to provide appropriate substitutes for each (incorrect) word entered, and it shows a list of terms that match the current word so the user can choose one to add to the statement. Both spelling errors are decreased and complicated word prediction is aided by this.

**OpenCV:** Projects involving hand gesture detection heavily rely on OpenCV, an open-source computer vision library. Its extensive feature set makes it an invaluable tool for creating reliable hand gesture recognition systems by facilitating effective image processing, hand detection, feature extraction, and gesture categorization.

**LSTM:** Recurrent neural networks (RNNs) with long short-term memory (LSTM) networks are especially well-suited for hand gesture identification tasks because of their capacity to

capture temporal information in sequential data and long-range relationships. The sequences of hand postures, orientations, and movements that make up hand gestures are intrinsically dynamic. Because LSTM networks are so good at modelling these temporal sequences, they are a very attractive option for hand gesture identification.

### **3.5 KEY CHALLENGES**

#### **1. Data Collection and Labelling:**

**Data Variation:** To train a balanced system, a diverse dataset covering a large range of hand positions, orientations, lighting conditions must be gathered.

**Labelling Accuracy:** Proper training of the classification model depends on accurate and consistent labelling of the gesture present in the dataset.

**Data Volume:** To achieve high recognition accuracy, a considerable amount of data must be collected to represent the variety of hand motions.

#### **2. Hand Segmentation and Detection:**

**Background Complexity:** It's critical to manage intricate backdrops and occlusions that may impede hand segmentation and detection.

**Lighting Variations:** Reliable hand detection depends on the system's resilience to changes in lighting, such as shadows and uneven brightness.

#### **3. Feature Recognition and Classification of Gestures:**

**Relevance of elements:** Accurate gesture categorization depends on the selection and extraction of pertinent elements that accurately depict the shape, alignment, and motion of the hand.

#### **4. Performance in Real Time:**

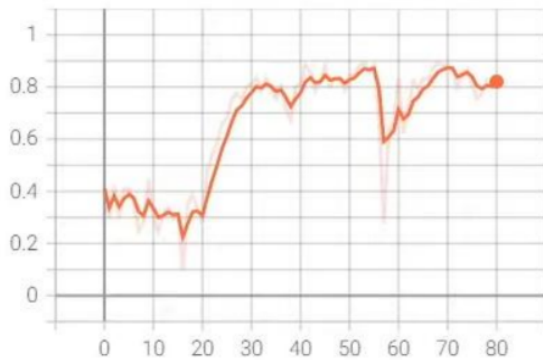
**Computational Efficiency:** To achieve real-time gesture recognition, algorithms must be optimised and processing times reduced.



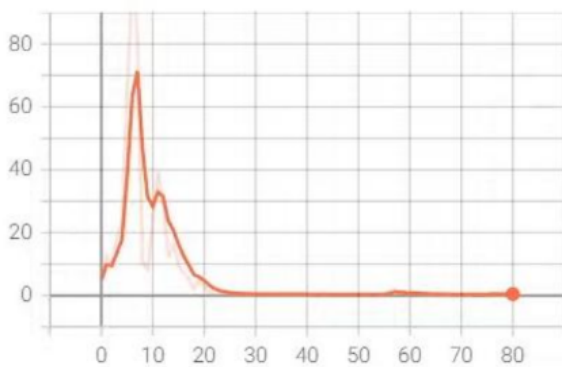
## CHAPTER 4 :TESTING

The dataset used for testing was taken from the original dataset where it was divided between training and testing data. The evaluation metrics we use is Accuracy. The LSTM model is working successfully(see figure).

Accuracy of 80% was achieved with the test dataset.



Graph 1. Epoch Categorical accuracy



Graph 2. Epoch Loss

Below is the confusion matrix for the code implementation of CNN model which includes two layers of algorithms for detection and classification of the different signs symbols.

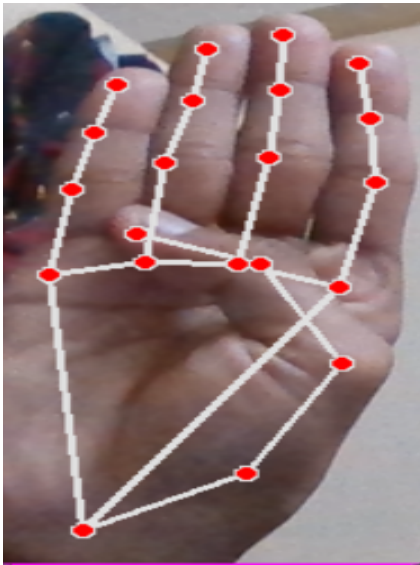
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
A	147	0	0	0	0	0	0	0	0	0	0	1	2	0	0	0	0	0	0	0	0	0	2	0	0
B	0	139	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0
C	0	0	152	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	145	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	152	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	135	0	0	0	0	0	4	0	0	0	0	0	1	0	0	2	10	0	0	0
G	0	0	0	0	0	0	150	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
H	1	0	0	0	0	0	7	143	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1
I	0	0	0	33	0	0	0	0	108	0	2	0	0	0	0	0	0	0	0	7	1	0	0	0	0
J	0	0	0	0	0	0	0	0	0	153	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	0	153	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	0	0	153	0	0	0	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	2	0	152	0	0	0	0	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	152	0	0	0	0	0	0	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	154	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	153	0	0	0	0	0	0	0	0	0
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	147	1	0	0	0	0	0	0	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	150	0	0	0	0	0	0	0
S	0	0	0	0	1	0	0	0	0	0	0	0	0	1	10	0	0	0	132	0	0	0	0	8	0
T	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	151	0	0	0	0	0
U	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	35	0	0	115	0	0	0	0
V	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	151	1	0	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	149	0	0
X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	148	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	151
Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

**Fig30** (Confusion Matrix for the CNN model that has more than 90% accuracy.)

# CHAPTER 5 : RESULTS AND EVALUATION

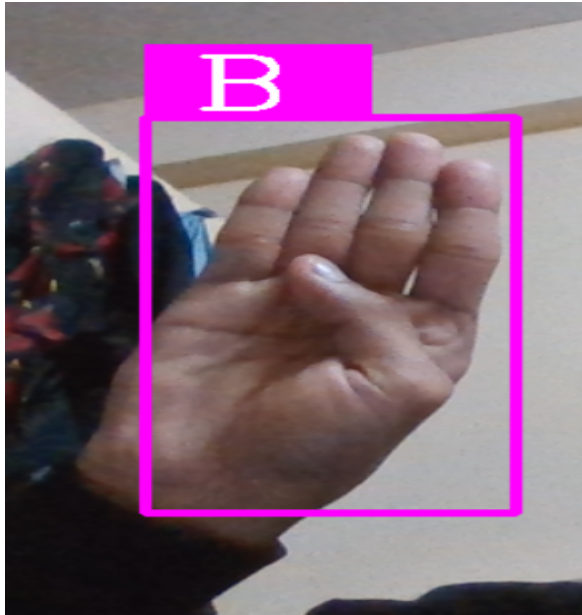
## Model 1(Pre-trained):

In this model, after the collection of images for dataset generation , the output window uses an inbuilt camera which detects the hand landmarks and detects the presence of the hand in the window.



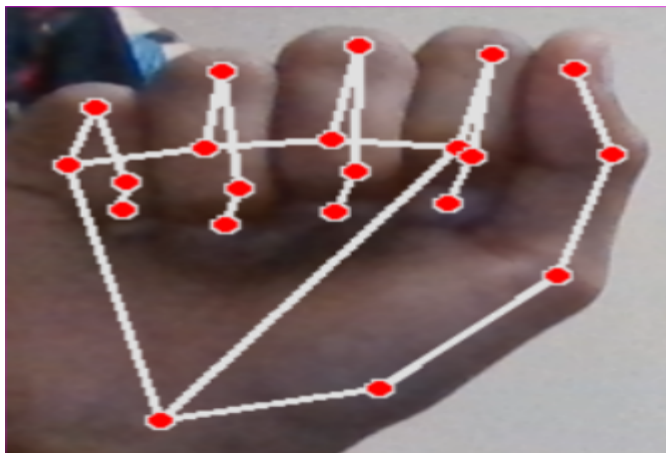
**Fig31** Results and Limitations

After the execution of the test file , it detects the hand gestures made in the output window and categorises them according to the corresponding accurate gesture.

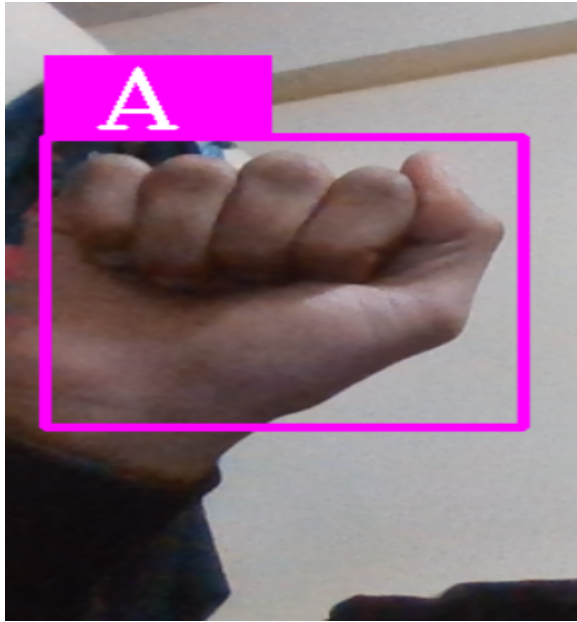


**Fig32** Results and Limitations

Here this is another one of the examples of a hand gesture being collecting the data and trying to accurately detect the gesture.



**Fig 33** Results and Limitations



**Fig34** Results and Limitations

**Limitation:**

Even if the self created dataset for the gesture detection is a good practice but due to the limited number of images it can gather it fails to show the diversity in real time applications and may detect some gesture other than the actual gesture as that gesture causing it to fail its usability. Here this is an example of 'V' gesture detection but it is showing it as a 'B' symbol.



**Fig35** Results and Limitations

**Model 2(LSTM):**

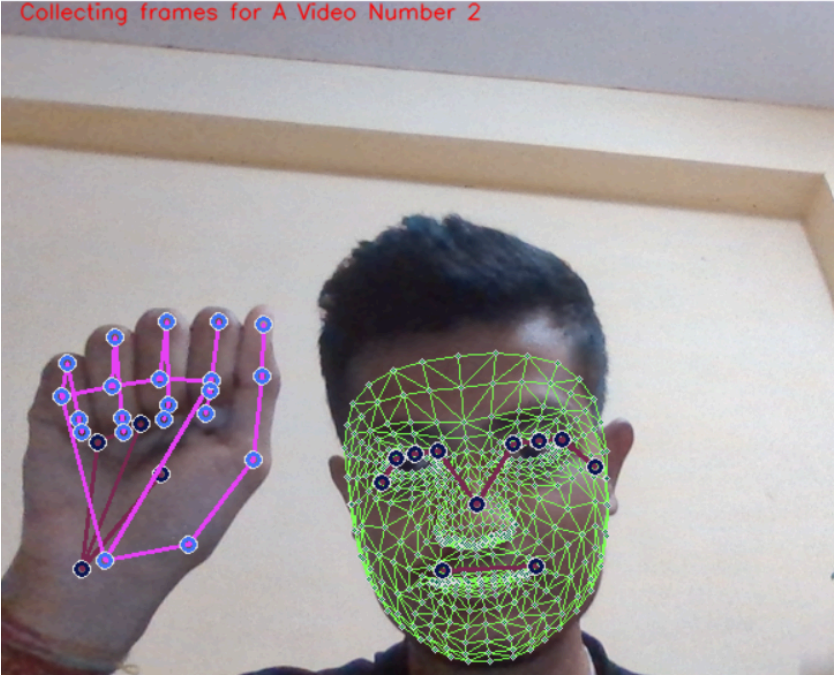
In this model we use a camera and take input from the camera and make 10 sequences for each of the symbols having different frame rates and save them in the folder in the .npy file.

Here is the image:

0.npy	11/30/2023 11:59 PM	NPY File	14 KB
1.npy	11/30/2023 11:59 PM	NPY File	14 KB
2.npy	11/30/2023 11:59 PM	NPY File	14 KB
3.npy	11/30/2023 11:59 PM	NPY File	14 KB
4.npy	11/30/2023 11:59 PM	NPY File	14 KB
5.npy	11/30/2023 11:59 PM	NPY File	14 KB
6.npy	11/30/2023 11:59 PM	NPY File	14 KB
7.npy	11/30/2023 11:59 PM	NPY File	14 KB
8.npy	11/30/2023 11:59 PM	NPY File	14 KB
9.npy	11/30/2023 11:59 PM	NPY File	14 KB
10.npy	11/30/2023 11:59 PM	NPY File	14 KB
11.npy	11/30/2023 11:59 PM	NPY File	14 KB

**Fig36** (.npy file for 30 frames of each sequence which is being given as input.)

Here is the image which shows the interface when we try to collect the data for the testing and training.



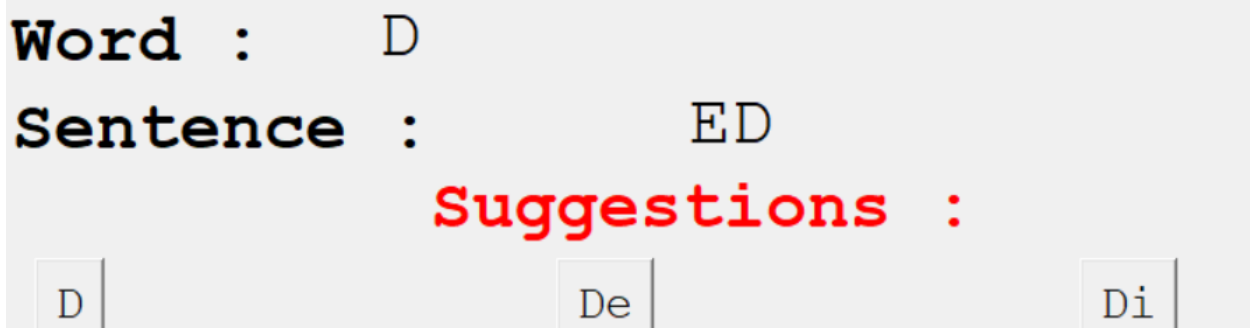
**Fig37** Input given after running the model.

**Model 3 (CNN):**



**Fig38**

It detects the symbol and makes a separate space for the symbol detected to be highlighted and then the detected symbol is stored in the word section.



**Fig39**

The detected word is then suggested with the possible number of words that can be combined next with the current word to create a sentence using the help of hunspell\_suggest python library.

**Table-2 Comparison between different Models**

Method	Accuracy (%)	Gesture Set Size	Data Modality	Model Type	Source
My Model	90	26	Images/Depth	CNN	
Dynamic Hand Gesture Recognition Using 3D-CNN and LSTM Networks	83-89	4	RGB-D	3D-CNN + LSTM	Tech Science



<p>A Compariso n of Bidirection al GRU and LSTM for Hand Gesture Recognitio n Using Leap Motion[2]</p>	<p>96.97- 97.28</p>	<p>10</p>	<p>Leap Motion Data</p>	<p>Bi-GRU/Bi -LSTM</p>	<p>IEEE Xplore</p>
<p>Dynamic Gesture Recognitio n Model Based on Millimeter- Wave Radar With ResNet-18 and LSTM[3]</p>	<p>92.55</p>	<p>11</p>	<p>Millimeter-Wa ve Radar Signals</p>	<p>ResNet-18 + LSTM</p>	<p>Frontiers in Robotics and AI</p>

LSTM Recurrent Neural Network for Hand Gesture Recognitio n Using EMG Signals[4]	99	5	EMG Signals	LSTM-RN N	MDPI
---	----	---	-------------	--------------	------

# CHAPTER 6 :CONCLUSION AND FUTURE SCOPE

## 6.1 CONCLUSION

In summary, significant findings and advancements in the field of sign language recognition have resulted from our extensive effort on Detection for Sign Language Gestures. We achieved this by using deep learning models and a dataset that we had developed ourselves. This section summarises the major findings, restrictions, and contributions to the field.

### 6.1.1 Key Findings:

#### 6.1.1.1 Effective Utilisation of Self-Created Dataset:

The study successfully demonstrated the benefits of utilising a user-developed dataset, allowing for a more in-depth and sophisticated explanation of sign language motions. This approach ensured that a range of expressions were covered and improved the model's adaptability to different signing styles.

#### 6.1.1.2 Real-time Detection:

Real-time action identification for sign language motions was made possible by the use of techniques like TensorFlow Keras and Mediapipe. Real-world applications of this field are crucial, especially in the development of assistive devices and communication tools for individuals with hearing impairments.

#### 6.1.1.3 Robustness to External Factors:

The project addressed noise immunity by implementing techniques to enhance the model's robustness in real-world scenarios. Considering backdrop clutter and lighting conditions led to a more reliable action detection system.

### 6.1.2 Limitations:

#### 6.1.2.1 Dataset Size and Variability:

The self-created dataset proved to be helpful, but the model's ability to generalise over a range of sign languages and user variations might be constrained by its size and diversity. Inclusion of the numbers for the detection is still one of the constraints.

#### **6.1.2.2 Real-world Variability:**

There is still difficulty in accounting for real-world unpredictability despite efforts to do so. Continued research and improvement are necessary to guarantee the model's flexibility in response to shifting environmental conditions.

### **6.2 FUTURE SCOPE:**

The course of this project sets up a noteworthy and expansive future. By delving deeper into these tactical options, researchers can improve the identification of sign language and create more inclusive and adaptive technology for those with hearing impairments.

#### **6.2.1 Creation of a Larger Dataset:**

Expanding the dataset presents a significant opportunity to capture the multifaceted nature and diversity of sign languages. Working together with sign language communities, linguistic experts, and cultural representatives can enable researchers to create a dataset that encompasses dialectical nuances, regional variations, and individual signing styles. This comprehensive dataset will enhance the model's accuracy and broaden its appeal to a wider range of user groups.

#### **6.2.2 Addition of Additional Modalities:**

Incorporating additional modalities beyond visual cues enriches the model's contextual understanding. Integrating sign language elements such as hand gestures, facial expressions, and even background information like the discourse's tone and cadence can facilitate a more refined interpretation. This holistic approach elevates the technology's responsiveness and intuitiveness, better aligning it with the intricacies of human communication.

#### **6.2.3 Enhancement of Real-time Processing:**

To achieve seamless real-time performance, the model must be optimised for low latency and efficient frame rate handling. Researchers can delve into cutting-edge edge computing methods by employing wearables and smartphones for real-time sign language recognition. This optimization is essential for applications demanding real-time interaction, such as assistive technology or live translations.

#### **6.2.4 Cross-linguistic Adaptability:**

Attaining cross-linguistic versatility necessitates a rigorous process of training the model on multiple sign languages. This undertaking entails comprehending not only the linguistic intricacies but also the cultural backdrop of each sign language. The objective is to develop a universal model that can seamlessly adapt to the diverse signing practices prevalent around the world.

#### **6.2.5 User-specific Customization:**

Customised models can be created to accommodate each user's unique preferences while accounting for each signer's unique characteristics. By adapting to the distinct features of each person's signing motion, machine learning algorithms can improve intuitiveness and user-friendliness. This customisation improves overall communication effectiveness and fosters deeper interactions between users and technology.

#### **6.2.6 Evaluation in Real-world Environments:**

To enhance the model's robustness, it must undergo rigorous testing in real-world scenarios. Extensive evaluations should be carried out in diverse environments, encompassing varying lighting conditions, intricate backgrounds, and individual signing variations. This hands-on testing replicates the irregularities of everyday interactions, guaranteeing the model's dependability in practical settings.

## REFERENCES

- [1] N. Adaloglou et al., "A Comprehensive Study on Deep Learning-Based Methods for Sign Language Recognition," in *IEEE Transactions on Multimedia*, vol. 24, pp. 1750-1762, 2022, doi: 10.1109/TMM.2021.3070438.
- [2] Kothadiya, Deep, et al. "Deepsign: Sign Language Detection and Recognition Using Deep Learning." *Electronics*, vol. 11, no. 11, 3 June 2022, p. 1780, <https://doi.org/10.3390/electronics11111780>.
- [3] Z. Pei, Y. Wang and L. Han, "Hand Gesture Recognition with Color Descriptors," *2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, Chongqing, China, 2020, pp. 1822-1826, doi: 10.1109/ITAIC49862.2020.9338985.
- [4] R. Golovanov, D. Vorotnev and D. Kalina, "Combining Hand Detection and Gesture Recognition Algorithms for Minimizing Computational Cost," *2020 22th International Conference on Digital Signal Processing and its Applications (DSPA)*, Moscow, Russia, 2020, pp. 1-4, doi: 10.1109/DSPA48919.2020.9213273.
- [5] J. Liu, K. Furusawa, T. Tateyama, Y. Iwamoto and Y. -W. Chen, "An Improved Hand Gesture Recognition with Two-Stage Convolution Neural Networks Using a Hand Color Image and its Pseudo-Depth Image," *2019 IEEE International Conference on Image Processing (ICIP)*, Taipei, Taiwan, 2019, pp. 375-379, doi: 10.1109/ICIP.2019.8802970.
- [6] R. Kabir, N. Ahmed, N. Roy and M. R. Islam, "A Novel Dynamic Hand Gesture and Movement Trajectory Recognition model for Non-Touch HRI Interface," *2019 IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE)*, Yunlin, Taiwan, 2019, pp. 505-508, doi: 10.1109/ECICE47484.2019.8942691.
- [7] J. Zhao, X. H. Li, J. C. D. Cruz, M. S. Verdadero, J. C. Centeno and J. M. Novelero, "Hand Gesture Recognition Based on Deep Learning," *2023 International Conference on Digital Applications, Transformation & Economy (ICDATE)*, Miri, Sarawak, Malaysia, 2023, pp. 250-254, doi: 10.1109/ICDATE58146.2023.10248500.

- [8] S. Y. Boulahia, E. Anquetil, F. Multon and R. Kulpa, "Dynamic hand gesture recognition based on 3D pattern assembled trajectories," *2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA)*, Montreal, QC, Canada, 2017, pp. 1-6, doi: 10.1109/IPTA.2017.8310146.
- [9] R. A. Bhuiyan, A. K. Tushar, A. Ashiquzzaman, J. Shin and M. R. Islam, "Reduction of gesture feature dimension for improving the hand gesture recognition performance of numerical sign language," *2017 20th International Conference of Computer and Information Technology (ICCIT)*, Dhaka, Bangladesh, 2017, pp. 1-6, doi: 10.1109/ICCITECHN.2017.8281833.
- [10] L. Tiantian, S. Jinyuan, L. Runjie and G. Yingying, "Hand gesture recognition based on improved histograms of oriented gradients," *The 27th Chinese Control and Decision Conference (2015 CCDC)*, Qingdao, China, 2015, pp. 4211-4215, doi: 10.1109/CCDC.2015.7162670.
- [11] R. Hartanto, A. Susanto and P. I. Santosa, "Real time hand gesture movements tracking and recognizing system," *2014 Electrical Power, Electronics, Communications, Control and Informatics Seminar (EECCIS)*, Malang, Indonesia, 2014, pp. 137-141, doi: 10.1109/EECCIS.2014.7003734.
- [12] N. Adaloglou *et al.*, "A Comprehensive Study on Deep Learning-Based Methods for Sign Language Recognition," in *IEEE Transactions on Multimedia*, vol. 24, pp. 1750-1762, 2022, doi: 10.1109/TMM.2021.3070438.
- [13] "Understanding of LSTM Networks." *GeeksforGeeks*, 10 May 2020, [www.geeksforgeeks.org/understanding-of-lstm-networks/](http://www.geeksforgeeks.org/understanding-of-lstm-networks/).
- [14] "MediaPipe Solutions API Reference." *Google for Developers*, [developers.google.com/mediapipe/api/solutions](https://developers.google.com/mediapipe/api/solutions). Accessed 1 Dec. 2023.
- [15] "How Is the LSTM RNN Forget Gate Calculated?" *Data Science Stack Exchange*, [datascience.stackexchange.com/questions/32217/how-is-the-lstm-rnn-forget-gate-calculated](https://datascience.stackexchange.com/questions/32217/how-is-the-lstm-rnn-forget-gate-calculated).

- [16]Cheok MJ, Omar Z, Jaward MH. A review of hand gesture and sign language recognition techniques. *Int J Mach Learn Cybern.* 2019;10(1):131–53.
- [17]Griffin, R. W. (2021). *Management*. Cengage Learning. Hampshire, UK. 11.
- [18]Pranjal Srivastava. “Essentials of Deep Learning : Introduction to Long Short Term Memory.” *Analytics Vidhya*,23Dec.2017, [www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/](http://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/).
- [19]Chen L, Wang F, Deng H, Ji K. A survey on hand gesture recognition. *Intern Conf Comput Sci Appl.* 2013;2013:313–6.
- [20]. Cheok MJ, Omar Z, Jaward MH. A review of hand gesture and sign language recognition techniques. *Int J Mach Learn Cybern.* 2019;10(1):131–53.
- [21]Chen,J. CS231A Course Project Final Report Sign Language Recognition with Unsupervised Feature Learning. 2012. Available online: [http://vision.stanford.edu/teaching/cs231a\\_autumn1213\\_internal/project/final/writeup/distributable/Chen\\_Paper.pdf](http://vision.stanford.edu/teaching/cs231a_autumn1213_internal/project/final/writeup/distributable/Chen_Paper.pdf) (accessed on 15 March 2022)
- [22]Yan, S. Understanding LSTM and Its Diagrams. Available online: <https://medium.com/mlreview/understanding-lstm-and-its-diagrams-37e2f46f1714>
- [23]Hochreiter, S.; Schmidhuber, J. Long Short-term Memory. *Neural Comput.* 1997, 9, 1735–1780
- [24]Liao, Y.; Xiong, P.; Min, W.; Min, W.; Lu, J. Dynamic Sign Language Recognition Based on Video Sequence with BLSTM-3D Residual Networks. *IEEE Access* 2019, 7, 38044–38054.
- [25]Muthu Mariappan, H.; Gomathi, V. Real-Time Recognition of Indian Sign Language. In *Proceedings of the International Conference on Computational Intelligence in Data Science*, Haryana, India, 6–7 September 2019.



[26]Mittal, A.; Kumar, P.; Roy, P.P.; Balasubramanian, R.; Chaudhuri, B.B. A Modified LSTM Model for Continuous Sign Language Recognition Using Leap Motion. IEEE Sens. J. 2019, 19, 7056–7063.

[27] <http://hunspell.github.io/>

ORIGINALITY REPORT

19%

SIMILARITY INDEX

13%

INTERNET SOURCES

11%

PUBLICATIONS

11%

STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to The Indian Institute Of Management And Engineering Society Student Paper	4%
2	<a href="http://www.ir.juit.ac.in:8080">www.ir.juit.ac.in:8080</a> Internet Source	1%
3	<a href="http://ijrpr.com">ijrpr.com</a> Internet Source	1%
4	Submitted to SASTRA University Student Paper	1%
5	Submitted to Indian Institute of Information Technology, Allahabad Student Paper	1%
6	<a href="http://www.ijsr.net">www.ijsr.net</a> Internet Source	1%
7	<a href="http://www.scilit.net">www.scilit.net</a> Internet Source	1%
8	Submitted to CSU, Chico Student Paper	1%
9	<a href="http://www.mdpi.com">www.mdpi.com</a>	

	Internet Source	<1 %
10	<a href="https://link.springer.com">link.springer.com</a> Internet Source	<1 %
11	<a href="https://ijarcce.com">ijarcce.com</a> Internet Source	<1 %
12	<a href="https://healthdocbox.com">healthdocbox.com</a> Internet Source	<1 %
13	<a href="https://ijsrcseit.com">ijsrcseit.com</a> Internet Source	<1 %
14	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	<1 %
15	<a href="https://rashedul.info">rashedul.info</a> Internet Source	<1 %
16	<a href="https://web.archive.org">web.archive.org</a> Internet Source	<1 %
17	<a href="https://www.ijnrd.org">www.ijnrd.org</a> Internet Source	<1 %
18	Submitted to The British College Student Paper	<1 %
19	Submitted to University of Wales Institute, Cardiff Student Paper	<1 %

**JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT**  
**PLAGIARISM VERIFICATION REPORT**

Date: .....

Type of Document (Tick):  PhD Thesis  M.Tech Dissertation/ Report  B.Tech Project Report  Paper

Name: \_\_\_\_\_ Department: \_\_\_\_\_ Enrolment No \_\_\_\_\_

Contact No. \_\_\_\_\_ E-mail. \_\_\_\_\_

Name of the Supervisor: \_\_\_\_\_

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): \_\_\_\_\_

**UNDERTAKING**

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

**Complete Thesis/Report Pages Detail:**

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

**FOR DEPARTMENT USE**

We have checked the thesis/report as per norms and found **Similarity Index** at .....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor)

Signature of HOD

**FOR LRC USE**

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	<ul style="list-style-type: none"> <li>• All Preliminary Pages</li> <li>• Bibliography/Images/Quotes</li> <li>• 14 Words String</li> </ul>		Word Counts	
<b>Report Generated on</b>			Character Counts	
		<b>Submission ID</b>	Total Pages Scanned	
			File Size	

Checked by  
Name & Signature

Librarian

**Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at [plagcheck.juit@gmail.com](mailto:plagcheck.juit@gmail.com)**

